Title
Catalytic Terpene Polycyclization Reactions and Structural Determination of Small Molecules by MicroED

Author
Burch, Jessica Elizabeth

Publication Date
2021

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Catalytic Terpene Polycyclization Reactions and Structural Determination of Small Molecules by MicroED

A dissertation submitted in partial satisfaction of the

Requirements for the degree Doctor of Philosophy

in Chemistry

by

Jessica Elizabeth Burch

2021

ABSTRACT OF THE DISSERTATION

Catalytic Terpene Polycyclization Reactions and Structural

Determination of Small Molecules by MicroED

by

Jessica Elizabeth Burch

Doctor of Philosophy in Chemistry

University of California, Los Angeles, 2021

Professor Hosea Martin Nelson, Co-Chair

Professor Yi Tang, Co-Chair

This dissertation describes the development of Lewis acid-based methodology to access polycyclized sesquiterpenes and diterpenes through simple catalytic methods. This challenging biomimetic reaction is difficult to replicate in synthesis and remains understudied. In addition, this dissertation explores the application and development of an electron diffraction technique for the elucidation of small molecules in an efficient manner. Overall, this work seeks to push the boundaries of synthesis and analysis of complex structures through development of synthetic and analytical methods described herein.

Chapter One is a brief overview of the current state of research on biomimetic terpene cyclization reactions and history of transmission electron microscopy techniques leading up to the development of microcrystal electron diffraction. This chapter serves as a prelude to the remaining chapters and will be referenced throughout this dissertation.

Chapter Two describes our efforts in the development of a lithium-weakly coordinating anion-mediated polycyclization reaction to generate polycyclic terpenes from acyclic fluorides precursors. The analysis of complex mixtures and early efforts to characterize these species utilizing microcrystal electron diffraction is discussed.

Chapter Three discusses our investigations into development and application of microcrystal electron diffraction to solve small molecule structural problems in chemistry. The analysis of pharmaceuticals and natural products is described, and the application of an automated data processing procedure is explored.

Chapter Four highlights an ongoing effort in our research group to create automated tools to make microcrystal electron diffraction a more practical and efficient tool for the

elucidation of small molecule structures. The development of Python automation scripts for data collection and processing is discussed.

The dissertation of Jessica Elizabeth Burch is approved.

Neil Kamal Garg

Patrick G. Harran

Yi Tang, Committee Co-Chair

Hosea Martin Nelson, Committee Co-Chair

University of California, Los Angeles

2021

This dissertation is dedicated to my mom, who taught me

to love learning from a young age.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER ONE

CHAPTER TWO

LIST OF TABLES

LIST OF ABBREVIATIONS

α = alpha

β = beta

br = broad

°C = degrees Celsius

cryoEM = cryo-electron microscopy

cryoET = cryo-electron tomography

δ = delta

d = doublet

dd = doublet of double

dr = diastereomeric ratio

δ = chemical shift

DCM = dichloromethane

EM = electron microscopy

ESI = electrospray ionization

Et = ethyl

equiv = equivalent

F_{20} = tetrakis(pentafluorophenyl)borate

γ = gamma

g = gram(s)

h = hour(s)

HMDS = hexamethyldisilazane

HPLC = high performance liquid chromatography

HRMS = high resolution mass spectroscopy

HT = head-to-tail

Hz = Hertz

IR = infrared spectroscopy

i-Pr = isopropyl

J = coupling constant

K = kelvin

L = liter

Li = lithium

LDA = lithium diisopropylamide

m = multiplet

m = meta

M = molar

m/z = mass to charge ratio

μ = micro

Me = methyl

MeCN = acetonitrile

MHz = megahertz

microED = microcrystal electron diffraction

min = minutes

mol = mole(s)

MOM = methoxymethyl ether

mp = melting point

NMR = nuclear magnetic resonance

o = ortho

ppm = parts per million

Pr = propyl

q = quartet

rt = room temperature

s = singlet

t = triplet

TEM = transmission electron microscope

temp = temperature

TES = triethylsilyl

Tf = trifluoromethanesulfonyl

TFA = trifluoroacetic acid

TH = tail-to-head

TLC = thin layer chromatography

UV = ultraviolet

ACKNOWLEDGEMENTS

I'd first like to thank my advisor, Prof. Hosea M. Nelson. Hosea's excitement for tackling challenging, interdisciplinary problems had a huge impact on how I view and perform science. The biggest lesson I learned from my time in the Nelson lab is to be unafraid of trying difficult experiments or projects, or of learning skillsets outside your discipline. Also known as "bet on yourself"!

I would like to thank my committee members, Prof. Yi Tang, Prof. Patrick Harran, and Prof. Neil Garg, for their support and advice throughout my graduate studies. Additionally, I would like to thank my undergraduate advisors, Prof. George Negrete and Prof. Oleg Larionov, as well as Dr. Gail Taylor, for encouraging and preparing me for graduate school.

I would like to thank Dr. Alex Bagdasarian, Dr. Brian Shao, Dr. Sydnee Green, and Dr. Stasik Popov for being great mentors and answering all my annoying questions when I joined the lab! For the electron microscopy project, I first have to thank Lee Joon Kim and Duilio Cascio, for their incredible patience and willingness to share their knowledge. I'd like to thank Prof. Jose A. Rodriguez for sharing his expertise and excitement for solving interesting structures. I'd also like to thank Dr. Matt Asay for his help getting started on the project, and Chris Jones for establishing this exciting project in our lab. I'd also like to thank our collaborators, including the Schroeder lab, Dr. Kyle Quasdorf, and Dr. Bing Shi, for sending us exciting compounds to study by microED. To Ben, Chloe, Sepand, Woojin, and Steven, thank you for being awesome labmates and friends over the past few years. I also couldn't have made it through grad school without my friends and wonderful scientists, Dr. Melissa Hardy and Ishika Saha.

Lastly, I’d like to thank my family for their support and words of wisdom. Even though I can tell they instantly regret asking “what is your research on?”, they still let me talk for way too long about plant metabolites and tiny crystals.

BIOGRAPHICAL SKETCH

Education:

University of California, Los Angeles, CA

- Ph.D. Candidate in Organic Chemistry
- Advanced to Candidacy, September 2019
- Current GPA: 3.71/4.00

University of Texas at San Antonio, San Antonio, TX

- Bachelor of Science in Chemistry, December 2015
- Undergraduate Researcher February 2015 – December 2016

Professional and Academic Experience:

Graduate Research Assistant: University of California, Los Angeles, CA

- July 2017 – present; Advisor: Prof. Hosea M. Nelson
- Advanced small molecule-catalyzed cationic polycyclization reactions
- Utilized microcrystal electron diffraction (microED) to elucidate natural product and pharmaceutical small molecule structures
- Developed Python automation regime to facilitate automated collection and data processing in small molecule microED
- Propelled projects forward by performing reaction and substrate design

Undergraduate Research Assistant: UTSA, San Antonio, TX

- May 2013 – December 2015; Advisors: Prof. Oleg Larionov and Prof. George Negrete
- Synthesized substrates for studies in borylation methodology
- Performed mechanistic studies on heterocycle dimerization
- Synthesized perylene dye derivatives for application in biophysical studies

Graduate Teaching Assistant: University of California, Los Angeles, CA

- Undergraduate organic chemistry labs (Fall 2017 – Spring 2018): Taught students organic reactions, mechanisms and laboratory techniques.

Honors and Awards:

- National Science Foundation Graduate Research Fellowship (2016 – 2021)
- NIH RISE Undergraduate Trainee (2014 – 2015)
- Highest Honors Program, UTSA Honors College (2015)

Publications:

1. Burch, J. E.; Wurz, R.; Smith, A.; Caille, S.; Walker, S. D.; Cee, V.; Rodriguez, J. A.; Gostovic, D.; Quasdorf, K.; Nelson, H. M. Putting MicroED to the test: an unabridged account of the evaluation of 30 diverse pharma compounds. *In Preparation.*

2. Curtis, B. J.; Kim, L. J.; Wrobel, C. J. J.; Eagan, J. M.; Smith, R. A.; Burch, J. E.; Le, H. H.; Artyukhin, A. B.; Nelson, H. M.; Schroeder, F. C. Identification of uric acid gluconucleosideascaroside conjugates in Caenorhabditis elegans by combining synthesis and MicroED. *Org. Lett.* **2020**, *22*, 6724–6728.

3. Burch, J. E.; Bagdasarian, A. L.; Hooshmand, T.; Nelson, H. M. Terpene tail-to-head polycyclization mediated by small molecule catalysts: a weakly coordinating anion approach. *ChemRxiv Pre-print* **2020**, doi.org/10.26434/chemrxiv.12719780.v2.

4. Mfuh, A. M.; Nguyen, V. T.; Chhetri, B.; Burch, J. E.; Doyle, J. D.; Nesterov, V. N.; Arman, H. D.; Larionov, O. V. Additive- and metal-free, predictably 1,2- and 1,3-regioselective, photoinduced dual C-H/C-X borylation of haloarenes. *J. Am. Chem. Soc.* **2016**, *138*, 8408–8411.

5. Stephens, D. E.; Lakey-Beitia, J.; Burch, J. E.; Arman, H. D.; Larionov, O. V. Mechanistic insights into the potassium tert-butoxide-mediated synthesis of N-heterobiaryls. *ChemComm* **2016**, *52*, 9945–9948.

6. Farooqi, M. J.; Penick, M. A.; Burch, J. E.; Negrete, G. R.; Brancaleon, L. Characterization of novel perylene diimides containing aromatic amino acid side chains *Spectrochimica Acta Part B.* **2015**, *153*, 124–131.

CHAPTER ONE

Terpene Tail-to-Head Polycyclization Reactions and Microcrystal Electron Diffraction of Small Molecules

1.1 Abstract

Many discoveries in science have been achieved through study of biochemical mechanisms and the incredible structural diversity of naturally occurring compounds. This chapter provides a summary of synthetic efforts towards the generation of terpene natural products through biomimetic polycyclization reactions, as well as the developments of an emerging electron microscopy technique, microcrystal electron diffraction (microED), for elucidation of small molecule structures.

1.2 Introduction

Advancements in fields that study small molecules, including drug discovery, biochemistry, and more, are dependent upon the strategies chemists have to build molecules and the analytical tools available to characterize them. Historically, many synthetic advancements have been realized through development of biomimetic strategies to access naturally occurring compounds. Terpene natural products have been of particular interest to the synthetic community given their structural diversity and complexity, as well as their biological activity. The complex biochemical transformation of acyclic terpene precursors to polycyclic natural products has been described as one of the most complex reactions in nature, presenting a significant synthetic and analytical challenge.

1.3 Cationic Polycyclization Reactions of Terpene Natural Products

For nearly 70 years, the study of terpene biosynthesis has inspired scientists across a wide array of disciplines. From enzymology and biophysics to computational and synthetic chemistry, many of the fundamental principles driving modern chemical science are rooted in studies of these remarkable enzymatic processes.[1–6] Early investigation of stereoselectivity in cation–π cyclizations by Stork and Eschenmoser led to the paradigm-shifting hypothesis that cyclase enzymes use stereoselective, extended cation–π cyclizations to create terpene-like backbones, as is exemplified in the conversion of squalene (**1.1**, Figure 1.1) to hopene (**1.3**) through polydecalin transition state **1.2**.[3,7] This groundbreaking hypothesis ultimately inspired the development of small-molecule

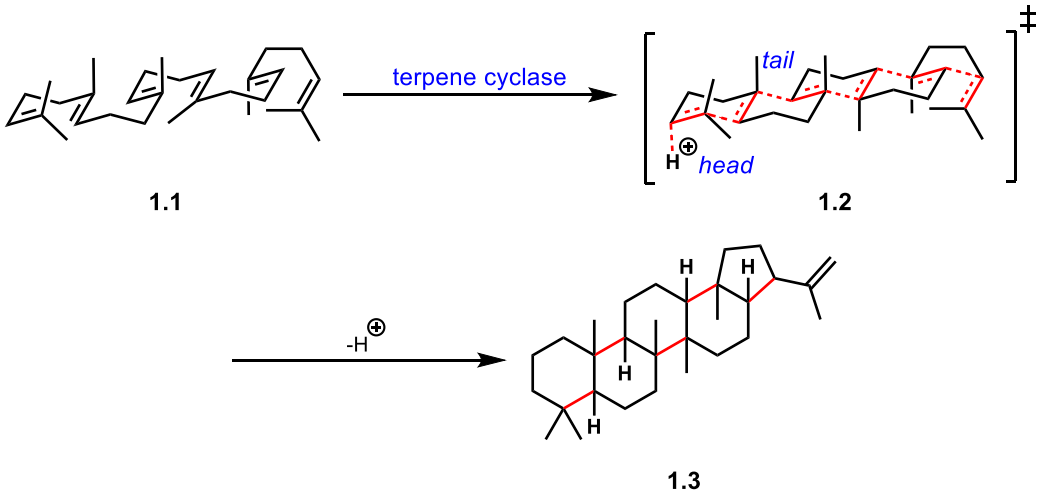


Figure 1.1 Polycyclization of squalene **1.1** through transition state **1.2** to produce hopene **1.3**

systems capable of producing polydecalin and steroid-like compounds with high levels of stereocontrol (**1.4** – **1.7**, Figure 1.2). Indeed, since the 1960s, Johnson, Corey, Ishihara, Yamamoto, Overman, and others have demonstrated that employment of a biomimetic head-to-tail (HT)

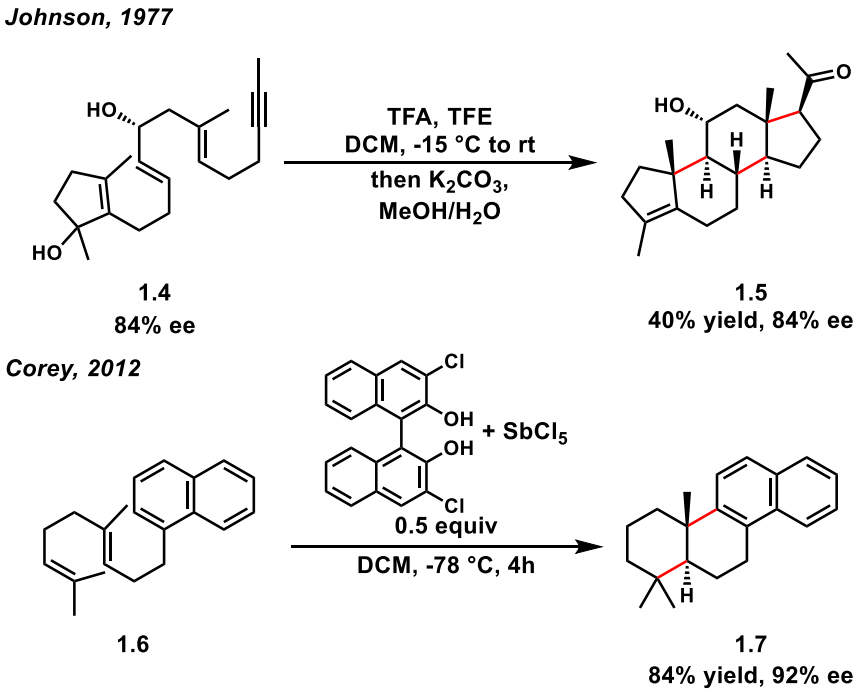


Figure 1.2 Small molecule-mediated HT cyclization reactions.

cation–π cyclization strategy in total synthesis provides a powerful platform to access polydecalin natural products.[8–12]

These classic studies highlight the value of using terpene biosynthesis as a sharpening stone for chemical synthesis and enzymology. However this symbiotic relationship is largely confined to HT processes that mimic type-II terpene cyclases to produce polydecalin frameworks from linear isoprenoids. Conversely, polycyclization reaction starting from linear isoprenoids that mimic Mg^{2+}-dependent type-I terpene cyclases (Figure 1.3) remain understudied in synthetic

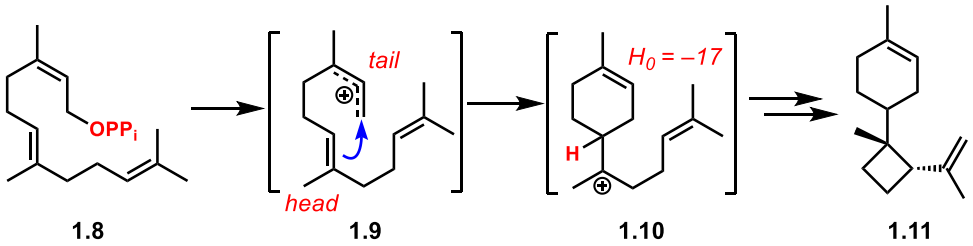


Figure 1.3 Biosynthetic tail-to-head polycyclization cascade of geranylgeranyl pyrophosphate **1.8** leading to generation of **1.11** through highly acidic cationic intermediates **1.9** and **1.10**.

organic chemistry; this is despite forging a larger and more structurally-diverse subset of polycyclic terpenes, including medicinally privileged natural products such as taxol (**1.12 – 1.17**, Figure 1.4).[7] These processes, originally coined tail-to-head (TH) cyclization by Shenvi and Pronin, proceed enzymatically *via* Mg^{2+}-mediated ionization of a phosphate head group (**1.8**, Figure 1.3 and **1.12**, Figure 1.4), followed by attack of an isoprenyl tail (**1.9**, Figure 1.3 and **1.13**, Figure 1.4) to ultimately form macrocyclic, medium, or small rings (**1.11**, Figure 1.3 and **1.16**, Figure 1.4).[13–15]

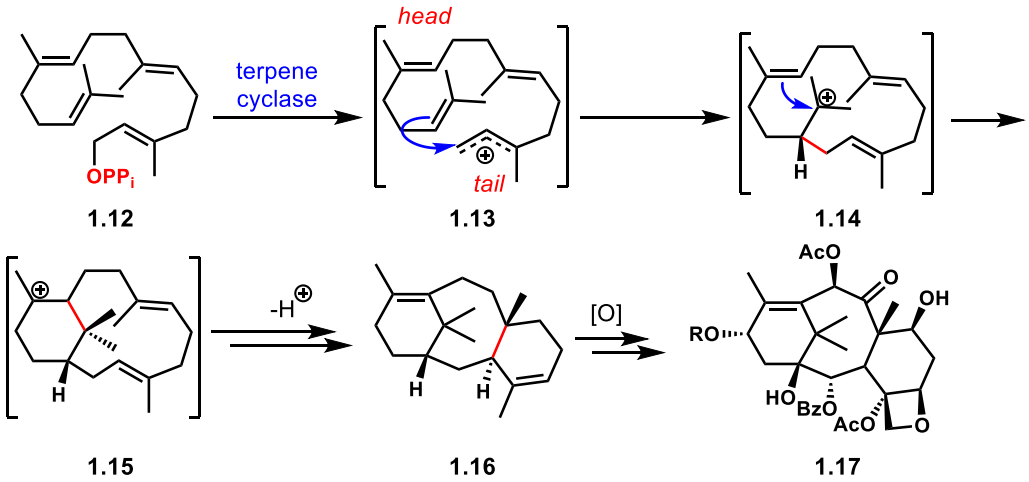


Figure 1.4 Biosynthetic tail-to-head polycyclization cascade of geranylgeranyl pyrophosphate **1.12** leading to generation of taxadiene **1.16**, the biosynthetic precursor to taxol **1.17**, through cationic intermediates **1.13** – **1.15**.

One reason for the lack of successful synthetic adaptation of TH polycyclization is that the savagely acidic and electrophilic carbocation intermediates in these processes (**1.10**, Figure 1.3) are prone to rapid E1 elimination or S_N1 reactions in bulk solvent, often precluding productive polycyclization in a synthetic setting. This stands in stark contrast to the well-studied HT processes where the bond-forming events occur through low-energy polydecalin transition states (**1.2**, Figure 1.1), allowing for rapid polycyclization that often outcompetes deleterious E1 or S_N1 pathways.

Many early efforts to synthesize sesquiterpenes through biomimetic TH polycyclization have been reported, often resulting in low-yielding complex mixtures largely comprised of

monocyclic bisabolenes.[16–19] An exception is a report from Hirose and coworkers, where treatment of farnesol (**1.18**, Figure 1.5) with BF_3-etherate produced a complex mixture containing numerous interesting polycyclic products with undetermined yields (**1.20**).[17] Shenvi and coworkers replicated this result and demonstrated that this acid-mediated process yielded 12% combined yield of α -cedrene and epi- α -cedrene (**1.19**) along with a complex mixture of volatile organic

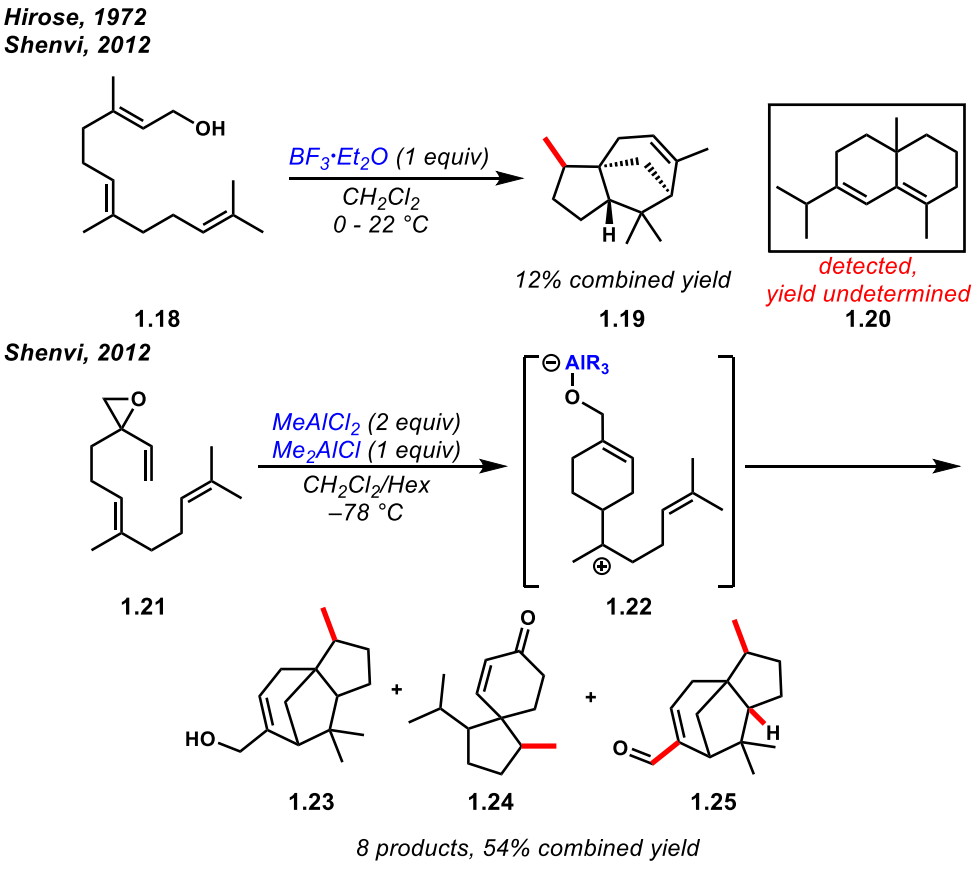


Figure 1.5 Lewis acid-mediated polycyclization of sesquiterpenoids. Bonds in red indicate sites at which multiple diastereomers were produced.

compounds and non-volatile polymers.[13] Shenvi and coworkers further demonstrated that treatment of farnesene oxide species **1.21** with stoichiometric Lewis acid yields a mixture of sesquiterpenoids (**1.23 – 1.25**) through a putative zwitterionic intermediate (**1.22**).[13] This mixture

includes both bi- and tricyclic species (**1.24** vs. **1.23** and **1.25**), a mixture of two oxidation states (**1.23** vs. **1.24** and **1.25**), and a mixture of diastereomers. Recently, the first example of non-enzymatic, catalytic TH cyclization was reported by Tiefenbacher and coworkers, who employed a supramolecular cluster to engage farnesyl acetate (**1.26**, Figure 1.6) affording an array of polycyclic sesquiterpene products (**1.19**, **1.20**, **1.28**, **1.29**). The product selectivity observed in this seminal example of catalytic TH cyclization, in particular the formation of δ-selinene (**1.20**), was attributed to encapsulation by the supramolecular assembly.[14–16]

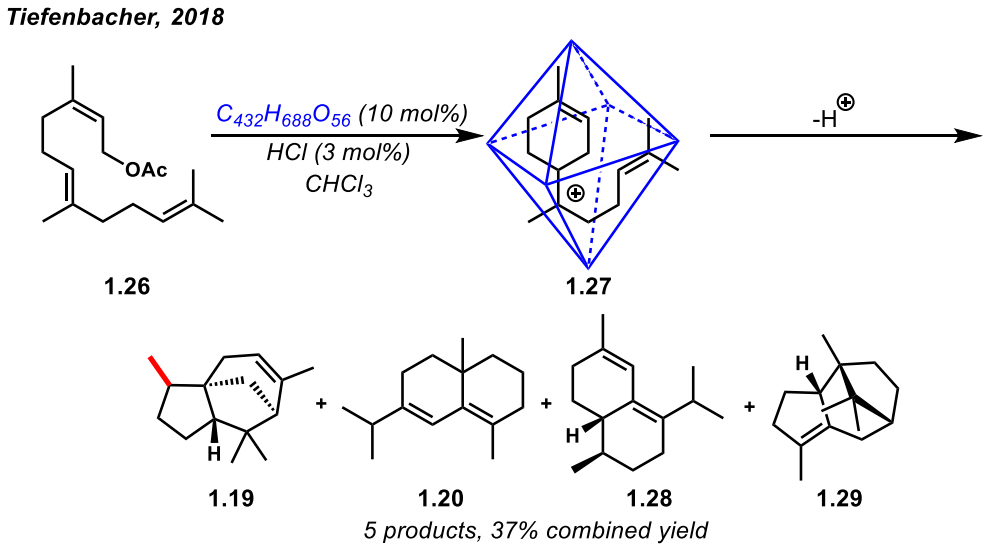


Figure 1.6 Supramolecular catalyst-promoted polycyclization of sesquiterpenoids. Bonds in red indicate sites at which multiple diastereomers were produced.

While the ability for a single biochemical feedstock to generate a vast number of complex secondary metabolites is advantageous in biology, in most synthetic adaptations, the generation of unselective mixtures of structurally complex terpene hydrocarbons presents a significant analytical challenge. Frequently, these reports identify reaction products from crude mixtures through comparison to natural product standards, due to purification challenges.[15,16–19] This can be an effective strategy for identification of known, commercially available terpene products, but the unambiguous identification of novel or uncommon terpene natural products requires isolation

of purified material. Development of methods capable of unambiguous structural analysis from mixtures of complex, unknown natural products would alleviate many of the analytical challenges associated with the synthetic adaptation of complex biomimetic processes.

1.4 Microcrystal Electron Diffraction

One possible solution to overcoming the analytical challenges presented in TH cyclization characterization is microcrystal electron diffraction, an emerging electron crystallography method. Recent reports have demonstrated the ability for this crystallographic technique to generate structural solutions from mixtures of pharmaceuticals and natural products. While chemists have been routinely utilizing X-ray crystallography for decades, the so called "nanocrystallography revolution" of electron diffraction began roughly one decade ago.[20] This is a bit surprising, given that the first electron spectrometer was built in the 1930's, only 20 years after the first X-ray spectrometer.[21] The disparity between routine use of these methods for structural solutions is partially attributable to differences in how X-rays and electrons interact with matter. Electrons interact very strongly with material relative to X-rays, and a single electron can be scattered multiple times as it passes through a crystal.[22] These dynamical scattering events were long believed to be too frequent and challenging for practical use of electron diffraction in three-dimensional structural elucidation.[23]

While multiple scattering events are rare in X-ray crystallography, the weak scattering of X-ray radiation by matter means that large (>50 micrometers per dimension, Figure 1.7) crystals are required to produce suitable data for crystallographic analysis from in-house X-ray sources.[24] Generation of these large crystals suitable for single crystal X-ray analysis is considered by many to be an "art" and is often the bottleneck when it comes to structural analysis by X-ray crystallography.

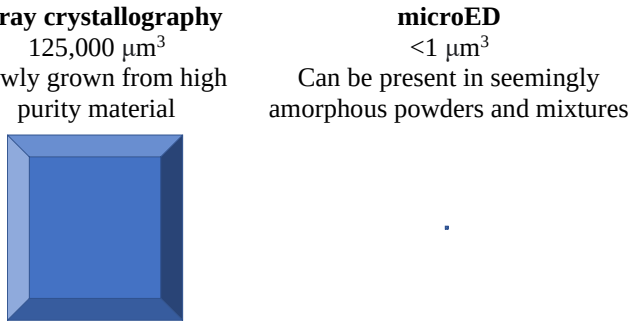


Figure 1.7 Difference in crystal size and purity for standard X-ray crystallography and microcrystal electron diffraction.

Although complex scattering events present a challenge, the strong interaction of electrons with matter has an incredibly useful outcome: crystals under a micron in size can produce diffraction patterns. These microcrystals, one-billionth the size of those needed for X-ray crystallography, can be present in mixtures, may be present in samples isolated directly after purification by rotary evaporation, or from crystallization screens that failed to produce X-ray quality species.[25,26] Additionally, as little as 200 nanograms of material has been reported to produce crystals suitable for an electron diffraction solution.[27] This not only allows chemists to obtain crystal structures of compounds that may not be able to be analyzed unambiguously by any other method, but also presents an exciting opportunity in which development of this technique could lead to crystallography becoming a standard step in a synthetic chemists' structural analysis workflow.

1.5 Early three-dimensional structural reconstruction

Early three-dimensional structure elucidation by electron diffraction required manual data collection by expert microscopists, who painstakingly aligned crystals to within 0.1^0 of a crystalline axis of interest. Manual alignment and collection of these zone axis allowed for

structural resolution of a few materials, including small molecule organic species, but this incredibly time-consuming process was impractical for routine structural analyses.[28,29] It wasn't until 2007, after the development of automated tomography processes from Kolb, Zou, and others, along with hardware and software improvements in modern TEMs, that electron crystallography became an increasingly promising technique for obtaining structural solutions.[30–32] Since these advancements, numerous methods for collecting and processing electron diffraction data have been reported in the literature. Broadly, most techniques can be categorized as convergent beam electron diffraction (CBED) or selected area electron diffraction (SAED or SAD); within those categories, collection techniques such as continuous rotation electron diffraction (cRED), precession electron diffraction tomography (PEDT), and microcrystal electron diffraction (microED) have been reported for three-dimensional structural analysis.[33] There are numerous books highlighting the differences in collection strategies and processing of the resultant data; however, this thesis will focus on use of microED, a SAED technique.[34,35]

The Gonen laboratory was the first to develop these electron crystallographic strategies for solution of protein structures, coining the name "microED" in 2013.[36] Data preparation involved deposition and vitrification of solvated protein microcrystals on a TEM grid. Once a suitable microcrystal was identified using TEM imaging and isolated from the background using a selected area aperture, a diffraction dataset could be collected by continuously rotating the crystal under the electron beam (Figure 1.8). Initial work studied microcrystals of a known protein species and

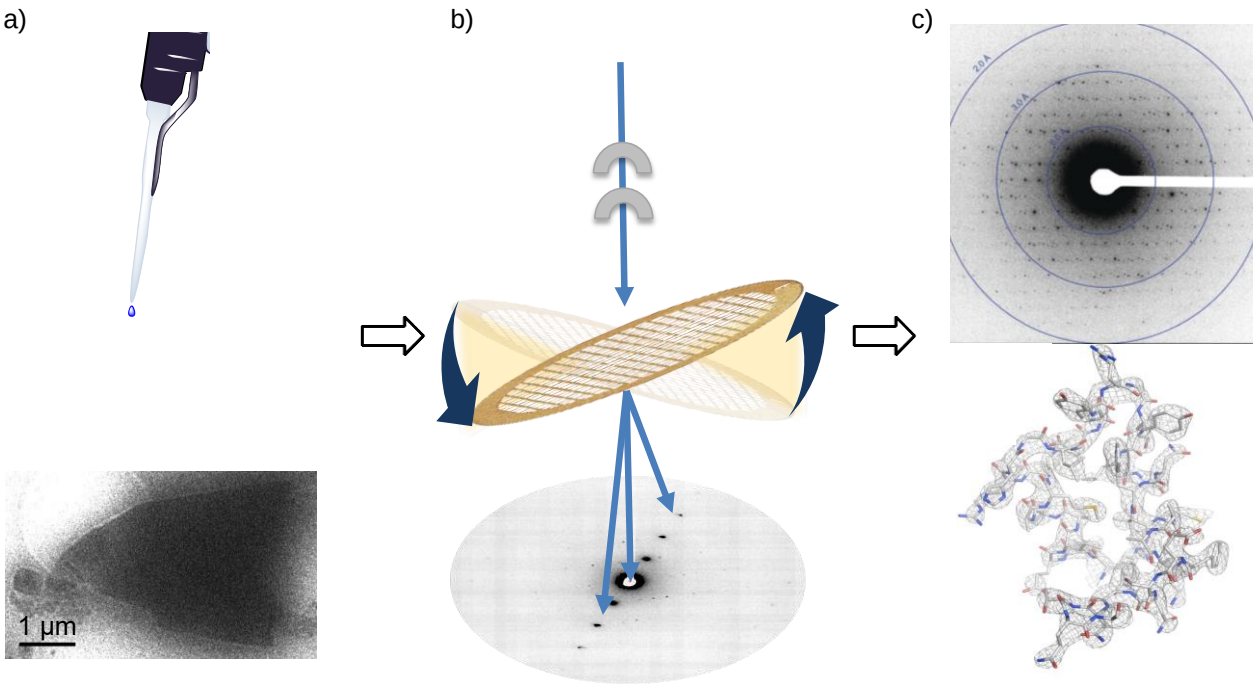


Figure 1.8 MicroED workflow: **a)** suspension of crystals is deposited and vitrified onto TEM grid, **b)** diffraction data of rotating crystal is obtained, **c)** molecular replacement is used to phase low resolution data to obtain protein structure.

produced low resolution (2.9 Å) data that could be solved using standard molecular replacement techniques, in which a similar structural model is used to phase the diffraction data and obtain a structural solution. Later work by the Gonen laboratory demonstrated the ability of microED to obtain high resolution crystal structures of small proteins utilizing *ab initio* methods.[37] Importantly, these *ab initio* solutions could be obtained using common small molecule X-ray crystallographic software, XDS and the SHELX suite, making it feasible for crystallographers to easily process electron diffraction datasets.[38–44]

In 2018, our laboratory in collaboration with the Gonen lab, as well as Gruene *et al.*, published back-to-back papers demonstrating the broad use of microED on small molecule compounds.[25,33] In both reports, commercial small molecule substances could be analyzed as-is, without any special crystallization attempts and minimal sample preparation. In contrast to

microED of proteins, these small molecule microcrystals did not require solvation or vitrification, and could simply be loaded onto a TEM grid as a dry powder before placing into liquid nitrogen for cryogenic analysis. Additionally, a mixture of microcrystalline powders could be sampled and structures of each individual component could be obtained. The refined structures from these reports were overlayed with previously obtained X-ray structures, demonstrating good agreement between these two methods, and hydrogen atoms could be resolved from density maps in many cases.

Since this initial report, our laboratory and others have solved structures of novel natural products, organometallic complexes, MOFs, pharmaceuticals, and more, establishing the broad applicability of microED to a wide range of small molecule species.[26,45–51] Unique examples include compounds that failed to produce X-ray quality crystals despite years of study, revision of mischaracterized natural products, and elucidation of minor impurities from a mixture of compounds.[26,27,51] The number of small molecule structures solved by electron diffraction is increasing (Figure 1.9), but the majority of these structures are still coming from a handful of

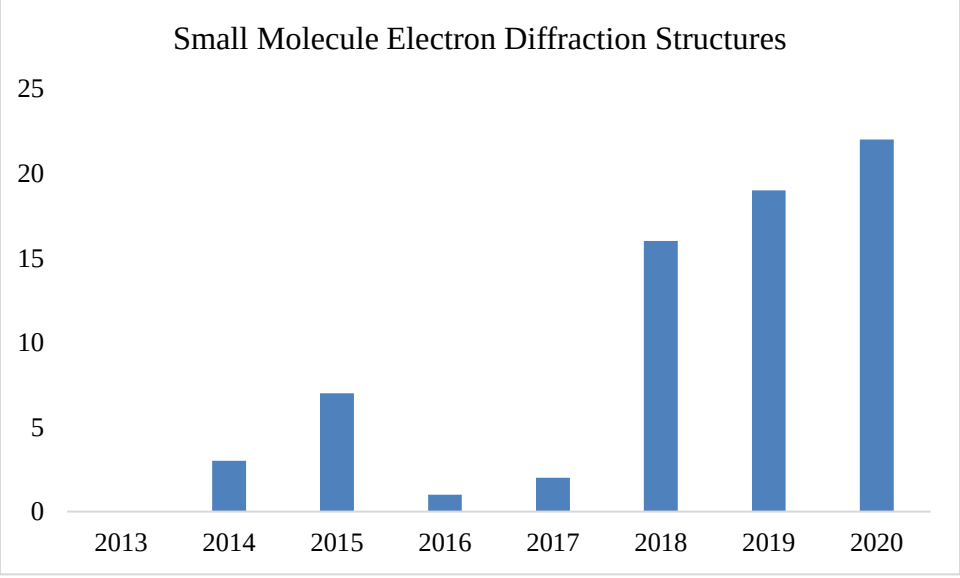


Figure 1.9 Number of electron diffraction structures deposited to CSD per year.

specialists in the field.[52] The lack of microED structures relevant to small molecule and drug discovery chemists until very recently piqued our interest, and we sought to solve complex chemical problems through application of this technique, as well as make the technique more accessible through development of automation regimes.

1.6 Efforts towards automation of microED

Automation of transmission electron microscope data collection and processing has been of significant research interest to the electron microscopy community for some time. In 2017, the Nobel Prize in Chemistry was awarded to Jacques Dubochet, Joachim Frank, and Richard Henderson for their work developing the cryo-electron microscopy (cryoEM) imaging technique.[53] The development of automation for cryoEM and cryo-electron tomography (cryoET) has been critical to the popularization of these techniques in biology, in which hundreds of thousands of images of biomolecules are merged together to produce high resolution images capable of resolving their structure.[54,55]

Automation of crystallographic workflows, on the other hand, have traditionally been focused on high-throughput sample preparation.[56] This is because the bottleneck of single crystal X-ray crystallography lies in the ability to grow large single crystals, typically not in data collection and processing. MicroED has a crystallographic data challenge more akin to that of cryoEM, where thousands of microcrystals can produce diffraction datasets from a single grid (Figure 1.10), and these patterns can be combined to generate a structural solution. This can be a repetitive, time-consuming process, amenable to automation.

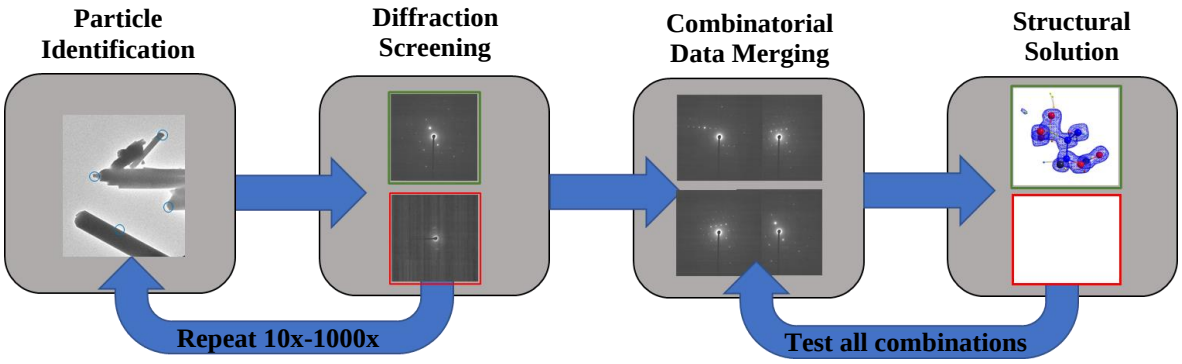


Figure 1.10 Crystallographic data challenges unique to microED.

An initial approach by the Gonen lab involved use of the open source TEM software SerialEM, which was developed for automation of electron tomography.[57,58] Users can input manually identified particle grid coordinates and utilize the electron tomography software to collect rotation data in diffraction mode.

Xiaodong Zou's group has developed data collection programs called Instamatic and InsteaDMatic.[59,60] These programs perform image recognition to find isolated particles and perform fully automated serialEM or microED data acquisition routines. Additionally, the programs can process the resulting datasets and perform hierarchical clustering to find datasets that may be suitable for combining to provide a solution.[61] The limitations of these programs are discussed in detail in Chapter 4, which explores alternative automation strategies to rapidly collect and process microED datasets.

1.5 Conclusion

Similar to NMR, once a niche technique only utilized by expert operators, electron crystallography has largely remained limited to specialists until recently. Just as NMR spectroscopy was brought to the forefront of small molecule research through improvements in theory, hardware, and automated software, we believe that similar improvements can result in routine use of microED for structural analysis of small molecules. These analytical advancements

increase the rate of discovery in synthetic chemistry, including those outlined in the biomimetic polycyclization reactions of terpene natural products.

1.6 Notes and References

(1) Ruzicka, L.; Eschenmoser, A; Heusser, H. The isoprene rule and the biogenesis of terpenic compounds. *Experientia* **1953**, *9*, 357–367.

(2) Stork, G.; Burgstahler, A. W. The stereochemistry of polyene cyclization. *J. Am. Chem. Soc.* **1955**, *77*, 5068–5077.

(3) Eschenmoser, A.; Ruzicka, L.; Jeger, O; Arigoni, D. A stereochemical interpretation of the biogenetic isoprene rule for the triterpenes. *Helv. Chim. Acta* **1955**, *38*, 1890–1904.

(4) Maimone, T. J.; Baran, P. S. Modern synthetic efforts toward biologically active terpenes. *Nat. Chem. Biol.* **2007**, *3*, 396–407.

(5) Alleman, R. K. Chemical wizardry? The generation of diversity in terpenoids biosynthesis. *Pure Appl. Chem.* **2008**, *80*, 1791–1798.

(6) Tantillo, D. J. Biosynthesis via carbocations: theoretical studies on terpene biosynthesis. *Nat. Prod. Rep.* **2011**, *28*, 1035–1053.

(8) (a) Johnson, W. S.; Semmelhack, M. F.; Sultanbawa, M. U. S.; Dolak, L. A. A new approach to steroid total synthesis. A nonenzymatic biogenetic-like olefinic cyclization involving the stereospecific formation of five asymmetric centers. *J. Am. Chem. Soc.* **1968**, *90*, 2994–2996. (b) Johnson, W. S; Brinkmeyer, R. S.; Kapoor, V. M.; Yarnell, T. M. Asymmetric total synthesis of 11-α-hydroxyprogesterone *via* a biomimetic polyene cyclization. *J. Am. Chem. Soc.* **1977**, *99*, 8341 –8343.

(9) (a) Corey, E. J.; Shouzhong, L. A short enantioselective total synthesis of Dammarenediol II. *J. Am Chem. Soc.* **1996**, *118*, 8765–8766. (b) Surendra, K.; Corey, E. J. Highly enantioselective proton-initiated polycyclization of polyenes. *J. Am. Chem. Soc.* **2012**, *134*, 11992–11994.

(10) Corey, E. J.; Luo, G.; Lin, L. S. A simple enantioselective synthesis of the biologically active tetracyclic marine sesterterpene Scalarenedial. *J. Am. Chem. Soc.* **1997**, *119*, 9927–9928.

(11) Ishihara, K.; Nakamura, S.; Yamamoto, H. The first enantioselective biomimetic cyclization of polyprenoids. *J. Am. Chem. Soc.* **1999**, *121*, 4906–4907.

(12) Bogenstatter, M.; Limberg, A.; Overman, L. E.; Tomasi, A. L. Enantioselective total synthesis of the kinesin motor protein inhibitor Adociasulfate 1. *J. Am. Chem. Soc.* **1999**, *121*, 12206–12207.

(13) Pronin, S. V.; Shenvi, R. A. Synthesis of highly strained terpenes by non-stop tail-to-head polycyclization. *Nat. Chem*. **2012**, *4*, 915–920.

(14) Zhang, Q.; Tiefenbacher, K. Terpene cyclization catalyzed inside a self-assembled cavity. *Nat. Chem.* **2015**, *7*, 197–202.

(15) (a) Zhang, Q.; Rinkel, J.; Goldfuss, B.; Dickschat, J. S.; Tiefenbacher, K. Sesquiterpene cyclizations catalysed in-side the resorcinarene capsule and application in the short synthesis of isolongifolene and isolongifolenone. *Nat. Cat.* **2018**, *1*, 609–615.

(b) Zhang Q.; Tiefenbacher, K. Sesquiterpene cyclizations inside the hexameric resorcinarene capsule: total synthesis of □-selinene and mechanistic studies *Angew. Chem. Int. Ed.* **2019**, *131*, 12818–12825.

(16) Gutsche, C. D.; Maycock, J. R; Chang, C. T. Acid-catalyzed cyclization of farnesol and nerolidol. *Tetrahedron* **1968**, *24*, 859–876.

(17) Ohta, Y.; Hirose, Y. Electrophile induced cyclization of farnesol. *Chem. Lett.* **1972**, *1*, 263–266.

(18) Polovinka, M. P. *et al*. Cyclization and rearrangements of farnesol and nerolidol stereoisomers in superacids. *J. Org. Chem.* **1994**, *59*, 1509–1517.

(19) Susumu, K.; Mikio, T.; Teruaki, M. Biogenetic-like cyclization of farnesol and nerolidol to bisabolene by the use of 2-fuorobenzothiazolium salt. *Chem. Lett.* **1977**, *6*, 1169–1172.

(20) Gemmi, M.; Mugnaioli, E.; Gorelik, T. E.; Kolb, U.; Palatinus, L.; Boullay, P.; Hovmoller, S.; Abrahams, J. P. 3D electron diffraction: the nanocrystallography revolution *ACS Cent. Sci.* **2019**, *5*, 1315–1329.

(21) Knoll, M.; Ruska, E. The electron microscope. *Z. Phys.* **1932**, *78*, 318–339.

(22) Spence, J. On the accurate measurement of structure-factor amplitudes and phases by electron diffraction. *Acta Crystallogr. Sec. A* **1993**, *49*, 231–260.

(23) Cowley, J. Electron Diffraction Techniques, Vol. 1. *IUCr, Oxford University Press*, **1992**.

(24) Dunitz, J. D. X-ray Analysis and the Structure of Organic Molecules. *Verlag Helvetica Chimica Acta: Zürich*, **1995**.

(25) Jones, C. G.; Martynowycz, M. W.; Hattne, J.; Fulton, T. J.; Stoltz, B. M.; Rodriguez, J. A.; Nelson, H. M.; Gonen, T. The CryoEM Method MicroED as a Powerful Tool for Small Molecule Structure Determination. *ACS Cent. Sci.* **2018**, *4*, 1587–1592.

(26) Jones, C. G.; Asay, M.; Kim, L. J.; Kleinsasser, J. F.; Saha, A.; Fulton, T. J.; Berkley, K. R.; Cascio, D.; Malyutin, A. G.; Conley, M. P.; Stoltz, B. M.; Lavallo, V.; Rodríguez, J. A.; Nelson, H. M. Characterization of Reactive Organometallic Species via MicroED. *ACS Centr. Sci.* **2019**, *5*, 1507–1513.

(27) Kim, L. J.; Xue, M.; Li, X.; Xu, Z.; Paulson, E.; Mercado, B. Q.; Nelson, H. M.; Herzon, S. Structural Revision of the Lomaiviticins *J. Am. Chem. Soc.* **2021**, *143*, 6578–6585.

(28) Kolb, U.; Matveeva, G. N. Electron crystallography on polymorphic organics. *Z. Kristallogr.* **2003**, *218*, 259–268.

(29) Zou, X. D.; Mo, Z. M.; Hovmoller, S.; Li, X. Z.; Kuo, K. H. Three-dimensional reconstruction of *v*-AlCrFe *Acta Cryst. A* **2003**, *59*, 526–539.

(30) Kolb, U.; Gorelik, T.; Kübel, C.; Otten, M. T.; Hubert, D. Towards automated diffraction tomography: Part I—Data acquisition. *Ultramicroscopy* **2007**, *107*, 507–513.

(31) Zhang, D.; Oleynikov, P.; Hovmöller, S.; Zou, X. Collecting 3D electron diffraction data by the rotation method. *Z. Kristallogr*. **2010**, *225*, 94–102.

(32) Franken, L. E.; Grunewald, K.; Boekema, E. J.; Stuart, M. C. A. A technical introduction to transmission electron microscopy for soft-matter: imaging, possibilities, choices, and technical developments *Small* **2020**, *16*, 1906198.

(33) Gruene, T.; Wennmacher, J. T. C.; Zaubitzer, C.; Holstein, J. J.; Heidler, J.; Fecteau-Lefebvre, A.; De Carlo, S.; Muller, E.; Goldie, K. N.; Regeni, I.; Li, T.; Santiso-Quinones, G.; Steinfeld, G.; Handschin, S.; van Genderen, E.; van Bokhoven, J. A.; Clever, G. H.; Pantelic, R. Rapid structure determination of microcrystalline molecular compounds using electron diffraction. *Angew. Chem. Int. Ed.* **2018**, *57*, 16313–16317.

(34) Gruene, T.; Holstein, J. J.; Clever, G. H.; Keppler, B. Establishing electron diffraction in chemical crystallography *Nat. Rev. Chem.* **2021**. https://doi.org/10.1038/s41570-021-00302-4.

(35) (a) Zuo, J. M.; Spence, J. C. H. Advanced Transmission Electron Microscopy. *Springer*, **2016**.
(b) Carter, C. B. & Williams, D. B. Transmission Electron Microscopy *Springer*, **2016**.

(36) Shi, D.; Nannenga, B. L.; Iadanza, M. G.; Gonen, T. Three-dimensional electron crystallography of protein microcrystals. *eLife* **2013**, *2*, e01345.

(37) Sawaya, M. R.; Rodriguez, J.; Cascio, D.; Collazo, M. J.; Shi, D.; Reyes, F. E.; Hattne, J.; Gonen, T.; Eisenberg, D. S. *Ab initio* structure determination from prion nanocrystals at atomic resolution by MicroED *PNAS* **2016**, *113*, 11232–11236.

(38) Hattne, J.; Reyes, F. E.; Nannenga, B. L.; Shi, D.; de la Cruz, M. J.; Leslie, A. G. W.; Gonen, T. MicroED data collection and processing. *Acta Crystallogr.*, Sect. A: Found. Adv. **2015**, *71*, 353−360.

(39) Kabsch, W. Xds. *Acta Crystallogr.* **2010**, *D66*, 125–132.

(40) Kabsch, W. Integration, scaling, space-group assignment and post-refinement. *Acta Crystallogr.* **2010,** *D66*, 133–144.

(41) Sheldrick, G. M. A short history of SHELX. *Acta Cryst.* **2008**, *A64*, 112–122.

(42) Sheldrick, G. M. SHELXT – Integrated space-group and crystal-structure determination. *Acta Cryst.* **2015** *A71*, 3–8.

(43) Sheldrick, G. M. Crystal structure refinement with SHELXL. *Acta Cryst.* **2015**, *C71*, 3–8.

(44) Hübschle, C. B., Sheldrick, G. M. & Dittrich, B. ShelXle: A Qt graphical user interface for SHELXL. *J. Appl. Cryst.* **2011,** *44*, 1281–1284.

(45) Mugnaioli, E.; Lanza, A. E.; Bortolozzi, G.; Righi, L.; Merlini, M.; Cappello, V.; Marini, L.; Athanassiou, A.; Gemmi, M. Electron Diffraction on Flash-Frozen Cowlesite Reveals the Structure of the First Two-Dimensional Natural Zeolite. *ACS Cent. Sci.* **2020**, *6*, 1578–1586.

(46) Clabbers, M. T. B.; Hongyi, X. Microcrystal electron diffraction in macromolecular and pharmaceutical structure determination. *Drug Discovery Today: Technologies*. **2020**, *In Press*.

(47) Das, P. P.; Perez, A. G.; Galanis, A. S.; Nicolopoulos, S. Structural Characterization of Beam Sensitive Pharmaceutical Compounds Using 3D Electron Diffraction-Micro-ED at Low Dose with Pixelated Detectors. *Microscopy and Microanalysis*. **2020**, *26*, 1522–1522.

(48) Wang, Y. *et al* Elucidation of the elusive structure and formula of the active pharmaceutical ingredient bismuth subgallate by continuous rotation electron diffraction *Chem. Commun.* **2017**, *53*, 7018–7021.

(49) Samkian, A.; Kiel, G. R.; Jones, C. G.; Bergman, H.; Oktawiec, J.; Nelson, H. M.; Tilley, T. D. Elucidation of Diverse Solid-State Packing in a Family of Electron-Deficient Expanded Helicenes via Microcrystal Electron Diffraction (MicroED) *Angew. Chem. Int. Ed.* **2020**, *5*, 2493–2499.

(50) Curtis, B. J.; Kim, L. J.; Wrobel, C. J. J.; Eagen, J. M.; Smith, R. A.; Burch, J. E.; Le, H. H.; Artyukhin, A. B.; Nelson, H. M.; Schroeder, F. C. Identification of Uric Acid Gluconucleoside–Ascaroside Conjugates in Caenorhabditis elegans by Combining Synthesis and MicroED *Org. Lett.* **2020**, *22*, 6724–6728.

(51) Kim, L. J.; Ohashi, M.; Zhang, Z.; Tan, D.; Asay, M.; Cascio, D.; Rodriguez, J. A.; Tang, Y.; Nelson, H. M. Prospecting for natural products by genome mining and microcrystal electron diffraction. *Nat. Chem. Biol.* **2021**, *17*, 872−877.

(52) Bruhn, J. F. *et al* Small molecule microcrystal electron diffraction for the pharmaceutical industry – lessons learned from examining over fifty samples. *Front. Mol. Biosci.* **2021**, *8*, 354.

(53) Shen, P. S. The 2017 nobel prize in chemistry: cryo-EM comes of age. *Anal. Bioanal. Chem.* **2018**, *410*, 2053–2057.

(54) Schorb, M.; Haberbosch, I.; Hagen, W. J. H.; Schwab, Y.; Mastronarde, D. N. Software tools for automated transmission electron microscopy *Nat. Methods* **2019**, *16*, 471–477.

(55) Tan, Y. Z.; Cheng, A.; Potter, C. S.; Carragher, B. Automated data collection in single particle electron microscopy. *Microscopy* **2016**, *65*, 43–56.

(56) Pusey, M. L.; Liu, Z.-J.; Tempel, W.; Praissman, J.; Lin, D.; Wang, B.-C.; Gavira, J. A.; Ng, J. D. Life in the fast lane for protein crystallization and X-ray crystallography. *Prog. Biophys. Mol. Biol.* **2005**, *88*, 359–386.

(57) Mastronarde, D. N. Automated electron microscope tomography using robust prediction of specimen movements. *J. Struct. Biol.* **2005**, *152*, 36–51.

(58) de la Cruz, J. M.; Martynowycz, M.; Hattne, J.; Gonen, T. MicroED data collection with SerialEM *Ultramicroscopy* **2019**, *201*, 77–80.

(59) Smeets, S.; Zou, X.; Wan, W. Serial electron crystallography for structure determination and phase analysis of nanocrystalline materials. *J. Appl. Crystallogr.* **2018**, *51*, 1262–1273.

(60) Roslova, M.; Smeets, S.; Wing, B.; Thersleff, T.; Xu, H.; Zou, X. InsteaDMatic: towards cross-platform automated continuous rotation electron diffraction *J. Appl. Crystallogr.* **2020**, *53*, 1217–1224.

(61) Wang, B.; Zou, X.; Smeets, S. Automated serial rotation electron diffraction combined with cluster analysis: an efficient multi-crystal workflow for structure determination *IUCrJ*. **2019**, *6*, 854–867.

CHAPTER TWO

Terpene Tail-to-Head Polycyclization Mediated by Lithium–Weakly Coordinating Anion Catalysis

Jessica E. Burch, Alex L. Bagdasarian, Tanin Hooshmand, Hosea M. Nelson *ChemRxiv Preprint* **2020**, DOI: 10.26434/chemrxiv.12719780.v2.

2.1 Abstract

Biomimetic total synthesis has played a pivotal role in the development of synthetic organic chemistry. In particular, efforts aimed at mimicking the head-to-tail (HT) cation–π cyclization cascades invoked in terpene biosynthesis, such as those catalyzed by type-II cyclases, have led to a multitude of new synthetic methods, chemical concepts, and total syntheses over the past century. Conversely, synthetic methodology that mimics tail-to-head (TH) cation–π cyclization cascades, mediated by Mg^{2+} type-I terpene cyclases, remains elusive in organic synthesis, despite key roles in the biosynthesis of privileged therapeutic molecules such as taxol and artemesinin. Here we report that Li^{+}/weakly-coordinating anion (WCA) salts catalyze the TH polycyclization of linaloyl fluoride, leading to high-yielding mixtures of polycyclic terpene natural products including cedrenes, cadinadiene, epizonarene, and δ-selinene. This report represents a seminal example of a small molecule-catalyzed TH polycyclization. Moreover we apply this strategy to geranyllinaloyl fluoride, demonstrating the application of small molecule-catalyzed TH polycyclization leading to mixtures of polycyclic diterpenes including the tricyclic

core of the gersemiols (named here as α-gersemiene), a recently discovered class of marine diterpenoid natural products.

2.2 Introduction

In contrast to head-to-tail (HT) cationic cyclizations (Figure 2.1a),[1–6] many early efforts to synthesize sesquiterpenes through biomimetic tail-to-head (TH) polycyclization reactions (Figure 2.1b) often resulted in low-yielding complex mixtures largely comprised of monocyclized compounds.[7–10] Modern examples, such as the stoichiometric Lewis acid-mediated polycyclization of farnesyl derivative **2.1** (Figure 2.1c) reported by Shenvi and coworkers, yields a mixture of bi- and tricyclic sesquiterpenoids of differing oxidation states, through a putative zwitterionic intermediate (**2.2**).[11] The seminal example of non-enzymatic, catalytic TH cyclization was reported by Tiefenbacher and coworkers, who engaged in supramolecular catalysis to encapsulate farnesyl acetate (**2.3**, Figure 2.1d) and generate an array of polycyclic sesquiterpene products.[12,13] The product selectivity was attributed to encapsulation by the supramolecular assembly, particularly in the case of δ-selinene (**2.4**, Figure 2.1e), which is biochemically proposed to originate from the formation of a 10-membered ring that then performs intramolecular cyclization.[14]

Here we diverge in approach from these seminal reports and describe the utilization of weakly-coordinating anion (WCA) catalysis to achieve the biomimetic conversion of sesquiterpene fluoride **2.5** (Figure 2.1e) to high yielding mixtures of polycyclic sesquiterpene natural products, including δ-selinene similarly observed in the supramolecular catalysis reaction (**2.4**, **2.6**–**2.9**).[12,13] Notably, cadinadiene (**2.6**), the trans-decalin variant of amorphadiene and a reported biosynthetic precursor of artemisinin, is formed in this reaction. Importantly, cadinadiene (**2.6**) was previously advanced in a 7-step synthetic sequence to racemic artemisinin

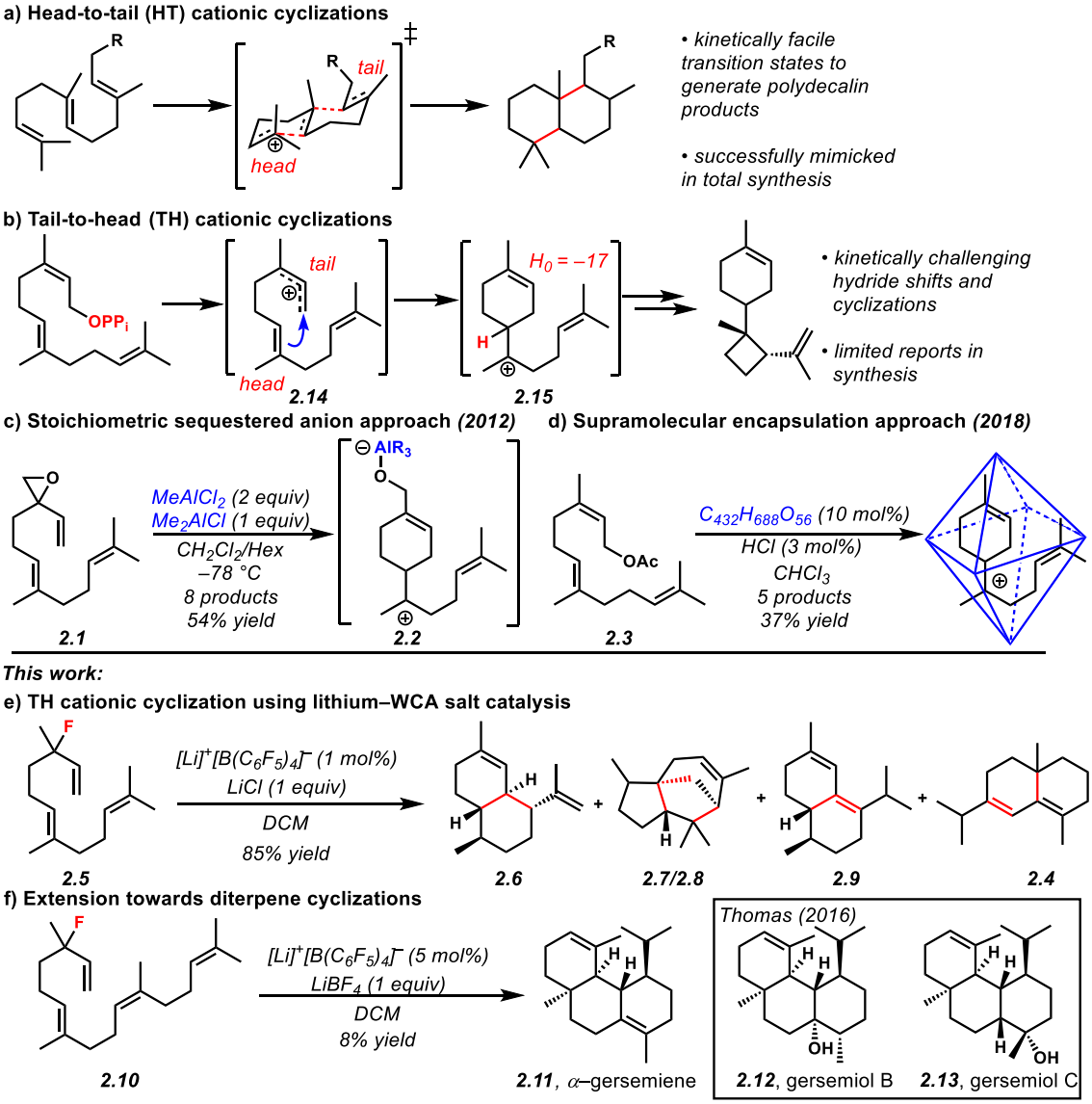


Figure 2.1. Summary of biomimetic terpene polycyclizations. **a**) Example of isoprenoid head-to-tail cyclization. **b**) Example of tail-to-head cationic cyclization with bisabolyl cationic intermediate. PP = pyrophosphate. **c**) Stoichiometric Lewis acid-promoted cationic cyclization of sesquiterpenoids. **d**) Supramolecular cluster-catalyzed cyclization of farnesyl acetate. **e/f**) This work: small molecule-catalyzed polycyclization of acyclic isoprenyl fluorides. *Yields determined by NMR using dimethyl sulfone as internal standard.*

by Yadav and coworkers.[15] Moreover, we report proof-of-principle that this strategy is amenable to diterpene polycyclization, as we demonstrate that geranyllinaloyl fluoride (**2.10**, Figure 2.1f)

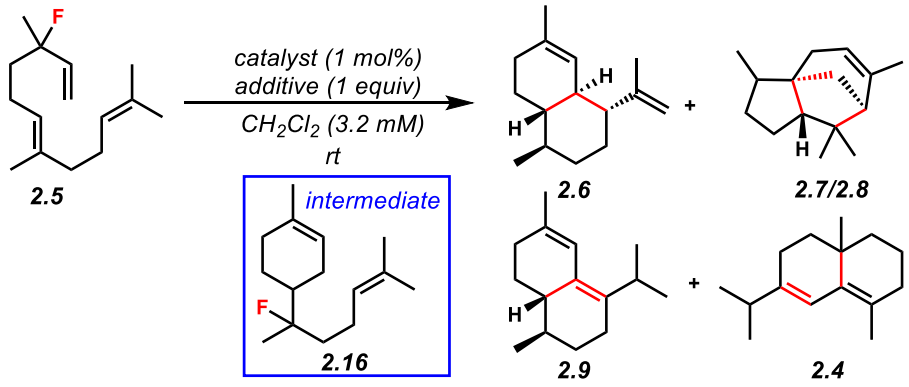
can be converted to the tricyclic diterpene, α-gersemiene (**2.11**) using this simple catalytic system. This species is the proposed bionsynthetic precursor to the recently isolated sesquiterpenoid products, gersemiol B (**2.12**) and gersemiol C (**2.13**).[16]

At the outset of our efforts, we hypothesized that the use of WCAs would allow for the generation of persistent carbocations with sufficient lifetimes to partake in polycyclization events before fast, counteranion-mediated E1 elimination or trapping by solvent.[17–19] Previous reports from our lab have demonstrated the competency of R_3Si^+/WCA-derived catalysts in generating long-lived carbocationic species that engage in intermolecular C–H insertion reactions.[20] Hence, we hypothesized that allylic fluoride **2.5** (Figure 2.1e) could be readily ionized under analogous R_3Si^+/WCA conditions to generate an allylic carbocation (**2.14**, Figure 1b) poised to undergo rapid *1,6*-cyclization to generate the bisabolyl cation (**2.15**). Despite having a Hammett acidity of ca. ≤ -17, we posited that this carbocation would have sufficient lifetime, when paired with a WCA, to undergo subsequent polycyclization without the formation of bisabolenes through E1 elimination as observed in previous literature reports. [7–10,21]

2.3 Sesquiterpenes

To validate our hypothesis, (*E*)-nerolidyl fluoride (**2.5**, Figure 2.2) was chosen as the model substrate due to the known fluorophilicity of R_3Si^+/WCA catalysts. Exposing fluoride **2.5** to triethylsilane and a catalytic amount of commercially available trityl tetrakis(pentafluorophenyl)borate, we observed low-yielding formation of α-cedrene (**2.7**), epi-α-cedrene (**2.8**), and cadinadiene (**2.6**) in addition to an intractable mixture of hydrocarbon products (Figure 2.2, entry 1). Gratifyingly, these polycyclic sesquiterpenes arise from multiple hydride shifts and cyclization events subsequent to the formation of the bisabolyl cation, supporting our mechanistic hypothesis.[22,23] Discouraged by the lack of selectivity and poor

efficiency, we posited that the incompatibility of R_3Si^+/WCA catalysts with dichloromethane and olefinic substrates was responsible for the poor reaction outcome.[24] These findings led us to explore the use of Li^+/WCA catalysts with (*E*)-nerolidyl fluoride (**2.5**) to attenuate this unproductive reaction outcome. Unfortunately, use of our reported Li^+/WCA catalytic conditions, featuring the *in situ* generation of $[Li]^+[B(C_6F_5)_4]^-$ through combination of LiHMDS and $[Ph_3C]^+[B(C_6F_5)_4]^-$, resulted in premature deprotonation of the intermediate carbocation, yielding a mixture of linear and monocyclic elimination products (entry 2).[25] However, we were gratified to find that pairing mild inorganic bases with metal/WCA salts yielded polycyclic products, albeit in low yields (entries 3–5). Utilization of stoichiometric $LiClO_4$ was found to



entry	additive	catalyst	yield (2.6, 2.7+2.8[a], 2.9, 2.4)[d]
1	Et_3SiH	$[Ph_3C]^+[B(C_6F_5)_4]^-$	12% (8, 4, 0, 0)
2	LiHMDS	$[Ph_3C]^+[B(C_6F_5)_4]^-$	0%
3	LiCl	$[Na]^+[BArF]^-$	trace
4	$MgCl_2$	$[Li]^+[B(C_6F_5)_4]^-$	53% (7, 17, 23, 6)
5	$ZnCl_2$	$[Li]^+[B(C_6F_5)_4]^-$	60% (5, 25, 18, 12)
6	$LiClO_4$	$[Li]^+[B(C_6F_5)_4]^-$	32% (0, 7, 0, 25)
7	LiCl	$[Li]^+[B(C_6F_5)_4]^-$	85% (7, 27, 30, 21)
8	–	$[Li]^+[B(C_6F_5)_4]^-$	17% (1, 6, 5, 5)
9	LiCl	–	0%
10[b]	LiCl	$[Li]^+[B(C_6F_5)_4]^-$	11% (1, 3, 6, 1)
11[c]	LiCl	$[Li]^+[B(C_6F_5)_4]^-$	0%
12[e]	LiCl	$[Li]^+[B(C_6F_5)_4]^-$	84% (9, 31, 44, 0)

Figure 2.2 Small molecule-catalyzed polycyclization of **2.5**. [a] Two diastereomers estimated from crude ^{1}H NMR (1:1 *d.r.*). [b] Reaction performed at –40 °C. [c] Reaction performed at 40 °C. [d]*Yields determined by NMR using dimethyl sulfone as internal standard.* [e] Utilizing bisabolyl fluoride **2.16** as starting material.

produce the cyclized terpene products in reduced yield, presumably due to promiscuous oxidative reactivity, as aromatic species such as cadalene were observed in the reaction mixture (entry 6, see Experimental Section). It was ultimately discovered that pre-formed $[Li]^+[B(C_6F_5)_4]^-$ (1 mol %) used in combination with stoichiometric LiCl (1 equivalent) provided a remarkable 85% combined yield of five known sesquiterpene natural products: cadinadiene (**2.6**), α-cedrene (**2.7**), epi-α-cedrene (**2.8**), epizonarene (**2.9**), and δ-selinene (**2.4**) (entry 7).[26] Control reactions were performed without stoichiometric LiCl resulting in reduced yields (entry 8). Use of LiCl in the absence of catalyst resulted in no reaction (entry 9). Performing the reaction at reduced temperature failed to ionize the substrate, while elevated temperatures resulted in formation of bisabolenes (entries 10–11). Interestingly, careful monitoring of the reaction by NMR and GC-FID revealed that bisabolyl fluoride (**2.16**) is an intermediate in this transformation. The origin of the fluorine atom in bisabolyl fluoride (**2.16**) is undetermined; however, reactive carbocations are known to undergo exchange reactions with typically inert molecules.[27]

The biosynthetic pathway for δ-selinene (**2.4**) is proposed to proceed *via* an initial *1,10*-cyclization of (E,E)-farnesyl pyrophosphate (**2.17 – 2.18**, Figure 2.3a) to generate germacrene A (**2.19**) which subsequently undergoes ring-closing to forge the selinene core (**2.20** and **2.14**).[14] Despite the detection of δ-selinene in Hirose's early work, the direct generation of a 10-membered carbocycle in our small molecule-catalyzed system seemed unlikely, given the potential for competitive formation of a 6-membered ring (i.e. **2.14** – **2.15**, Figure 2.1b, vs. **2.18** – **2.19**, Figure 2.3a).[8,11] Interestingly, upon subjection of bisabolyl fluoride (**2.16**) to the

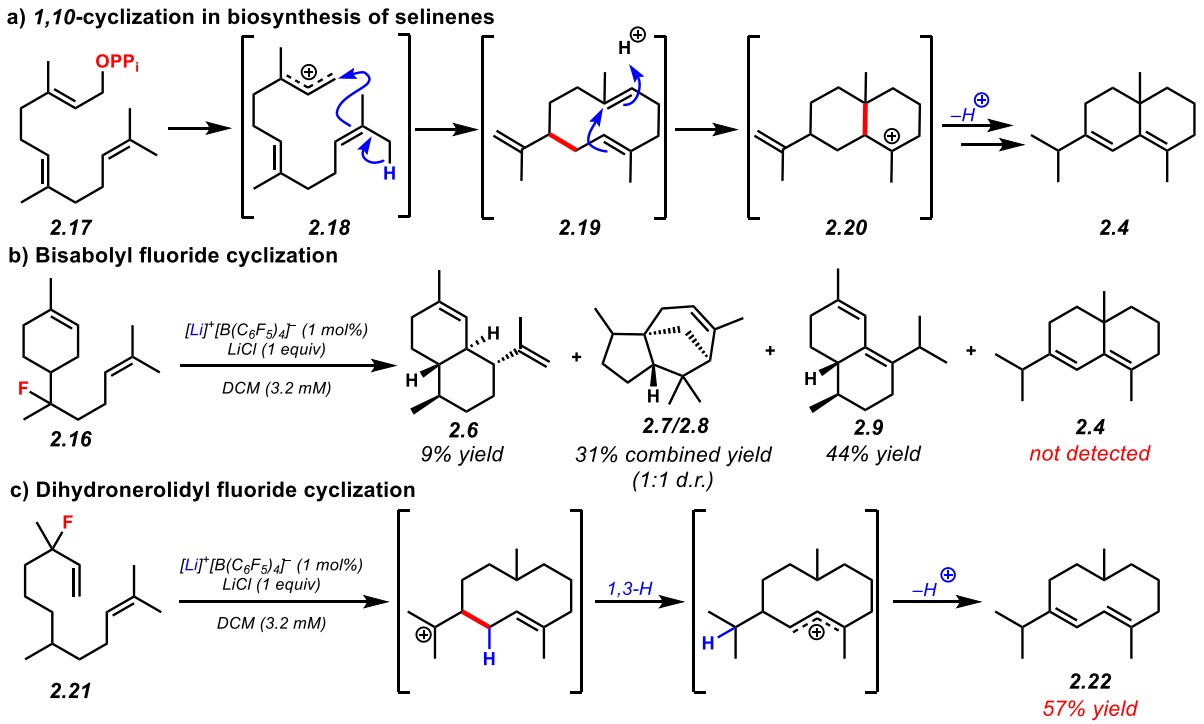


Figure 2.3 Investigation of δ-selinene formation. **a**) Biosynthetic cyclization to generate δ-selinene **2.4** through formation of a medium-sized ring **2.19**. PP = pyrophosphate. Anions excluded for clarity. **b**, Cyclization of prepared bisabolyl fluoride **2.16** to optimized Li^+/WCA reaction conditions. **c**,Generation of 10-membered carbocyclic product through WCA-catalyzed polycyclization. *Yields determined by NMR using dimethyl sulfone as internal standard.*

optimized reaction conditions, we found high conversion (84%) to cadinadiene (**2.6**), α-cedrenes (**2.7**/**2.8**), and epizonarene (**2.9**), however no δ-selinene (**2.4**) was detected (entry 12, Figure 2.2). The lack of formation of δ-selinene (**2.4**) from bisabolyl fluoride (**2.6**) supports the hypothesis that a biomimetic *1,10*-cyclization could be occurring in our system. To probe this hypothesis further, 6,7-dihydronerolidyl fluoride (**2.21**, Figure 2.3b), lacking the central olefin required to undergo a *1,6*-cyclization, was subjected to the optimized reaction conditions. In the event, the 10-membered carbocyclic product **2.22** was formed in 57% yield by NMR. Taken together, these experiments support biomimetic formation of δ-selinene (**2.4**) through a *1,10*-cyclization process without preorganization within an enzyme active site or supramolecular capsule.[14] While enzyme-mediated medium size ring formation from linear precursors is commonly invoked in biosynthesis, analogous catalytic transformations remain rare in synthetic chemistry.[28–30]

2.4 Diterpenes

Intrigued by this enzyme-free, medium-sized ring forming reaction, we wondered whether this catalytic strategy could be utilized in the biomimetic syntheses of more complex polycyclic diterpenes, which often proceed through medium- or macro-sized rings in nature. For example, taxadiene (**2.23**, Figure 2.4a), the oxidative precursor of taxol (**2.24**), originates from a tail-to-head cyclization of geranylgeranyl pyrophosphate (**2.25**) to forge 14-membered carbocation (**2.26**).[31] To the best of our knowledge, biomimetic approaches to taxadiene (**2.23**) have not been reported, presumably due in part to the high entropic cost of forging such rings outside the confines of an enzyme.[32] Interestingly, we found that treatment of geranyllinaloyl fluoride (**2.10**, Figure 2.4b) with stoichiometric $LiBF_4$ and catalytic Li^+/WCA led to formation of a complex mixture including α-gersemiene (**2.11**) in 8% yield by NMR. It is worthy to note that this putative natural product has also been observed in the acid-promoted cyclization of isocembrene (**2.27**), suggesting that the cembrenyl carbocation (**2.26**) is an intermediate in this process as well.[33] α-Gersemiene (**2.11**) is posited to be the biosynthetic precursor of the gersemiols, a class of recently isolated natural products from the soft coral species *Gersemia fruiticosa* (**2.12** and **2.13** Figure 2.1f).[16]

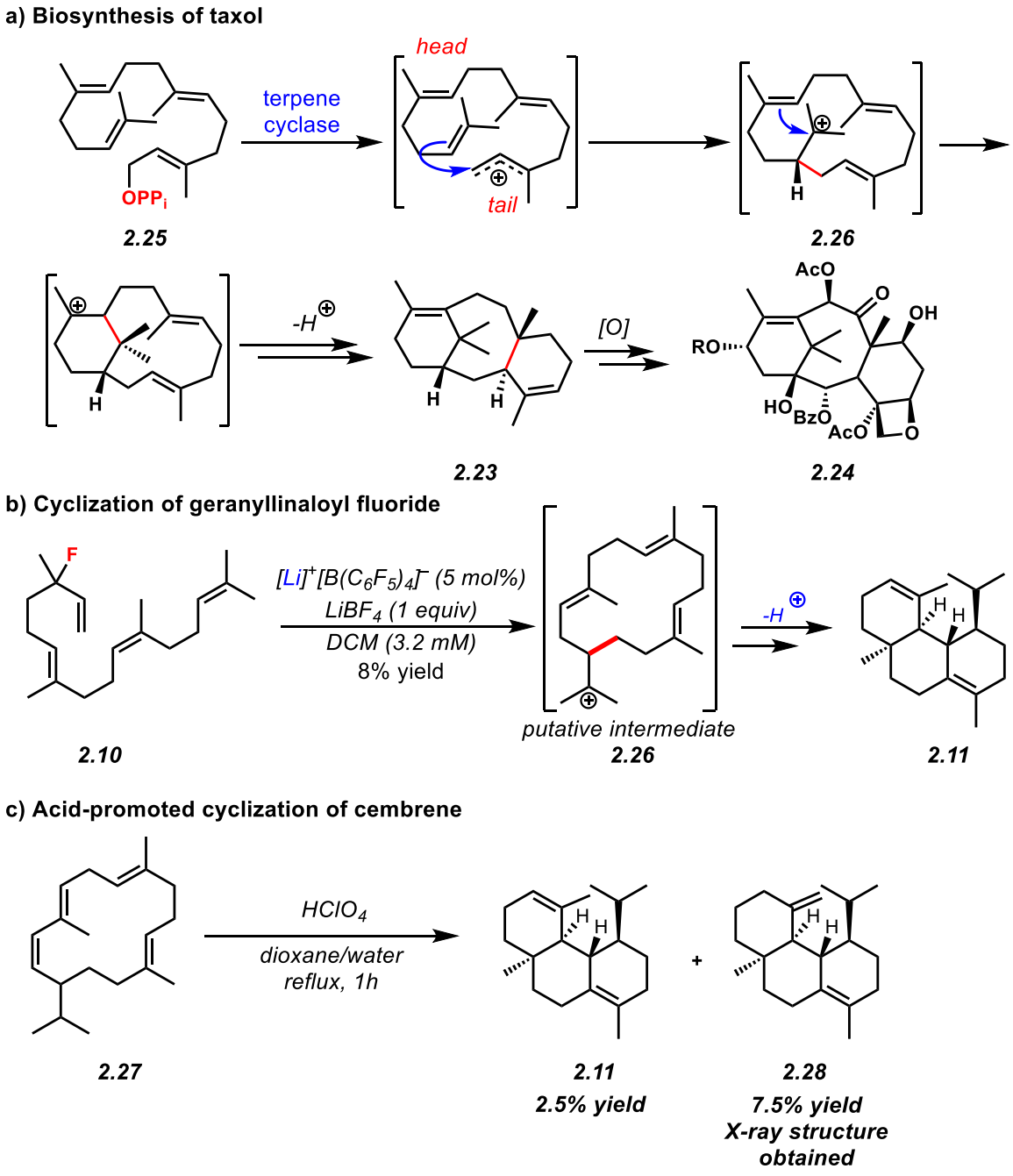


Figure 2.4 Diterpene polycyclizations. **a**) Biosynthetic cyclization to generate taxadiene **2.23** through formation of a large-sized ring **2.26**. PP_i = pyrophosphate. Anions excluded for clarity. **b**) Extension of methodology to generate tricyclic diterpene product **2.11** through putative intermediate **2.26**. *Yields determined by NMR using dimethyl sulfone as internal standard.* **c**) Acid-mediated cyclization of cembrene to generate tricyclic diterpenes **2.11** and **2.28**.

2.5 Microcrystal Electron Diffraction of Terpene Natural Products

Given the significant challenges associated with isolating and identifying terpene products, particularly in the diterpene study, we sought to explore analytical techniques to solve this problem. A thorough understanding of the diterpene polycyclization reaction outcome would allow for development of strategies to optimize this complex transformation. While many of the sesquiterpene products are oils under ambient conditions, cyclic di- and triterpene hydrocarbons are often reported as solids.[33] We hypothesized that crude or fractionated mixtures from diterpene polycyclization reactions could be amenable to study by the recently developed solid state structural analysis technique, microcrystal electron diffraction (microED).[34] At the time, our laboratory had reported the use of microcrystal electron diffraction (microED) for the elucidation of commercial and synthetic small molecule compounds, but its application to solving complex structural problems was limited.[35]

Initial attempts to evaluate diterpenes obtained from polycyclization were undertaken by performing slow evaporation from the crude reaction mixture or flash column chromatography fractions to generate microcrystals. Upon evaluation in the TEM, we were surprised to solve structures of minor impurities, including the generation of a preliminary structure proposed to be $LiBF_4$. Microcrystalline silicates, presumably from contamination after column chromatography, also presented a challenge.

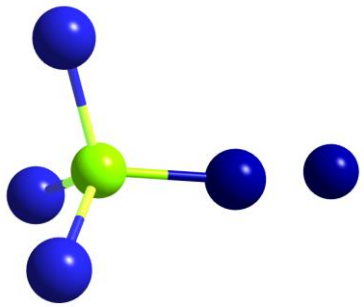


Figure 2.5 Preliminary microED structure of a tetrahedral salt impurity obtained from diterpene crude reaction mixture.

This is both a strength and weakness of microED: microcrystals not representative of the bulk sample can produce structural solutions.[36] Careful preparation by filtration of solvated mixtures in hydrocarbon solvent through 0.45 μm syringe filters reduced the presence of these impurities.

Most crystallization attempts from the mixture of terpene hydrocarbons produced oils, or oil-coated particles that diffracted to very low resolution or produced no diffraction. The X-ray structure of an isomer of α-gersemiene (**2.28**) has been reported from the acid-mediated cyclization of macrocyclic cembrene (**2.27**), so we sought to take HPLC fractionated α-gersemiene (**2.11**) and attempt to study it by electron diffraction.[33]

Slow evaporation from ethanol produced microcrystals that visibly decomposed under the electron beam but produced indexable movies with a unit cell similar to that of the X-ray structure of the gersemiene isomer (**2.28**, Figure 2.4 and Figure 2.5).[37] Evaluation of this sample, even under solvated cryogenic conditions, failed to produce a structure in our hands. Although these attempts proved unsuccessful, given the tremendous promise of this method and the lack of literature evaluating complex small molecule samples, we sought to apply and optimize this

method to make it a more practical tool for organic chemists to routinely utilize, as is discussed in Chapters 3 and 4.

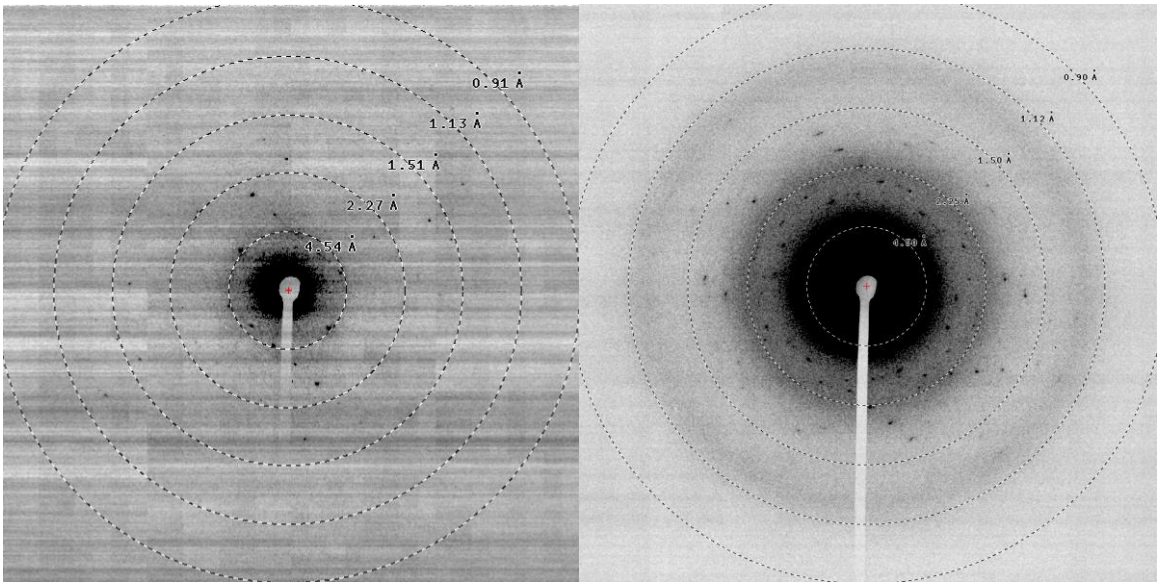


Figure 2.6 MicroED TEM diffraction images collected from particles obtained through crystallization of α-gersemiene-containing fraction.

2.6 Conclusion

In summary, we report Li^+/WCA-catalyzed tail-to-head, biomimetic cation-π cyclization reactions of sesquiterpenes and diterpenes. To the best of our knowledge, these are some of the first small molecule-catalyzed TH polycyclizations reported. We demonstrate that simple acyclic polyisoprenes can be converted to a mixture of naturally occurring polycyclic terpenes in high yield using a Li^+/WCA approach. We also report the first small molecule-catalyzed, biomimetic TH cyclization to forge a mixture of polycyclic diterpenes, including α-gersemiene (**2.11**). In the polycyclization of sesquiterpenes, we observe competitive formation of a 10-membered terpene intermediate despite the availability of a more facile pathway leading to a 6-membered ring. While terpene polycyclizations have been described as some of the most complex biosynthetic

reactions known, it is remarkable that we can achieve analogous transformations with simple lithium salts. We also explore use and limitations of the emerging electron microscopy technique, microED, for evaluation of diterpene reaction mixtures.

2.6 Notes and References

(1) Ruzicka, L.; Eschenmoser, A; Heusser, H. The isoprene rule and the biogenesis of terpenic compounds. *Experientia* **1953**, *9*, 357–367.

(2) Stork, G.; Burgstahler, A. W. The stereochemistry of polyene cyclization. *J. Am. Chem. Soc.* **1955**, *77*, 5068–5077.

(3) Eschenmoser, A.; Ruzicka, L.; Jeger, O; Arigoni, D. A stereochemical interpretation of the biogenetic isoprene rule for the triterpenes. *Helv. Chim. Acta* **1955**, *38*, 1890–1904.

(4) Maimone, T. J.; Baran, P. S. Modern synthetic efforts toward biologically active terpenes. *Nat. Chem. Biol.* **2007**, *3*, 396–407.

(5) Alleman, R. K. Chemical wizardry? The generation of diversity in terpenoids biosynthesis. *Pure Appl. Chem.* **2008**, *80*, 1791–1798.

(6) Tantillo, D. J. Biosynthesis via carbocations: theoretical studies on terpene biosynthesis. *Nat. Prod. Rep.* **2011**, *28*, 1035–1053.

(7) Gutsche, C. D.; Maycock, J. R; Chang, C. T. Acid-catalyzed cyclization of farnesol and nerolidol. *Tetrahedron* **1968**, *24*, 859–876.

(8) Ohta, Y.; Hirose, Y. Electrophile induced cyclization of farnesol. *Chem. Lett.* **1972**, *1*, 263–266.

(9) Polovinka, M. P. *et al*. Cyclization and rearrangements of farnesol and nerolidol stereoisomers in superacids. *J. Org. Chem.* **1994**, *59*, 1509–1517.

(10) Susumu, K.; Mikio, T.; Teruaki, M. Biogenetic-like cyclization of farnesol and nerolidol to bisabolene by the use of 2-fuorobenzothiazolium salt. *Chem. Lett.* **1977**, *6*, 1169–1172.

(11) 13. Pronin, S. V.; Shenvi, R. A. Synthesis of highly strained terpenes by non-stop tail-to-head polycyclization. *Nat. Chem*. **2012**, *4*, 915–920.

(12) Zhang, Q.; Tiefenbacher, K. Terpene cyclization catalyzed inside a self-assembled cavity. *Nat. Chem.* **2015**, *7*, 197–202.

(13) (a) Zhang, Q.; Rinkel, J.; Goldfuss, B.; Dickschat, J. S.; Tiefenbacher, K. Sesquiterpene cyclizations catalysed in-side the resorcinarene capsule and application in the short synthesis of isolongifolene and isolongifolenone. *Nat. Cat.* **2018**, *1*, 609–615.

(b) Zhang Q.; Tiefenbacher, K. Sesquiterpene cyclizations inside the hexameric resorcinarene capsule: total synthesis of δ-selinene and mechanistic studies *Angew. Chem. Int. Ed.* **2019**, *131*, 12818–12825.

(14) Wu, Q. X.; Shi, Y. P.; Jia, Z. J. Eudesmane sesquiterpenoids from the asteraceae family. *Nat. Prod. Rep.* **2006**, *23*, 699–734.

(15) Yadav, J. S.; Thirupathaiah, B.; Srihari, P. A concise stereoselective total synthesis of (+)-artemisinin. *Tetrahedron* **2010**, *66*, 2005–2009.

(16) Preckler-Angulo, C.; Genta-Jouve, G.; Mahajan, N.; de la Cruz, M.; de Pedro, N.; Reyes, F.; Iken, K.; Avila, C.; Thomas, O. P. Gersemiols A–C and eunicellol A, diterpenoids from the arctic soft coral *Gersemia fruticosa. J. Nat. Prod.* **2016**, *79*, 1132–1136.

(17) Olah, G. A.; Lukas, J. Stable carbonium ions. XXXIX. Formation of alkylcarbonium ions via hydride ion abstraction from alkanes in fluorosulfonic acid-antimony pentafluoride solution. Isolation of some crystalline alkylcarbonium ion salts. *J. Am. Chem. Soc.* **1967**, *89*, 2227–2228.

(18) Duttwyler, S.; Douvris, C.; Fackler, N. L. P.; Tham, F. S.; Reed, C. A.; Baldridge, K. K.; Siegel, J. S. C–F activation of fluorobenzene by silylium carboranes: Evidence for incipient phenyl cation reactivity. *Angew. Chem. Int. Ed.* **2010**, *210*, 7681–7684.

(19) Reed, C. A. H^+, CH_3^+, R_3Si^+ carborane reagents: when triflates fail. *Acc. Chem. Res.* **2010**, *43*, 121–128.

(20) Shao, B.‡; Bagdasarian, A. L.‡; Popov, S.; Nelson, H. M. Arylation of hydrocarbons enabled by organosilicon reagents and weakly coordinating anions. *Science* **2017**, *355*, 1403–1407.

(21) Reed, C. A. Carborane acids. New 'strong yet gentle' acids for organic and inorganic chemistry. *Chem. Commun.* **2005**, *13*, 1669–1677.

(22) Hong, Y. J.; Tantillo, D. J. Consequences of conformational preorganization in sesquiterpene biosynthesis. Theoretical studies on the formation of the bisabolene, curcumene, acoradiene, zizaene, cedrene, duprezianene, and sesquithuriferol sesquiterpenes. *J. Am. Chem. Soc.* **2009**, *131*, 7999–8015.

(23) Hong, Y. J.; Tantillo, D. J. Branching out from the bisabolyl cation. Unifying mechanistic pathways to barbatene, bazzanene, chamigrene, chamipinene, cumacrene, cuprenene, dunniene, isobazzanene, iso-γ-bisabolene, isochamigrene, laurene, microbiotene, sesquithujene, sesquisabinene, thujopsene, trichodiene, and widdradiene sesquiterpenes. *J. Am. Chem. Soc.* **2014**, *136*, 2450–2463.

(24) Kira, M.; Hino, T.; Sakurai, H. An NMR study of the formation of silyloxonium ions by using tetrakis[3,5-bis(trifluoromethyl)phenyl]borate as counteranion. *J. Am. Chem. Soc.* **1992**, *114*, 6697–6700.

(25) Wigman, B.; Popov, S.; Bagdasarian, A. L.; Shao, B.; Benton, T. R.; Williams, C. G.; Fisher, S. P.; Lavallo, V.; Houk, K. N.; Nelson, H. M. Strong bases and weak anions in catalytic C–H insertion reactions of vinyl carbocations. *J. Am. Chem. Soc.* **2019**, *141*, 9410–9144.

(26) Kuprat, M.; Lehmann, M.; Shulz, A.; Villinger, A. Synthesis of pentafluorophenyl silver by means of Lewis acid catalysis: Structure of silver solvent complexes. *Organometallics* **2010**, *29*, 1421–1427.

(27) Olah, G.A.; Molnár A. Hydrocarbon Chemistry, 2nd revised ed. **2003**, Wiley Interscience, New York.

(28) Shippey, M. A.; Dervan, P. B. Synthesis of cycloalkanes by intramolecular titanium-induced dicarbonyl coupling. *J. Org. Chem.* **1977**, *42*, 2655–2656.

(29) Winkler, J. D.; Sridar, V.; Siegel, M. G. Ten-membered ring templates for stereoselective radical cyclizations. *Tet. Lett.* **1989**, *30*, 4943–4946.

(30) Nevalainen, M.; Koskinen, A. M. P. Synthesis of a 10-membered carbocycle by olefin metathesis. *Angew. Chem. Int. Ed.* **2001**, *40*, 4060–4062.

(31) Hong, Y. J.; Tantillo, D. J. The taxadiene-forming carbocation cascade. *J. Am. Chem. Soc.* **2011**, *133*, 18249–18256.

(32) (a) Christianson, D. W. Structural biology and chemistry of the terpenoid cyclases. *Chem. Rev.* **2006**, *106*, 3412–3442.

(b) Christianson, D. W. Structural and chemical biology of terpenoid cyclases *Chem. Rev.* **2017**, *117*, 11570–11648.

(33) Dauben, W. G.; Hubbell, J. P.; Oberhansli, P.; Thiessen, W. E. Acid-catalyzed cyclization of cembrene and isocembrol. *J. Org. Chem.* **1979**, *44*, 669–673.

(34) Gemmi, M.; Mugnaioli, E.; Gorelik, T. E.; Kolb, U.; Palatinus, L.; Boullay, P.; Hovmöller, S.; Abrahams, J. P. 3D Electron Diffraction: The Nanocrystallography Revolution. *ACS Cent. Sci.* **2019**, *5*, 1315–1329.

(35) Jones, C. G.; Martynowycz, M. W.; Hattne, J.; Fulton, T. J.; Stoltz, B. M.; Rodriguez, J. A.; Nelson, H. M.; Gonen, T. The CryoEM Method MicroED as a Powerful Tool for Small Molecule Structure Determination. *ACS Cent. Sci.* **2018**, *4*, 1587–1592.

(36) Kim, L. J.; Ohashi, M.; Zhang, Z.; Tan, D.; Asay, M.; Cascio, D.; Rodriguez, J. A.; Tang, Y.; Nelson, H. M. Prospecting for natural products by genome mining and microcrystal electron diffraction. *Nat. Chem. Biol.* **2021**, *17*, 872−877.

(37) Hattne, J.; Shi, D.; Glynn, C.; Zee, C.-T.; Gallagher-Jones, M.; Martynowycz, M. W.; Rodriguez, J. A.; Gonen, T. Analysis of Global and Site-Specific Radiation Damage in Cryo-EM *Structure* **2018**, *5*, 759–766.

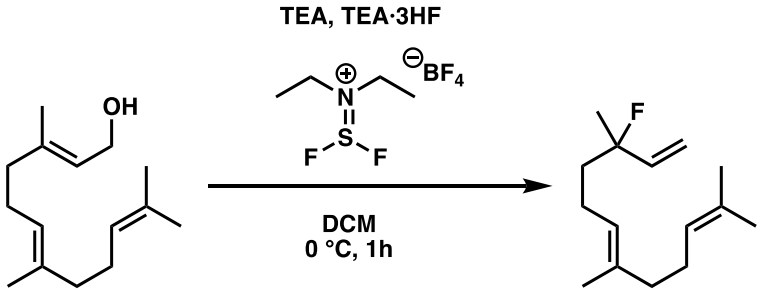
2.7 Experimental Section

2.7.1 Materials and Methods

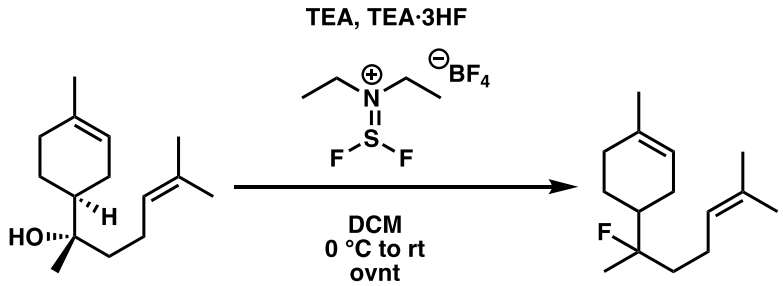
Unless otherwise stated, all reactions were performed in an MBraun glovebox under nitrogen atmosphere with ≤ 0.5 ppm O_2 levels. All glassware and stir-bars were dried in a 160 °C oven for at least 12 hours and dried in vacuo before use. All substrates were dried over P_2O_5. All solvents were rigorously dried before use. Dichloromethane was degassed and dried in a JC Meyer solvent system and stored inside a glovebox. Triethylsilane was dried over sodium and stored inside a glovebox. $[Li]^+[B(C_6F_5)_4]^-$ salt was synthesized according to literature procedure.[1] XtalFluor-E was prepared according to literature procedure.[2] Preparatory thin layer chromatography (TLC) was performed using Millipore silica gel 60 F_{254} pre-coated plates (0.25 mm) and visualized by UV fluorescence quenching. SiliaFlash P60 silica gel (230–400 mesh) was used for flash chromatography. $AgNO_3$-impregnated silica gel was prepared by mixing with a solution of $AgNO_3$ (150% v/w of 10% w/v solution in acetonitrile), removing solvent under reduced pressure, and drying at 120 °C. NMR spectra were recorded on a Bruker AV-300 (^1H, ^{19}F), Bruker AV-400 (^1H, ^{13}C, ^{19}F), Bruker DRX-500 (^1H), and Bruker AV-500 (^1H, ^{13}C). ^1H NMR spectra are reported relative to $CDCl_3$ (7.26 ppm) unless noted otherwise. Data for ^1H NMR spectra are as follows: chemical shift (ppm), multiplicity, coupling constant (Hz), integration. Multiplicities are as follows: s = singlet, bs = broad singlet d = doublet, t = triplet, dd = doublet of doublet, dt = doublet of triplet, ddd = doublet of doublet of doublet, td = triplet of doublet, m = multiplet. ^{13}C NMR spectra are reported relative to $CDCl_3$ (77.0 ppm) unless noted otherwise. GC spectra were recorded on an Agilent 6850 series GC using an Agilent HP-1 (50 m, 0.32 mm ID, 0.25 mm DF) column. GCMS spectra were recorded on a Shimadzu GCMS-QP2010 using a Restek XTI-5 (50m, 0.25 mm ID, 0.25 mm DF) column interface at room

temperature. IR spectra were record on a Perkin Elmer 100 spectrometer and are reported in terms of frequency absorption (cm^{-1}). High resolution mass spectra (HR-MS) were recorded on an Agilent GC EI-MS, and are reported as m/z (% relative intensity). Purification by preparative HPLC was done on an Agilent 1200 series instrument with a reverse phase Alltima C18 (5m, 25 cm length, 1 cm internal diameter) column.

2.7.2 Preparation of Allylic and Tertiary Fluoride Substrates

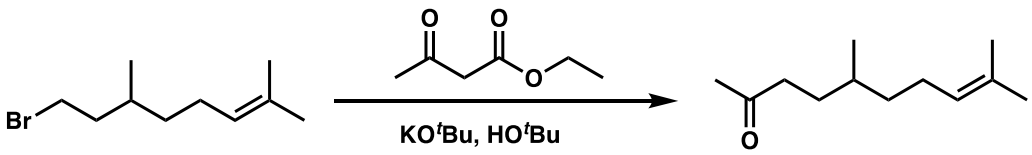


(*E*)-3-fluoro-3,7,11-trimethyldodeca-1,6,10-triene (2.5). Synthesized using (2E,6E)-farnesol according to a modified literature method.[2] A flame-dried round bottom flask under nitrogen atmosphere was charged with triethylamine (1.2 mL, 9.0 mmol, 1.0 equiv), triethylamine trihydrofluoride (2.9 mL, 18 mmol, 2.0 equiv), and dichloromethane (20 mL). After cooling to 0 °C, XtalFluor-E (3.1 g, 13 mmol, 1.5 equiv) and (2E,6E)-farnesol were added successively. The reaction was stirred at 0 °C for 30 minutes and allowed to warm to room temperature. Upon warming, reaction was quenched with saturated aqueous bicarbonate solution until a neutral pH was reached. Aqueous layer was extracted with DCM 3x, combined, and washed with sat. sodium bicarbonate solution, brine, dried over Na_2SO_4, filtered, and concentrated under reduced pressure. The resulting crude oil was purified *via* flash chromatography (1% Et_2O/hexanes) to afford **2.5** as a colorless oil. Compound was stored in plastic vial to avoid decomposition. Spectral data match those reported in the literature.[3]

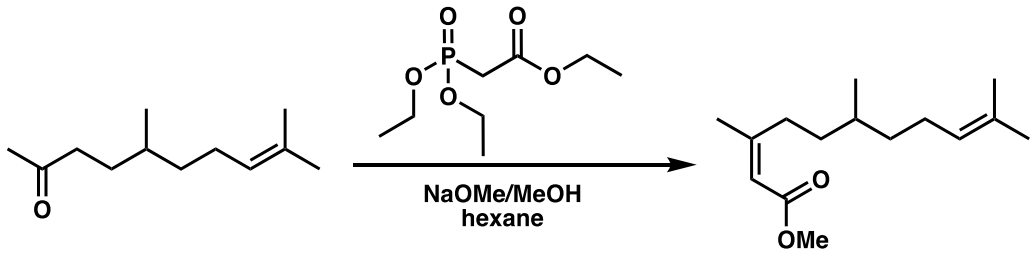


4-(2-fluoro-6-methylhept-5-en-2-yl)-1-methylcyclohex-1-ene (2.16). Synthesized using α-(–)-bisabolol according to a modified literature method.[2] A flame-dried round bottom flask under nitrogen atmosphere was charged with triethylamine (1.2 mL, 9.0 mmol, 1.0 equiv), triethylamine trihydrofluoride (2.9 mL, 18 mmol, 2.0 equiv), and dichloromethane (20 mL). After cooling to 0 °C, XtalFluor-E (3.1 g, 13 mmol, 1.5 equiv) and α-(–)-bisabolol were added successively. The reaction was stirred at 0 °C for 30 minutes and allowed to warm to room temperature. Reaction was allowed to stir overnight. Upon completion, reaction was quenched with saturated aqueous bicarbonate solution until a neutral pH was reached. Aqueous was extracted with DCM 3x, combined, and washed with sat. sodium bicarbonate solution, brine, dried over Na_2SO_4, filtered, and concentrated under reduced pressure. The resulting crude oil was purified *via* flash chromatography (1% Et_2O/hexanes) to afford **2.16** as a colorless oil. Spectral data match those reported in the literature.[42]

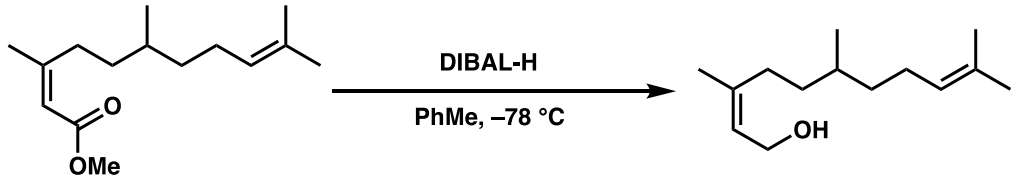
8-bromo-2,6-dimethyloct-2-ene (2.29). Synthesized according to known procedures. Spectral data match those reported in the literature.[5]



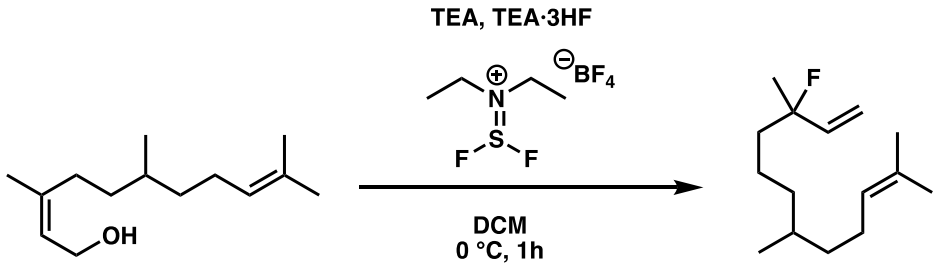
5,9-dimethyldec-8-en-2-one (2.30). Synthesized according to known procedures. Spectral data match those reported in the literature.[5]



methyl-3,6,10-trimethylundeca-2,9-dienoate (2.31). Synthesized according to known procedures. Obtained as mixture of diastereomers and utilized in next reaction without purification. Spectral data match those reported in the literature.[6]



3,6,10-trimethylundeca-2,9-dien-1-ol (2.32). Synthesized according to known procedures and isolated as a mixture of diastereomers. Spectral data match those reported in the literature.[6]



3-fluoro-3,7,11-trimethyldodeca-1,10-diene (2.21). Synthesized from **2.32** (mixture of isomers) according to a modified literature method outlined in **2.5** on a 1 mmol scale. The resulting crude

oil was purified *via* flash chromatography (1% Et_2O/hexanes) to afford 140 mg (58%) of **2.21** as a colorless oil, mixture of diastereomers.

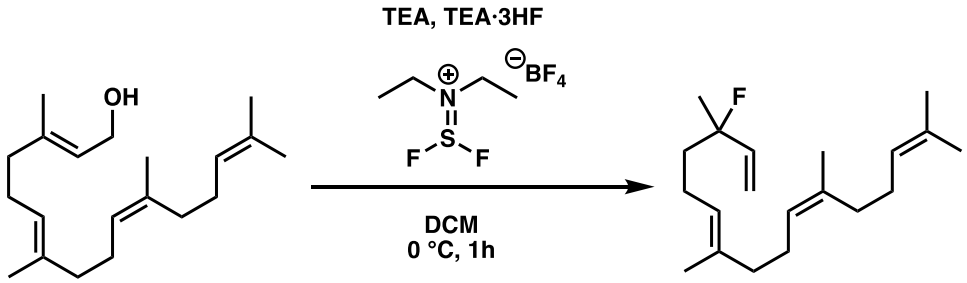
^1H NMR (500 MHz, $CDCl_3$) δ 5.88 (td, J = 17.8, 11.3 Hz, 1H), 5.24, (dt, 17.4, 1.3 Hz, 1H), 5.12 - 5.04 (m, 2 H), 1.95 (m, 2H), 1.68 (s, 3H), 1.60 (s, 3H), 1.40 (d, J = 21.8 Hz, 3H), 1.41 - 1.23 (m, 6H), 1.19 - 1.04 (m, 3H) 0.86 (d, J = 6.5 Hz, 3H).

^{13}C NMR (125 MHz, $CDCl_3$) δ 135.12, 135.05, 134.91, 134.59, 131.27, 124.65, 124.41, 124.34, 124.29, 124.27, 96.45, 95.11, 41.02, 40.84, 39.79, 39.76, 39.73, 28.28, 26.77, 26.67, 26.62, 26.57, 25.71, 22.24, 22.20, 17.69, 16.05, 16.04, 16.01, 15.79.

^{19}F NMR (282 MHz, $CDCl_3$) δ -147.31, -147.35.

FTIR (Neat Film NaCl): 2914, 2870, 1458, 1410, 1376, 1187, 989, 925, 902, 882, 747, 685 cm^{-1}.

HRMS (EI-MS): Calculated for $C_{15}H_{27}F$: 226.2097; Measured: 226.2099.



3-fluoro-3,7,11,15-tetramethylhexadeca-1,6,10,14-tetraene (2.10). Synthesized from geranyllinalool (mixture of isomers) according to a modified literature method outlined in **2.5** on a 5 mmol scale.[2] The resulting crude oil was purified via flash chromatography (1% Et-$_2$O/hexanes) to afford 640 mg (42%) of **2.10** as a colorless oil, mixture of diastereomers.

^1H NMR (500 MHz, $CDCl_3$) δ 5.89 (td, J = 17.7, 11.1 Hz, 1H), 5.30 – 5.21 (m, 1H), 5.14 – 5.06 (m, 4H), 2.10 – 1.96 (m, 10H), 1.68 (s, 3H), 1.59 (bs, 9H), 1.41 (dd, J = 21.7, 1.1 Hz, 3H), 1.30 – 1.22 (m, 2H).

^{13}C NMR (101 MHz, CDCl3) δ 140.91, 140.88, 140.65, 135.72, 135.52, 135.40, 135.27, 135.16, 131.54, 131.34, 124.99, 124.54, 124.45, 124.39, 124.37, 124.16, 124.05, 123.69, 113.35, 113.24, 113.21, 40.48, 40.21, 39.97, 39.76, 32.01, 31.87, 26.73, 26.65, 26.56, 26.48, 25.74, 25.71, 25.40, 25.15, 23.43, 23.40, 22.24, 22.19, 17.70, 17.65, 15.95.

^{19}F NMR (282 MHz, $CDCl_3$) δ -148.14, -148.20, -148.28, -148.32, -150.52, -150.58, -150.61, -150.69

FTIR (Neat Film NaCl): 3090, 2965, 2925, 2855, 1712, 1647, 1595, 1448, 1375, 1152, 1108, 989, 926, 897, 832, 741, 507 cm^{-1}.

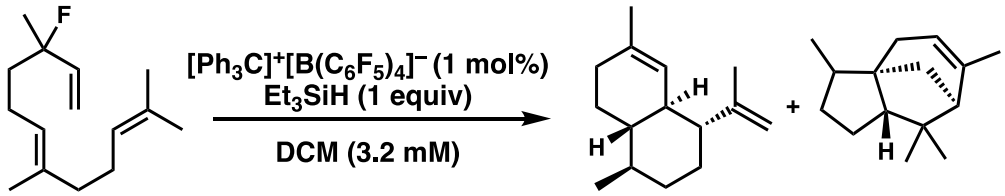
HRMS (EI-MS): Calculated for $C_{20}H_{33}F$: 292.2566; Measured: 292.2572.

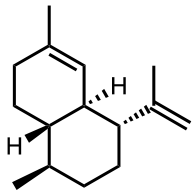
2.7.3 Sesquiterpene R_3Si^+/WCA Tail-to-Head Cyclization Reactions

2.7.3.1 Procedure for for R_3Si^+/WCA Tail-to-Head Cyclizations

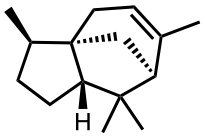
In a well-kept glovebox, H_2O, $O_2 \leq 0.5$ ppm, a scintillation vial was charged with dichloromethane (0.064 mM, 20 mL) and $[Ph_3C]^+[B(C_6F_5)_4]^-$ (0.01 equiv, 0.6 mg) by adding 60 uL of a freshly prepared 10.8 mM stock solution in dichloromethane. To this clear yellow solution, triethylsilane (2.0 equiv, 15 mg) was added at room temperature and stirred until solution turned colorless. Fluoride substrate **2.5** (0.064 mmol, 14 mg) was added and reaction was stirred at 30 °C for 1–2 hours. Reactions were monitored by GC-FID spectra. Upon completion, reactions were brought out of the glovebox, diluted with DCM, washed with water, dried over Na_2SO_4, and concentrated under reduced pressure. Yields were estimated by use of dimethylsulfone internal standard and integration of olefinic proton from crude reaction mixture. All yields are reported from triplicate data. Products isolated by use of $AgNO_3$-impregnated

silica through flash column chromatography to provide fractionated mixtures. These are additionally fractionated by reverse phase HPLC using an Alltima C18 (5m, 25 cm length, 1 cm internal diameter) column, 98% MeCN/H_2O for 60 minutes at 3mL/min. Impure products re-subjected to reverse phase HPLC (98% MeCN/H_2O) at 1 mL/min 1 – 3 times until sufficiently purified as determined by ^{1}H NMR. Solutions dried over $MgSO_4$ and concentrated to produce purified products.

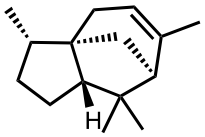




amorpha-4,11-diene (2.6). Yield calculated based on crude ^{1}H NMR estimate utilizing dimethylsulfone as an external standard (4% yield). A small amount of purified sample was obtained by fractionating the crude reaction mixture utilizing $AgNO_3$ impregnated silica gel (100% hexanes). These fractions were subjected to reverse phase HPLC chromatography (98% MeCN/H_2O) multiple times. Spectral data match those reported in the literature.[8]



(±)-α-cedrene (2.7). Yield calculated as mixture with the 2-methyl epimer (**2.8**) based on crude ^{1}H NMR estimate utilizing dimethylsulfone as an external standard (8% yield of two diastereomers). Spectral data match those reported in the literature and matches natural α-cedrene standard.[9]



2-epi-α-cedrene (2.8). Yield calculated as mixture with the 2-methyl epimer (**2.7**) based on crude ^{1}H NMR estimate utilizing dimethylsulfone as an external standard (8% yield of two diastereomers). Spectral data match those reported in the literature.[9]

2.7.3.2 GC-FID and Crude ^{1}H NMR Spectra

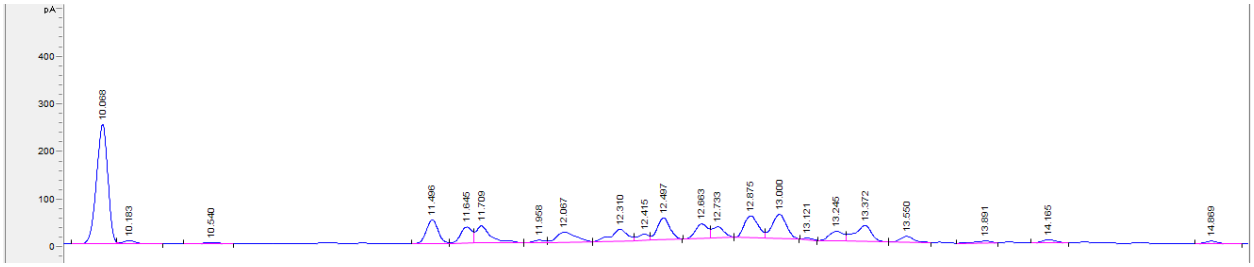


Figure 2.7 Crude GC-FID trace of compound **2.5** cyclization.

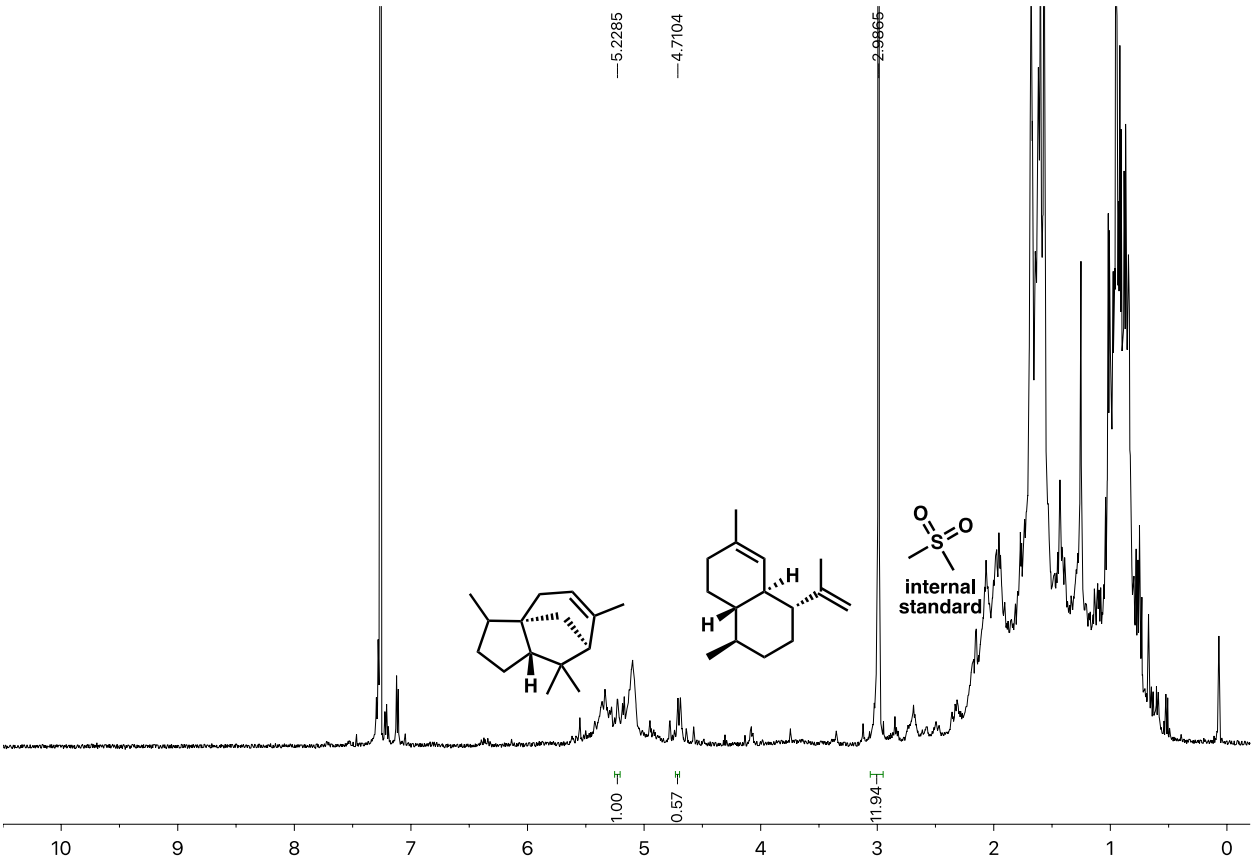


Figure 2.8 Crude ^{1}H NMR (300 MHz, $CDCl_3$) of compound **2.5** cyclization.

2.7.3.3 Reaction Monitoring

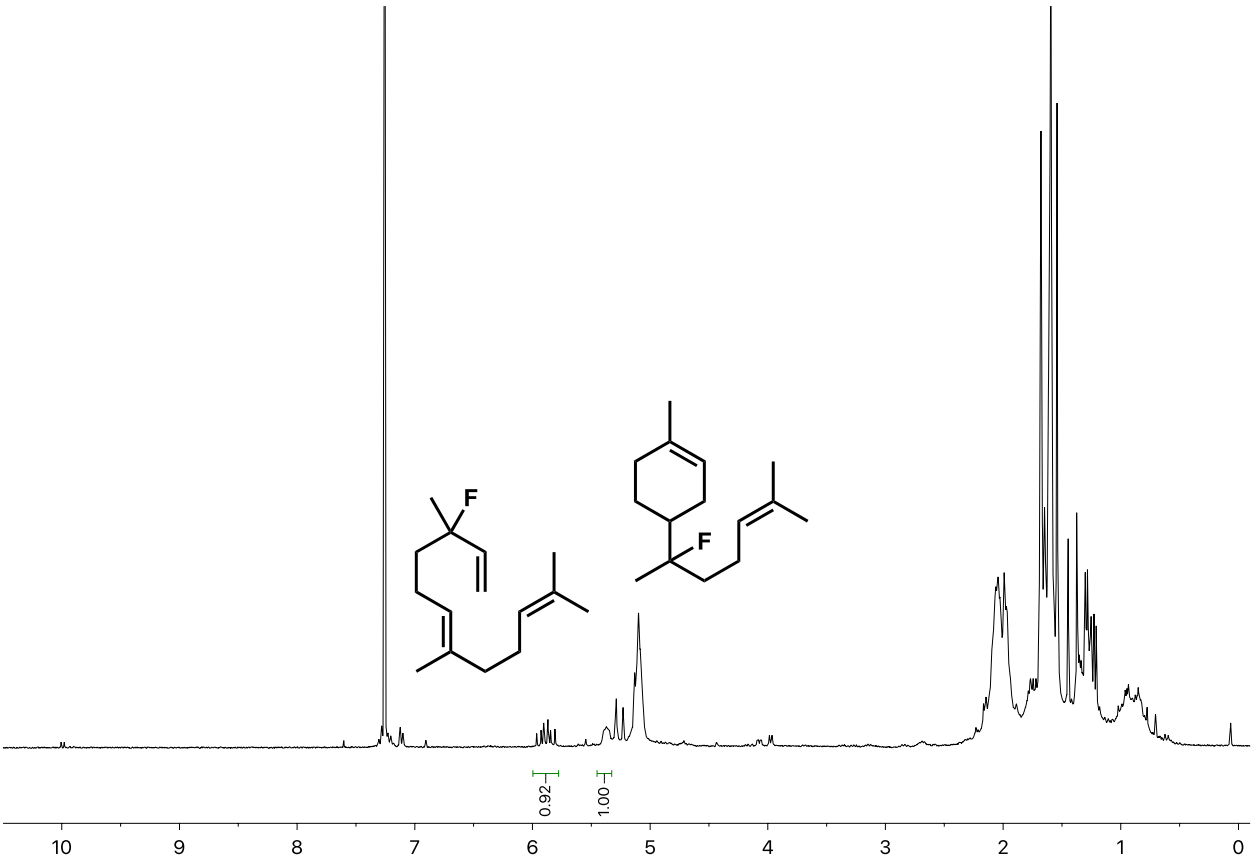
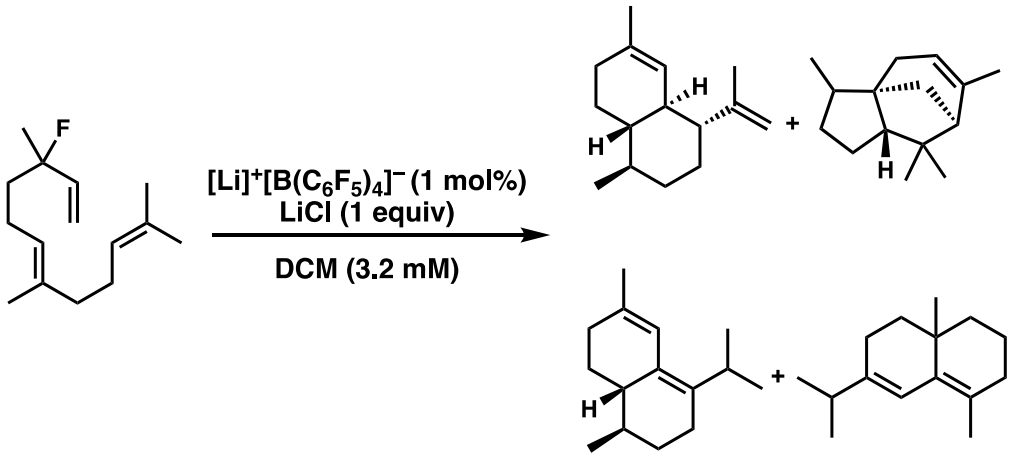


Figure 2.9 Crude ^{1}H NMR (300 MHz, $CDCl_3$) of compound **2.5** cyclization quenched after 5 minutes of reaction time.

2.7.4 Sesquiterpene Li+/WCA Sesquiterpene Tail-to-Head Cyclizations

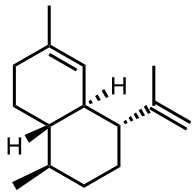
2.7.4.1 General Procedure for Li+/WCA Sesquiterpene TH Cyclization

In a well-kept glovebox, H_2O, $O_2 \leq 0.5$ ppm, a scintillation vial was charged with $[Li]^+[B(C_6F_5)_4]^-$ (0.01 equiv, 0.4 mg) and lithium chloride (1.0 equiv, 1.7 mg). Dichloromethane was added (0.064 mM, 20 mL) and heterogeneous solution was stirred for one minute before addition of fluoride substrate (0.064 mmol). Reactions were stirred at 30 °C for 1–2 hours. Reactions were monitored by GC-FID spectra unless noted otherwise. Upon completion, reactions were brought out of the glovebox, diluted with DCM, washed with water, dried over Na_2SO_4, and concentrated under reduced pressure. Yields were estimated by use of dimethylsulfone internal standard and integration of olefinic proton from crude reaction mixture. All yields are reported from triplicate data. Products isolated by use of $AgNO_3$-impregnated silica through flash column chromatography and/or HPLC.

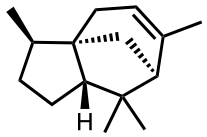


2.7.4.2 Cyclization of 2.5 in Li^+/WCA Sesquiterpene TH Cyclization.

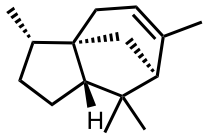
Synthesized according to general procedure 2.7.4.1. In a well-kept glovebox, H_2O, $O_2 \leq$ 0.5 ppm, a scintillation vial was charged with $[Li]^+[B(C_6F_5)_4]^-$ (0.01 equiv, 0.4 mg) and lithium chloride (1.0 equiv, 1.7 mg). Dichloromethane was added (0.064 mM, 20 mL) and heterogeneous solution was stirred for one minute before addition of fluoride substrate **2.5** (0.064 mmol, 14 mg). Reactions were stirred at 30 °C for 1–2 hours until complete by GC-FID reaction monitoring. Upon completion, reactions were brought out of the glovebox, diluted with DCM, washed with water, dried over Na_2SO_4, and concentrated under reduced pressure. Yields were estimated by use of dimethylsulfone internal standard and integration of olefinic proton from crude reaction mixture. All yields are reported from triplicate data. Products isolated by use of $AgNO_3$-impregnated silica through flash column chromatography to provide fractionated mixtures. These are additionally fractionated by reverse phase HPLC using an Alltima C18 (5m, 25 cm length, 1 cm internal diameter) column, 98% MeCN/H_2O for 60 minutes at 3mL/min. Impure products re-subjected to reverse phase HPLC at 98% MeCN/H_2O and 1 mL/min 1 – 3 times until sufficiently purified as determined by ^1H NMR. Solutions dried over $MgSO_4$ and concentrated to produce purified products.



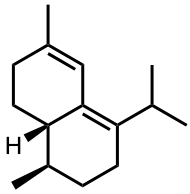
amorpha-4,11-diene (2.6). Yield calculated based on crude ^1H NMR estimate utilizing dimethylsulfone as an external standard (7% yield). Spectral data match those reported in the literature.[8]



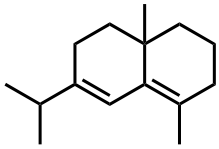
(±)-α-cedrene (2.7). Yield calculated as mixture with the 2-methyl epimer (**2.8**) based on crude ^{1}H NMR estimate utilizing dimethylsulfone as an external standard (27% yield of two diastereomers). Spectral data match those reported in the literature and matches natural α-cedrene standard.[9]



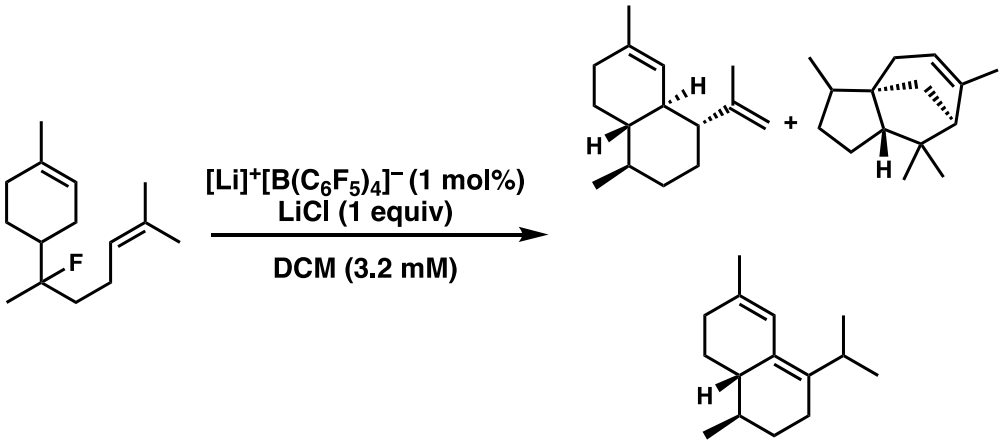
2-epi-α-cedrene (2.8). Yield calculated as mixture with the 2-methyl epimer (**2.7**) based on crude ^{1}H NMR estimate utilizing dimethylsulfone as an external standard (27% yield of two diastereomers). Spectral data match those reported in the literature.[9]



epizonarene (2.9). Yield calculated based on crude ^{1}H NMR estimate utilizing dimethylsulfone as an external standard (30% yield). Spectral data match those reported in the literature.[10]



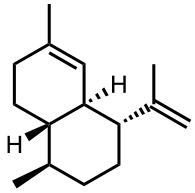
δ-selinene (2.4). Yield calculated based on crude ^{1}H NMR estimate utilizing dimethylsulfone as an external standard (21% yield). Spectral data match those reported in the literature from crude reaction mixture.[11]



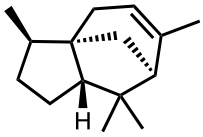
2.7.4.3 Cyclization of 2.16 in Li^{+}/WCA Sesquiterpene TH Cyclization.

Synthesized according to general procedure 2.7.4.1. In a well-kept glovebox, H_2O, $O_2 \leq 0.5$ ppm, a scintillation vial was charged with $[Li]^{+}[B(C_6F_5)_4]^{-}$ (0.01 equiv, 0.4 mg) and lithium chloride (1.0 equiv, 1.7 mg). Dichloromethane was added (0.064 mM, 20 mL) and heterogeneous solution was stirred for one minute before addition of fluoride substrate **2.16** (0.064 mmol, 14 mg). Reactions were stirred at 30 °C for 1–2 hours until complete by GC-FID reaction monitoring. Upon completion, reactions were brought out of the glovebox, diluted with DCM, washed with water, dried over Na_2SO_4, and concentrated under reduced pressure. Yields were estimated by use of dimethylsulfone internal standard and integration of olefinic proton from crude reaction mixture. All yields are reported from triplicate data. Products isolated by use of $AgNO_3$-

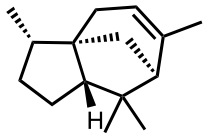
impregnated silica through flash column chromatography to provide fractionated mixtures. These are additionally fractionated by reverse phase HPLC using an Alltima C18 (5m, 25 cm length, 1 cm internal diameter) column, 98% MeCN/H_2O for 60 minutes at 3mL/min. Impure products re-subjected to reverse phase HPLC at 98% MeCN/H_2O and 1 mL/min 1 – 3 times until sufficiently purified. Solutions dried over $MgSO_4$ and concentrated to produce purified products.



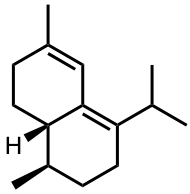
amorpha-4,11-diene (2.6). Yield calculated based on crude ^{1}H NMR estimate utilizing dimethylsulfone as an external standard (9% yield). Spectral data match those reported in the literature.[8]



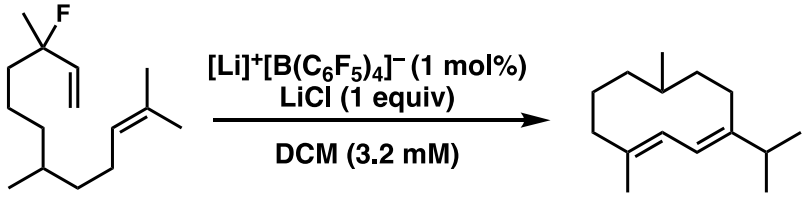
(±)-α-cedrene (2.7). Yield calculated as mixture with the 2-methyl epimer (**2.8**) based on crude ^{1}H NMR estimate utilizing dimethylsulfone as an external standard (31% yield of two diastereomers). Spectral data match those reported in the literature and matches natural α-cedrene standard.[9]



2-epi-α-cedrene (2.8). Yield calculated as mixture with the 2-methyl epimer (**2.7**) based on crude ^{1}H NMR estimate utilizing dimethylsulfone as an external standard (31% yield of two diastereomers). Spectral data match those reported in the literature.[9]



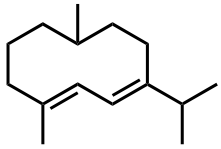
epizonarene (2.9). Yield calculated based on crude ^{1}H NMR estimate utilizing dimethylsulfone as an external standard (44% yield). Spectral data match those reported in the literature.[10]



2.7.4.4 Cyclization of 2.21 in Li^{+}/WCA Sesquiterpene TH Cyclization.

In a well-kept glovebox, H_2O, $O_2 \leq 0.5$ ppm, a scintillation vial was charged with $[Li]^{+}[B(C_6F_5)_4]^{-}$ (0.01 equiv, 0.4 mg) and lithium chloride (1.0 equiv, 1.7 mg). Dichloromethane was added (0.064 mM, 20 mL) and heterogeneous solution was stirred for one minute before addition of fluoride substrate **2.21** (0.064 mmol, 14 mg). Reaction stirred at 30 °C for 1 hour until complete by GC-FID reaction monitoring. Upon completion, reaction was brought out of

the glovebox, diluted with DCM, washed with water, dried over Na_2SO_4, and concentrated under reduced pressure. Yield is estimated by use of dimethylsulfone internal standard and integration of olefinic proton from crude reaction mixture. All yields are reported from triplicate data. Product isolated by use of silica flash column chromatography to provide impure product. This was subjected to reverse phase HPLC using an Alltima C18 (5m, 25 cm length, 1 cm internal diameter) column, 98% MeCN/H_2O for 60 minutes at 2mL/min. Solution dried over $MgSO_4$ and concentrated to produce purified product.



(1E,3E)-4-isopropyl-1,7-dimethylcyclodeca-1,3-diene (2.22). Yield calculated based on crude ^1H NMR estimate utilizing dimethylsulfone as an external standard (57% yield). A small amount of purified compound was obtained by silica flash column chromatography (1% Et_2O/Hexanes) and reverse phase HPLC (98% MeCN/H_2O) to afford a colorless oil.

^1H NMR (500 MHz, $CDCl_3$) δ 5.66 (bs, 2H), 2.58-2.40 (m, 2H), 2.22 (m, 1H), 1.85 (m, 2H), 1.65 (s, 3H), 1.52 - 1.23 (m, 7H), 1.07 (d, J = 6.7 Hz, 3H) 1.02 (d, J = 6.7 Hz, 3H) 0.83 (d, J = 6.7 Hz, 3H).

^{13}C NMR (125 MHz, $CDCl_3$) δ 148.35, 137.75, 124.26, 120.57, 33.61, 31.26, 30.04, 29.71, 29.63, 27.42, 23.08, 22.49, 22.32, 21.66, 21.60.

FTIR (Neat Film NaCl): 2952, 2922, 2867, 1456, 1375, 814, 649 cm^{-1}.

HRMS (EI-MS): Calculated for $C_{15}H_{26}$: 206.2035; Measured: 206.2045.

2.7.4.5 GC-FID and Crude ^{1}H NMR Spectra

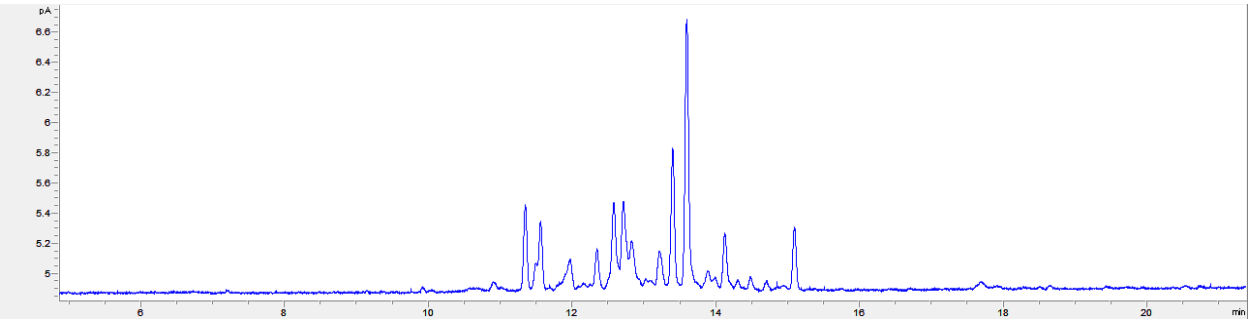


Figure 2.10 Crude GC-FID trace of compound **2.5** cyclization.

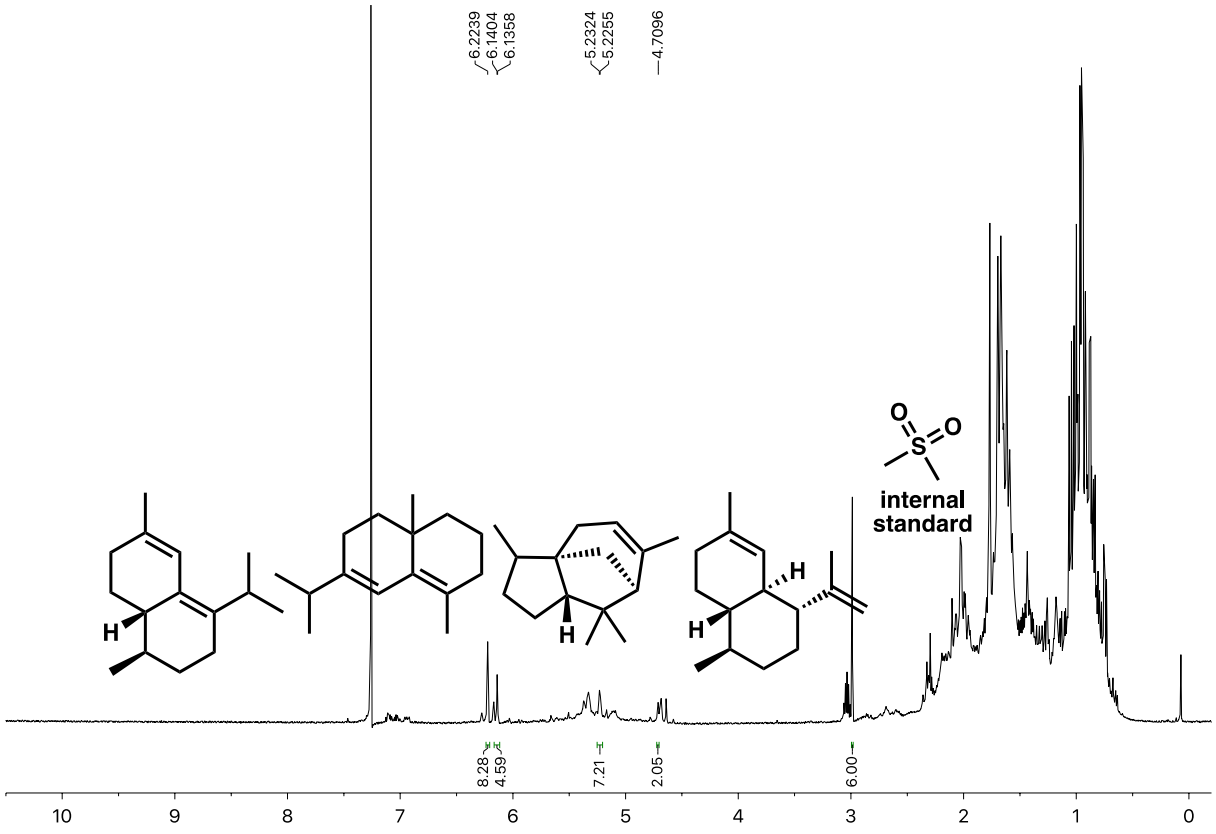


Figure 2.11 Crude ^{1}H NMR (300 MHz, $CDCl_3$) of compound **2.5** cyclization.

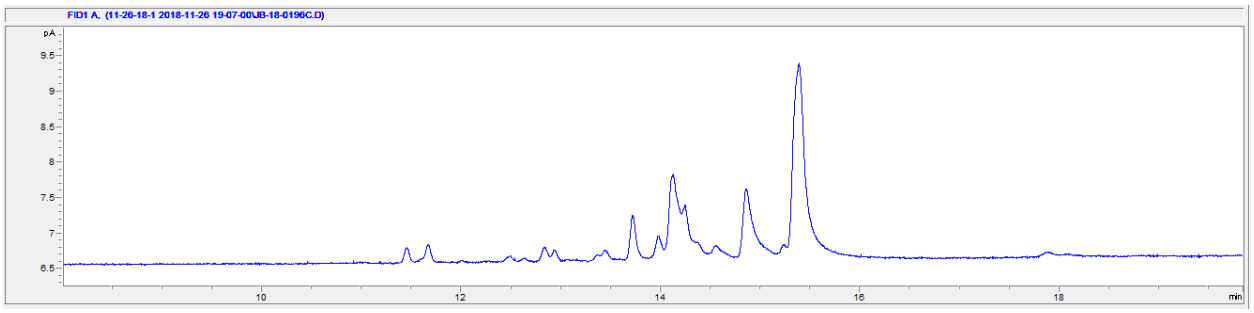


Figure 2.12 Crude GC-FID trace of compound **2.16** cyclization.

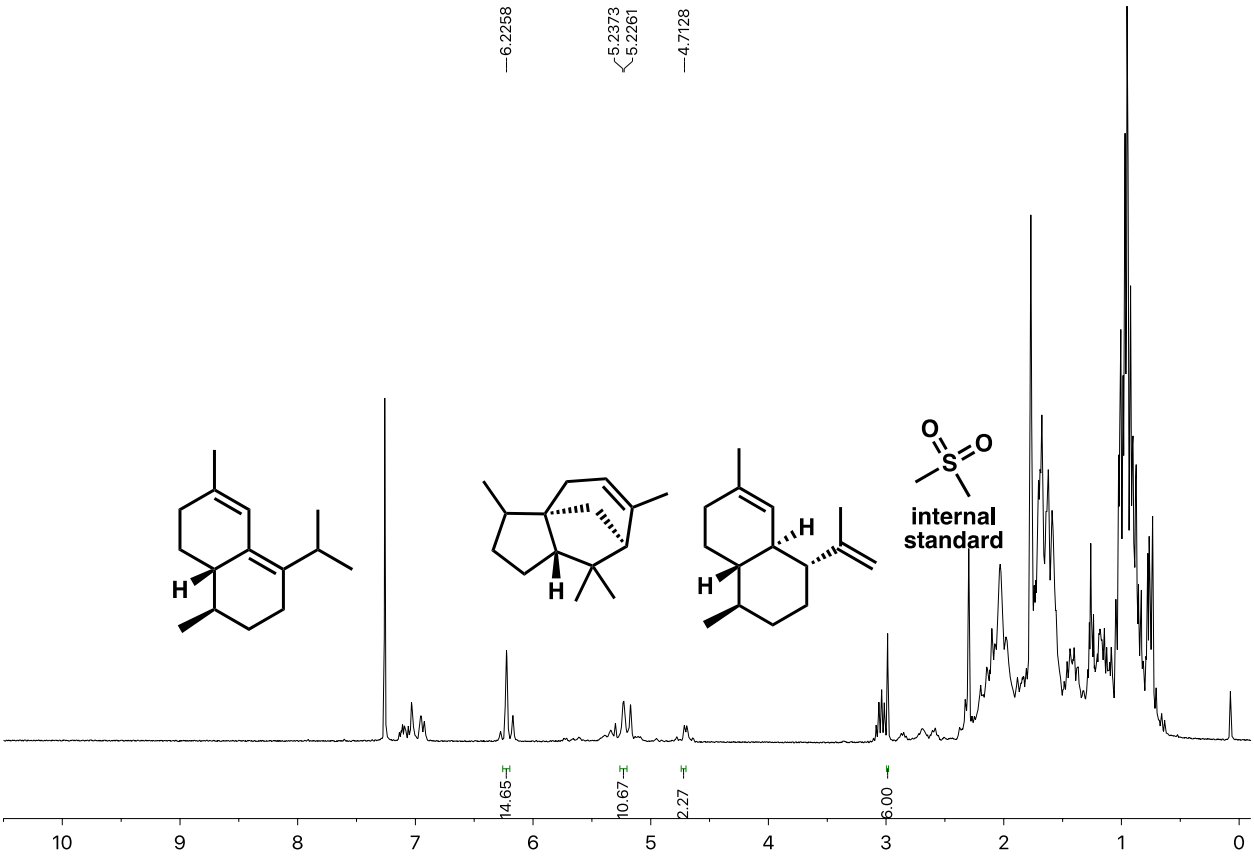


Figure 2.13 Crude ^{1}H NMR (300 MHz, $CDCl_3$) of compound **2.16** cyclization.

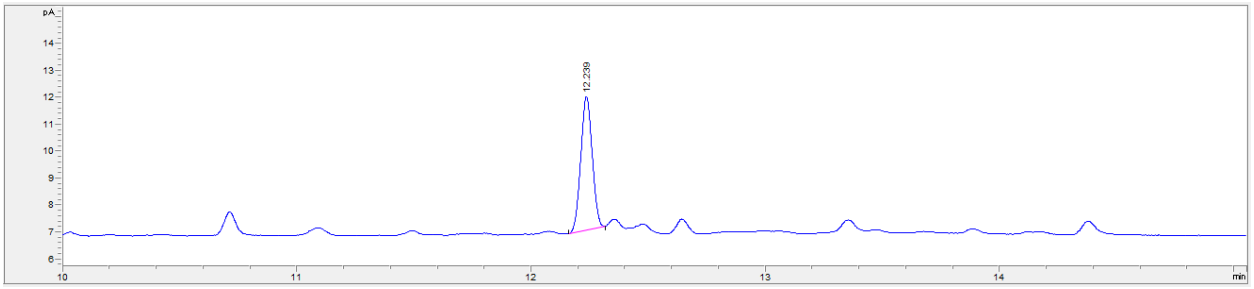


Figure 2.14 Crude GC-FID trace of compound **2.21** cyclization.

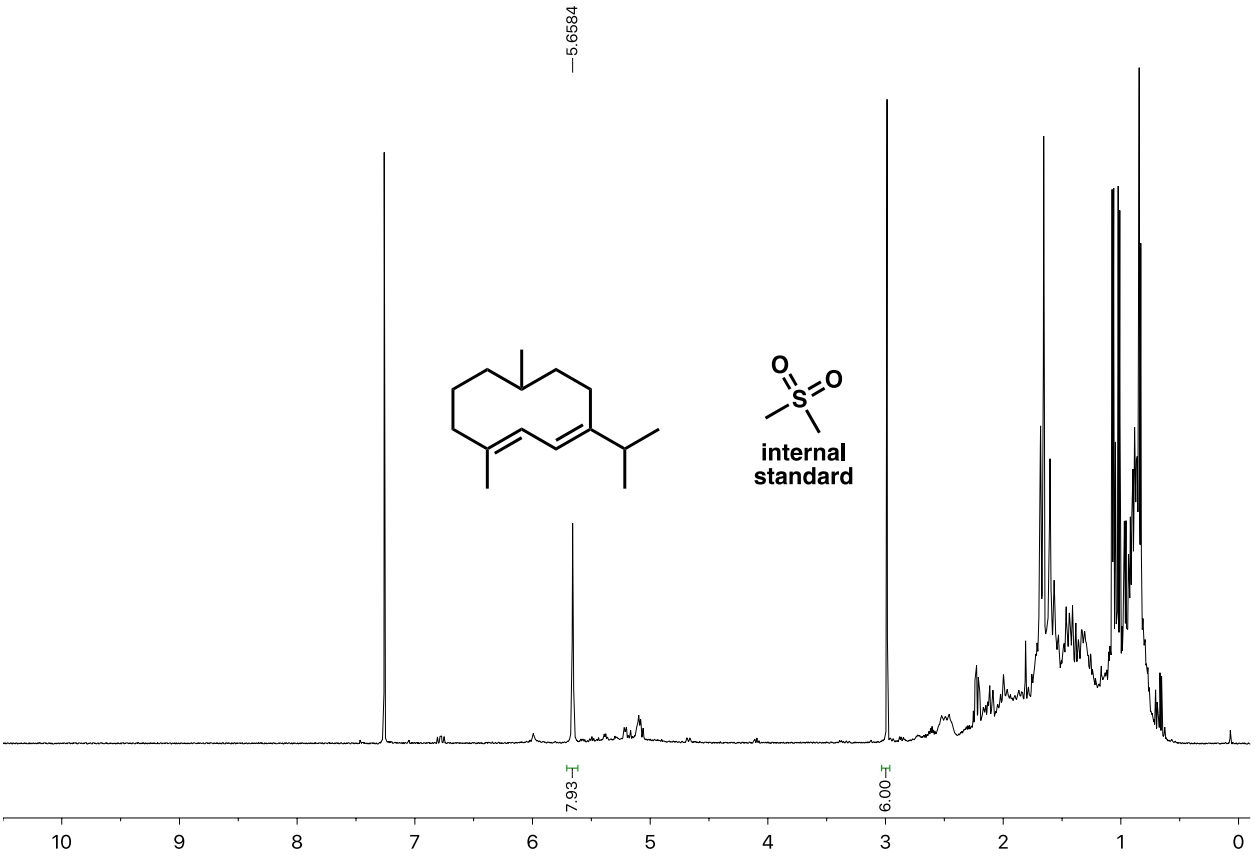
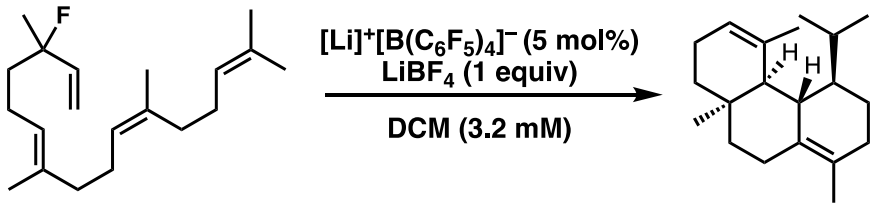


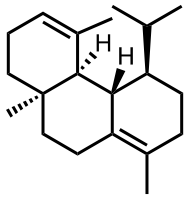
Figure 2.15 Crude ^{1}H NMR (300 MHz, $CDCl_3$) of compound **2.21** cyclization.

2.7.5 Procedure for Diterpene Tail-to-Head Cyclizations

In a well-kept glovebox, H_2O, $O_2 \leq 0.5$ ppm, a scintillation vial was charged with $[Li]^+[B(C_6F_5)_4]^-$ (0.05 equiv, 2.2 mg) and lithium tetrafluoroborate (1.0 equiv, 6.0 mg). Dichloromethane was added (0.064 mM, 20 mL) and heterogeneous solution was stirred for one minute before addition of fluoride substrate **2.10** (0.064 mmol, 19 mg). Reactions were stirred at 30 °C for 1–2 hours. Reactions were monitored by GC-FID spectra unless noted otherwise. Upon completion, reactions were brought out of the glovebox, diluted with DCM, washed with water, dried over Na_2SO_4, and concentrated under reduced pressure. Yields were estimated by use of dimethylsulfone internal standard and integration of olefinic proton from crude reaction mixture. Yields are reported from triplicate data. Product isolated by use of $AgNO_3$-impregnated silica through flash column chromatography to provide fractionated mixtures. These are additionally fractionated by reverse phase HPLC using an Alltima C18 (5m, 25 cm length, 1 cm internal diameter) column, 98% MeCN/H_2O for 60 minutes at 3mL/min. Impure product re-subjected to reverse phase HPLC at 98% MeCN/H_2O and 1 mL/min 3 times until sufficiently purified. Solution dried over $MgSO_4$ and concentrated to produce purified product.



2.7.5.1 Cyclization of 2.10 in Li^+/WCA Diterpene TH Cyclization.



(4aS,4bS,5R,10aR)-5-isopropyl-4,8,10a-trimethyl-1,2,4a,4b,5,6,7,9,10,10a-decahydrophenanthrene (2.11). Yield calculated based on crude ^{1}H NMR estimate utilizing 1,2-dimethoxybenzene as an external standard (8%). Spectral data match those reported in the literature.[12]

2.7.5.2 GC-FID and Crude ^{1}H NMR Spectra

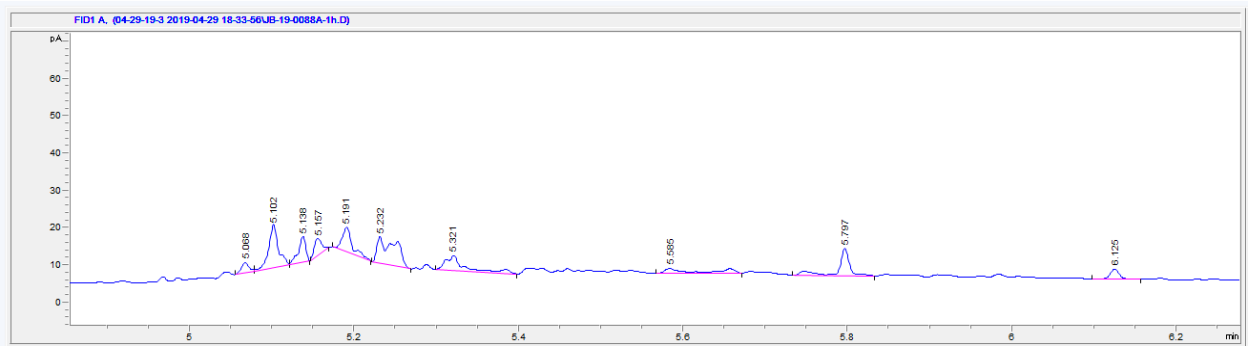


Figure 2.16 Crude GC-FID trace of compound **2.10** cyclization.

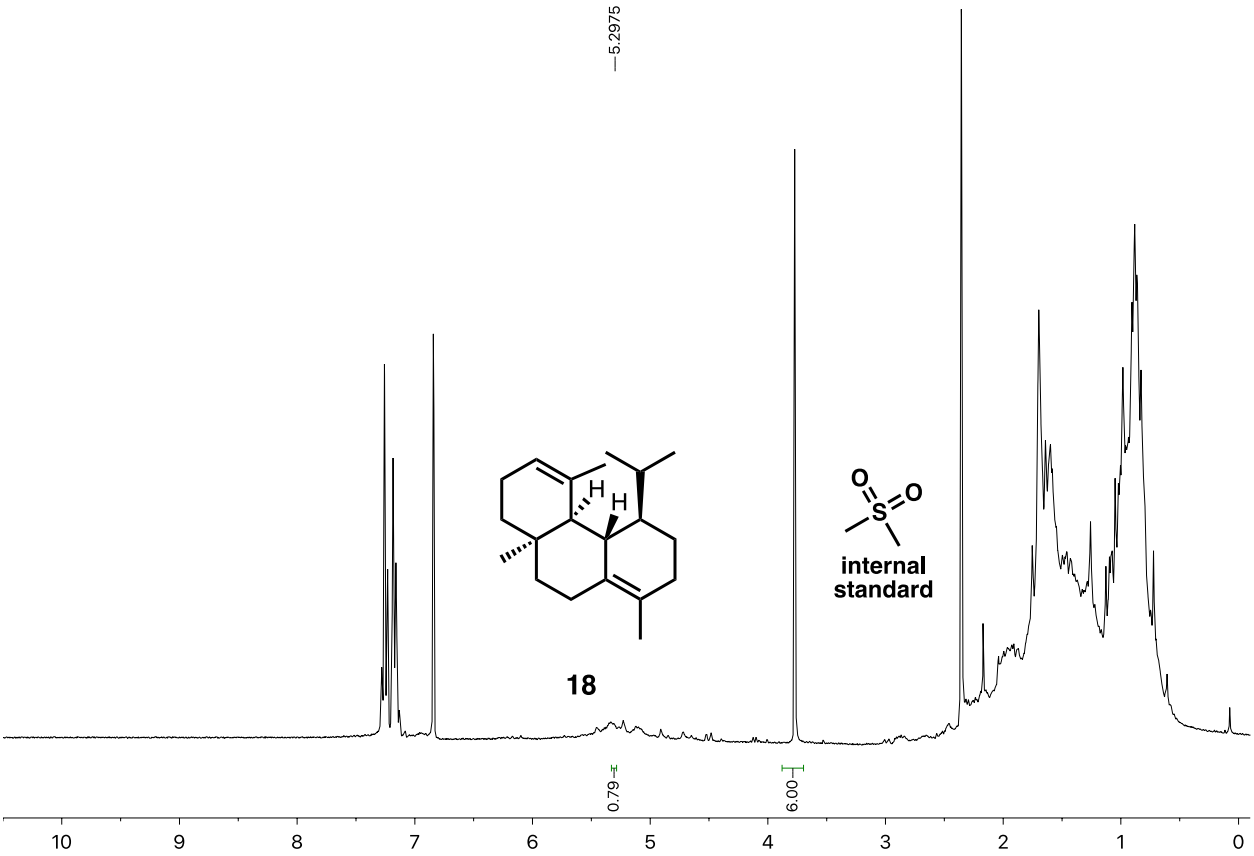


Figure 2.17 Crude ^{1}H NMR (300 MHz, $CDCl_3$) of compound **2.10** cyclization.

2.7.6 ^{1}H, ^{13}C, ^{19}F NMR Spectral Data

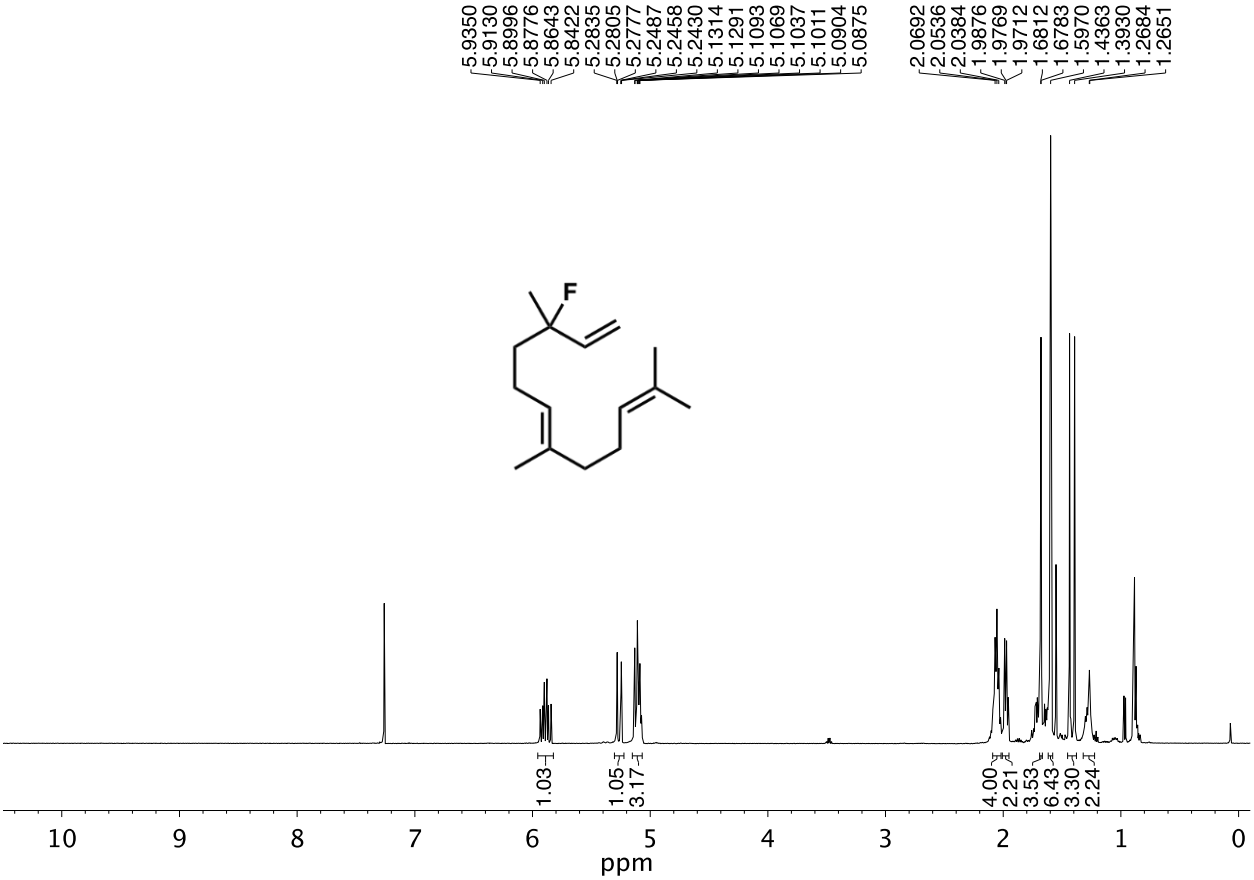


Figure 2.18 ^{1}H NMR (500 MHz, $CDCl_3$) of compound **2.5**.

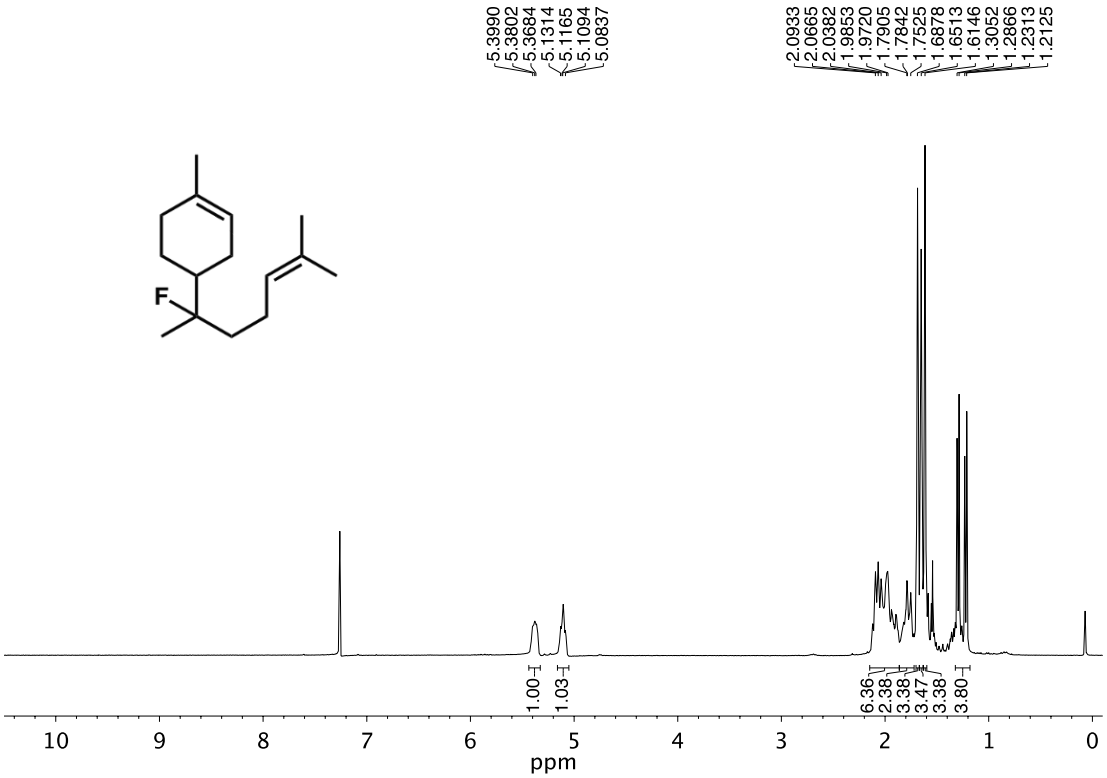


Figure 2.19 ^{1}H NMR (300 MHz, $CDCl_3$) of compound **2.16**.

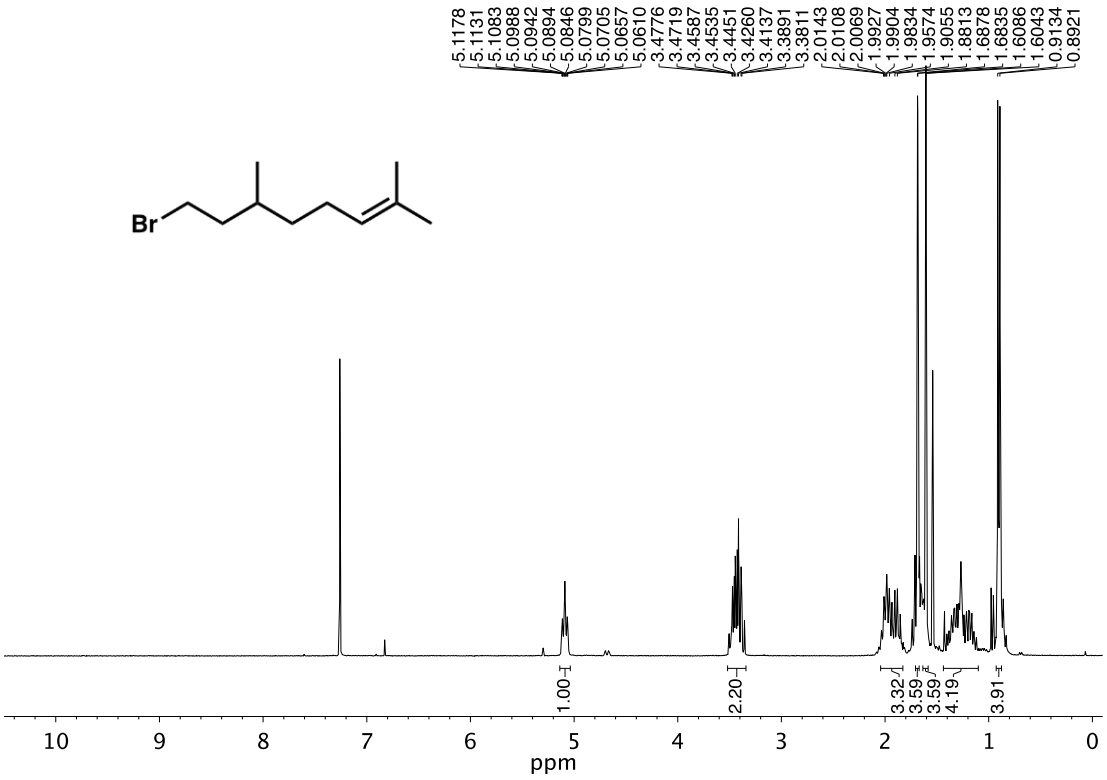


Figure 2.20 ^{1}H NMR (300 MHz, $CDCl_3$) of compound **2.29**.

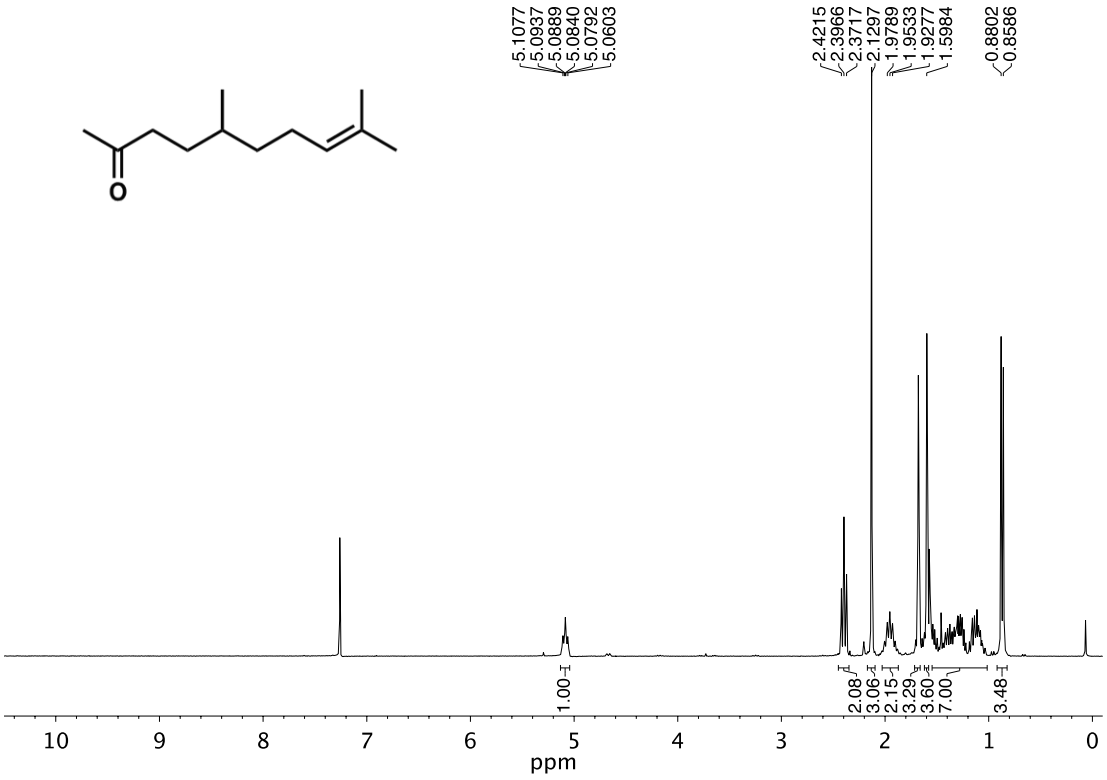


Figure 2.21 ^{1}H NMR (300 MHz, $CDCl_3$) of compound **2.30**.

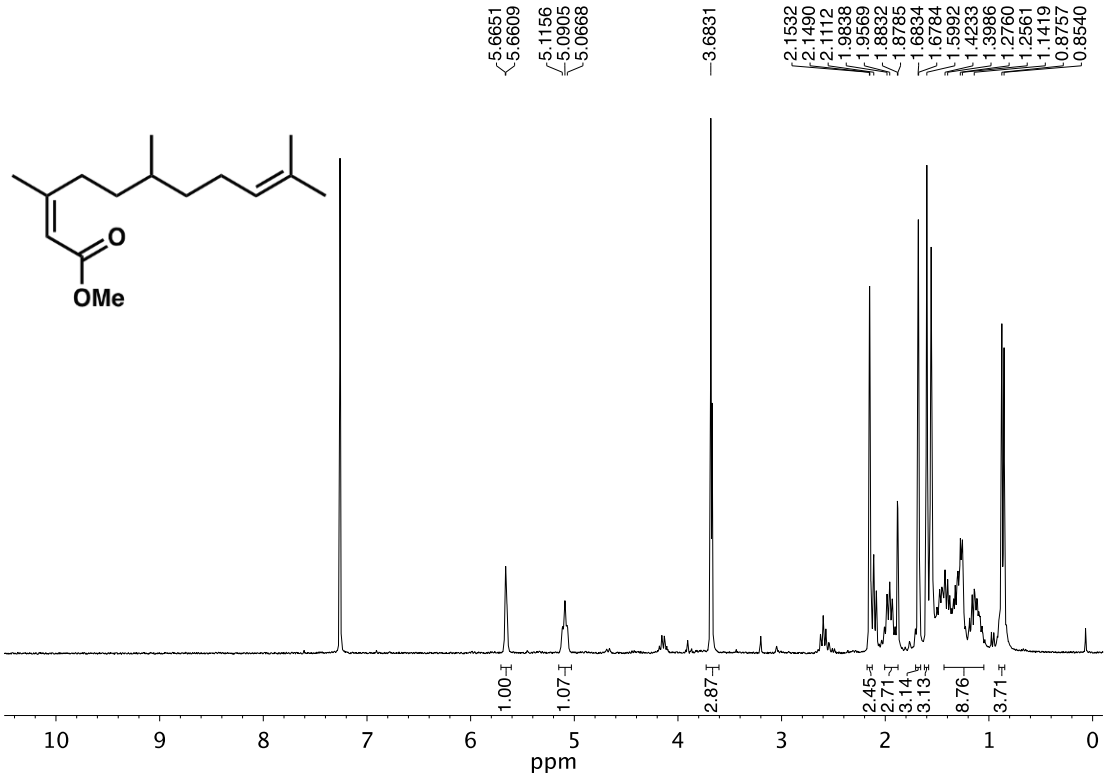


Figure 2.22 ^{1}H NMR (300 MHz, $CDCl_3$) of compound **2.31**.

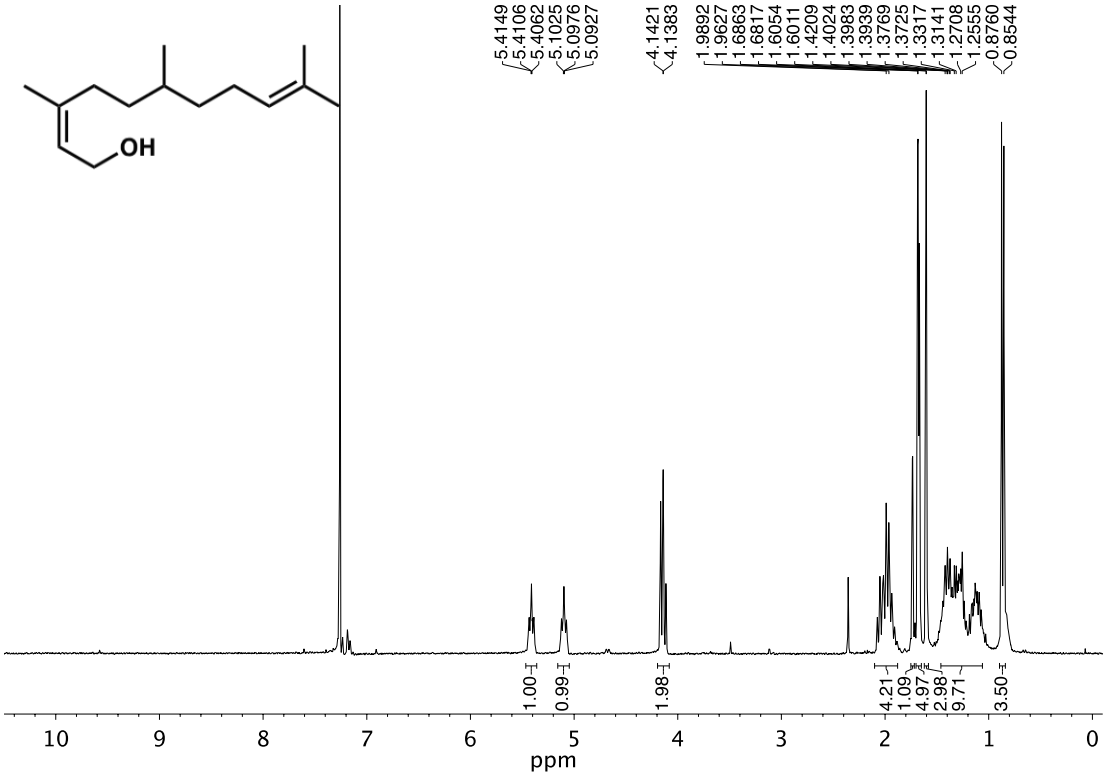


Figure 2.23 ^{1}H NMR (300 MHz, $CDCl_3$) of compound **2.32**.

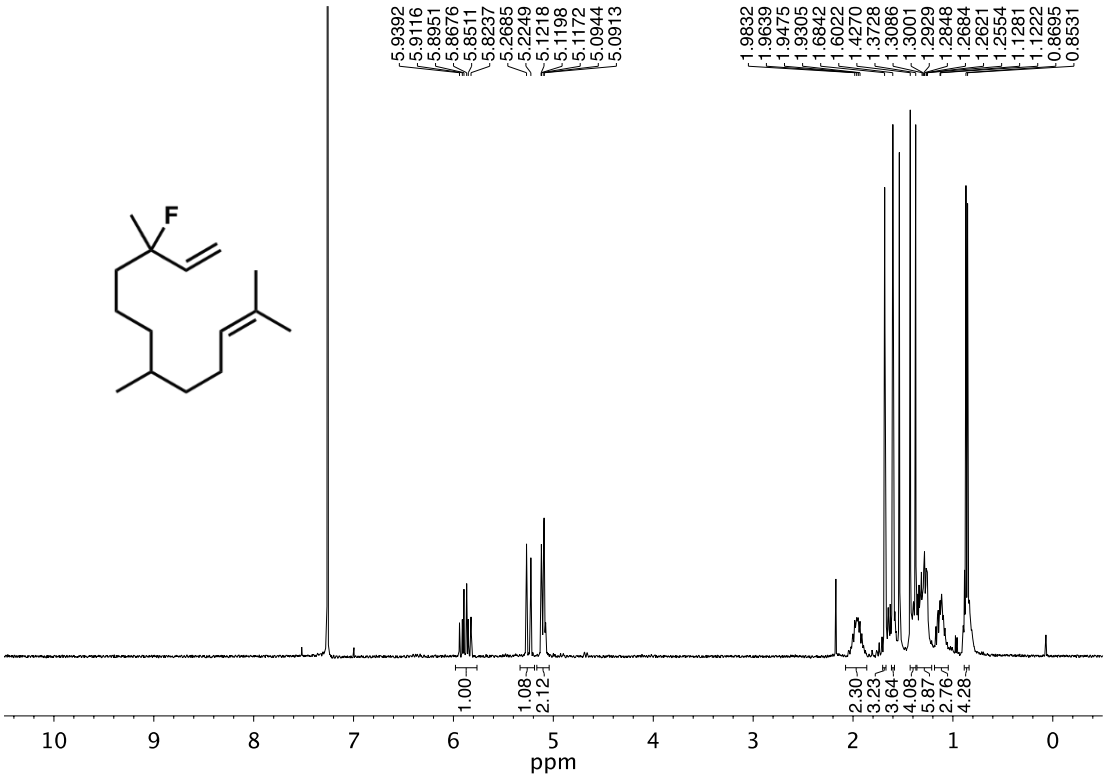


Figure 2.24 ^{1}H NMR (400 MHz, $CDCl_3$) of compound **2.21**.

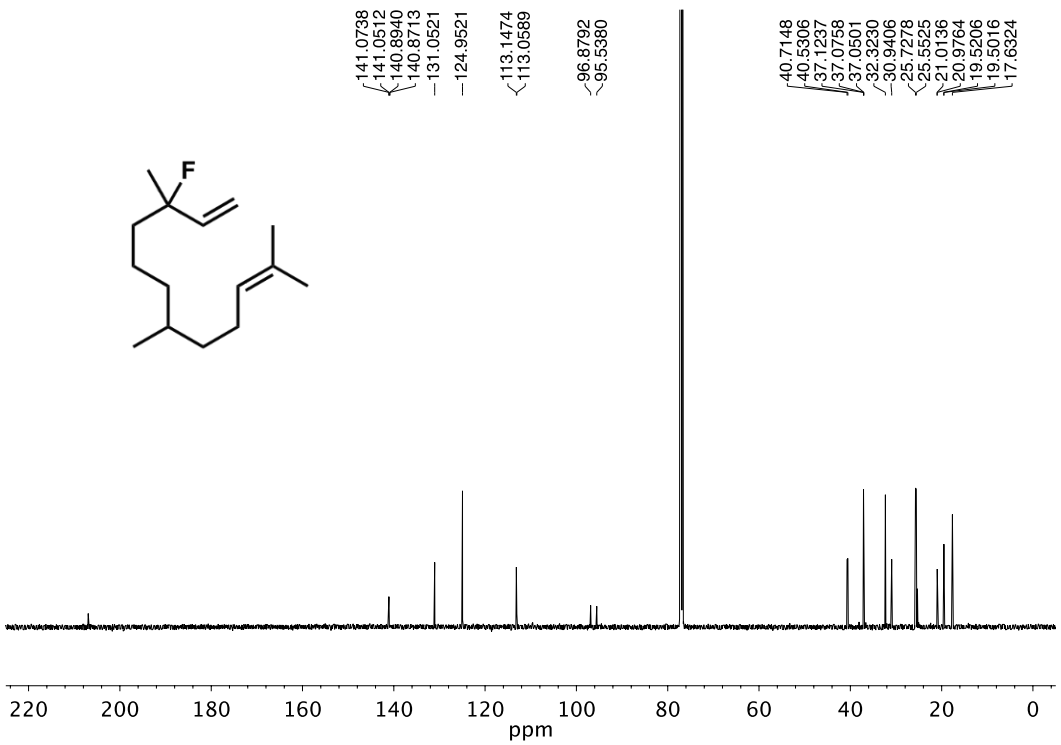


Figure 2.25 ^{13}C NMR (126 MHz, $CDCl_3$) of compound **2.21**.

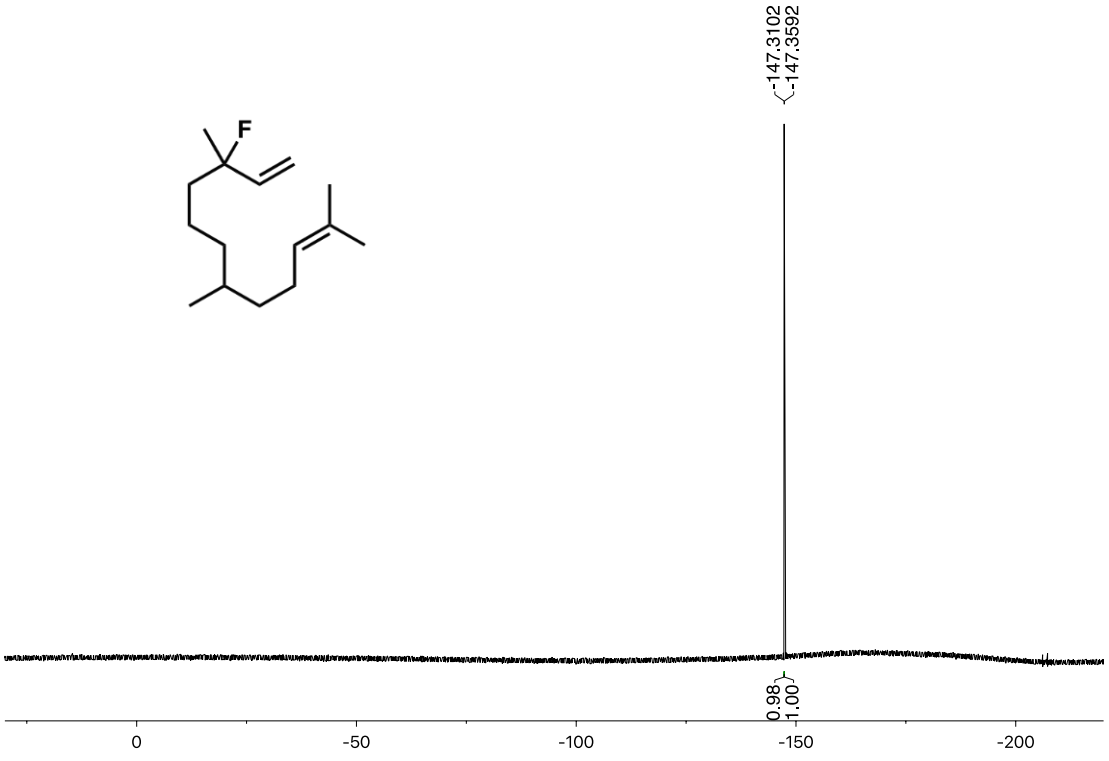


Figure 2.26 ^{19}F NMR (282 MHz, $CDCl_3$) of compound **2.21**.

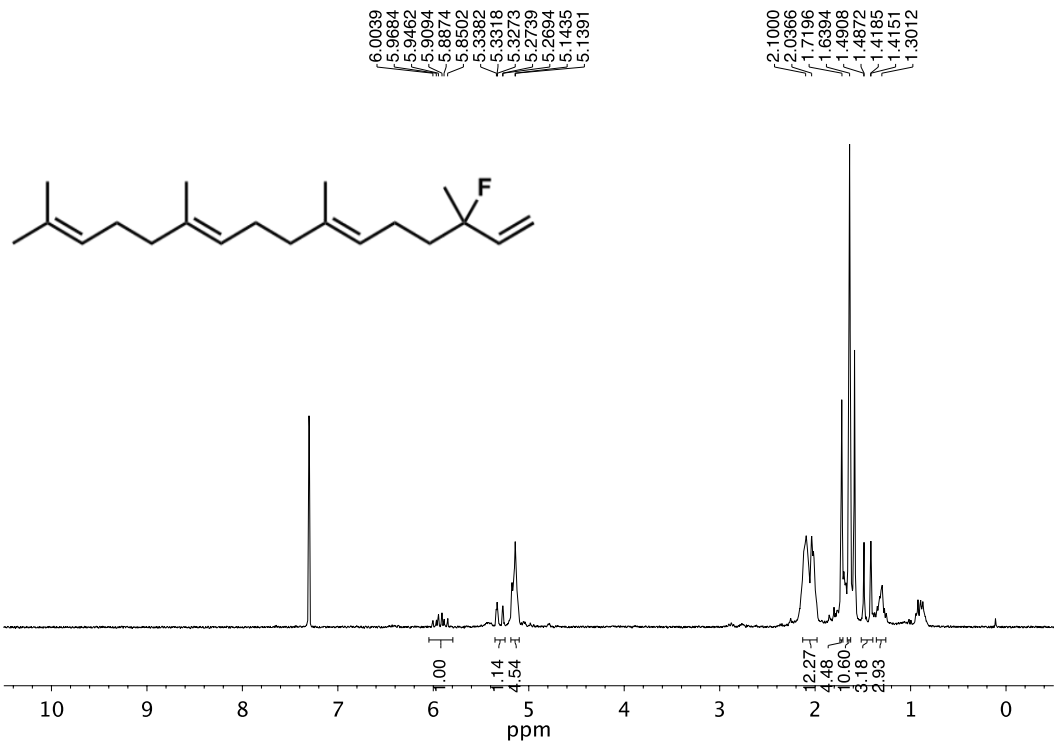


Figure 2.27 ^{1}H NMR (300 MHz, $CDCl_3$) of compound **2.10**.

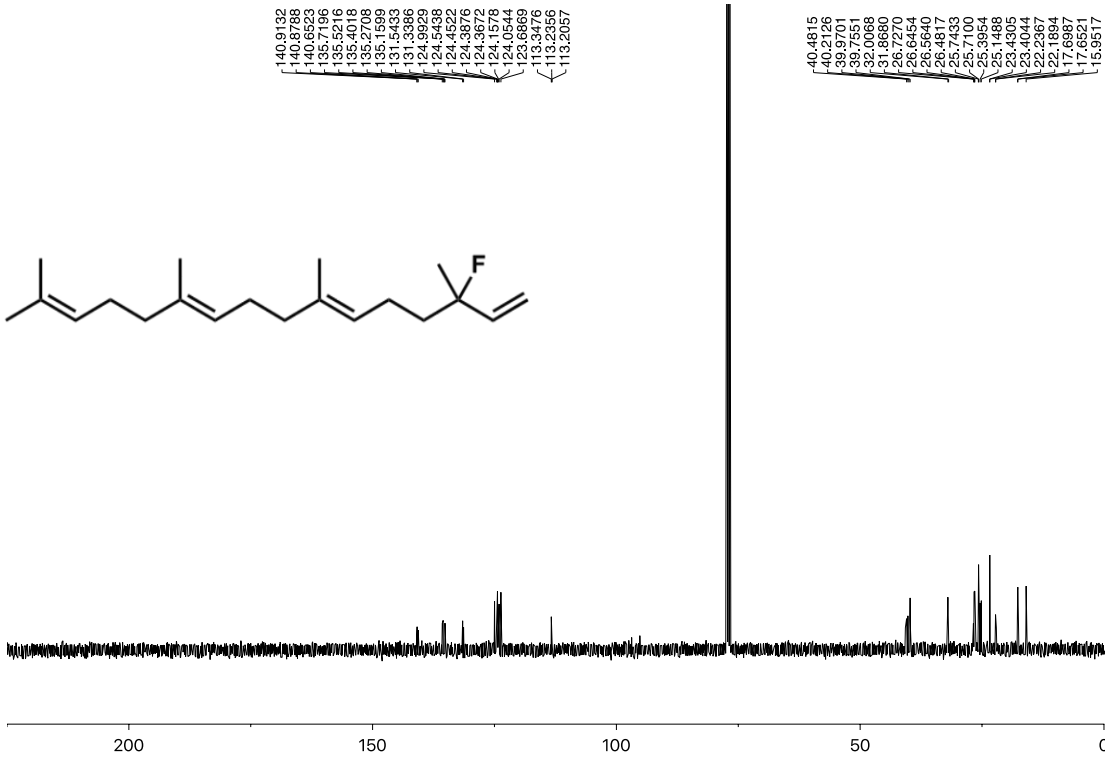


Figure 2.28 ^{13}C NMR (101 MHz, $CDCl_3$) of compound **2.10**.

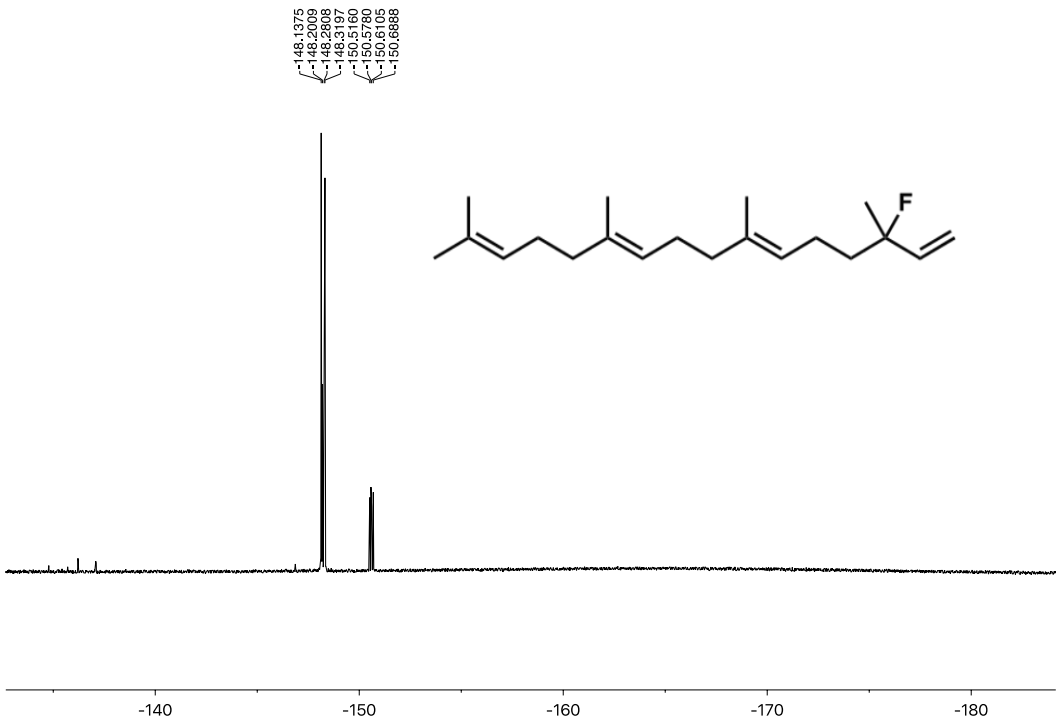


Figure 2.29 ^{19}F NMR (376 MHz, $CDCl_3$) of compound **2.10**.

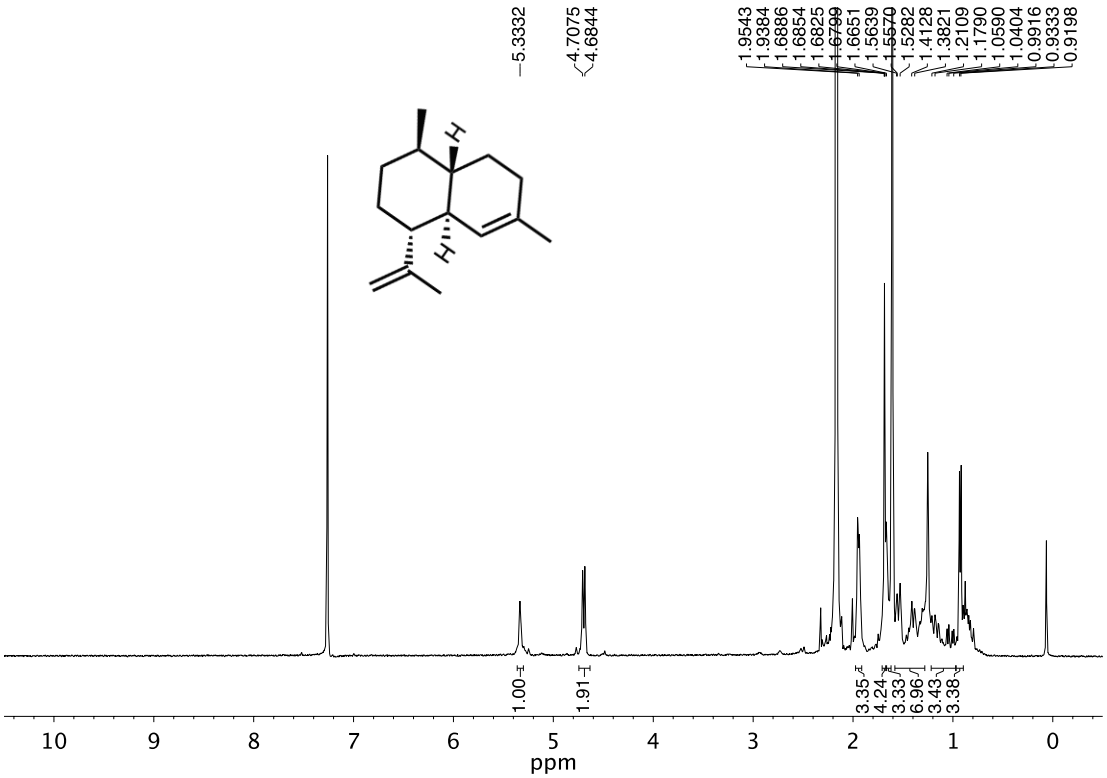


Figure 2.30 ^{1}H NMR (400 MHz, $CDCl_3$) of compound **2.6**.

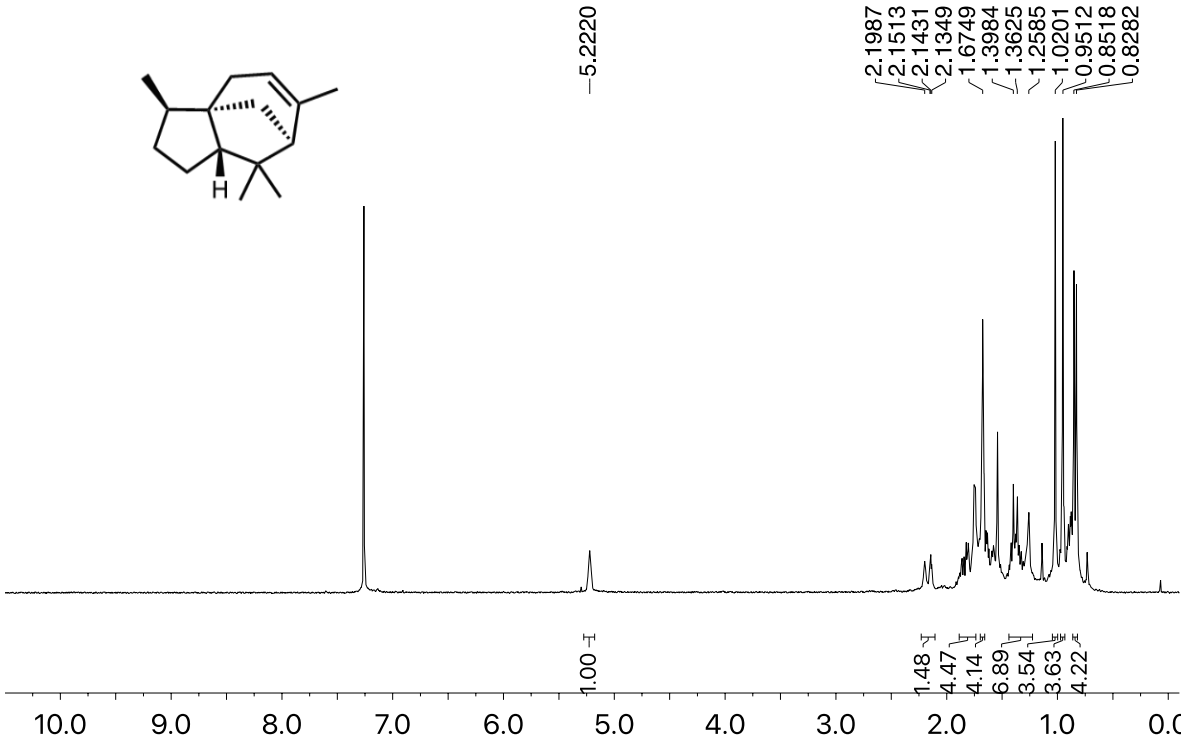


Figure 2.31 ^{1}H NMR (400 MHz, $CDCl_3$) of compound **2.7**.

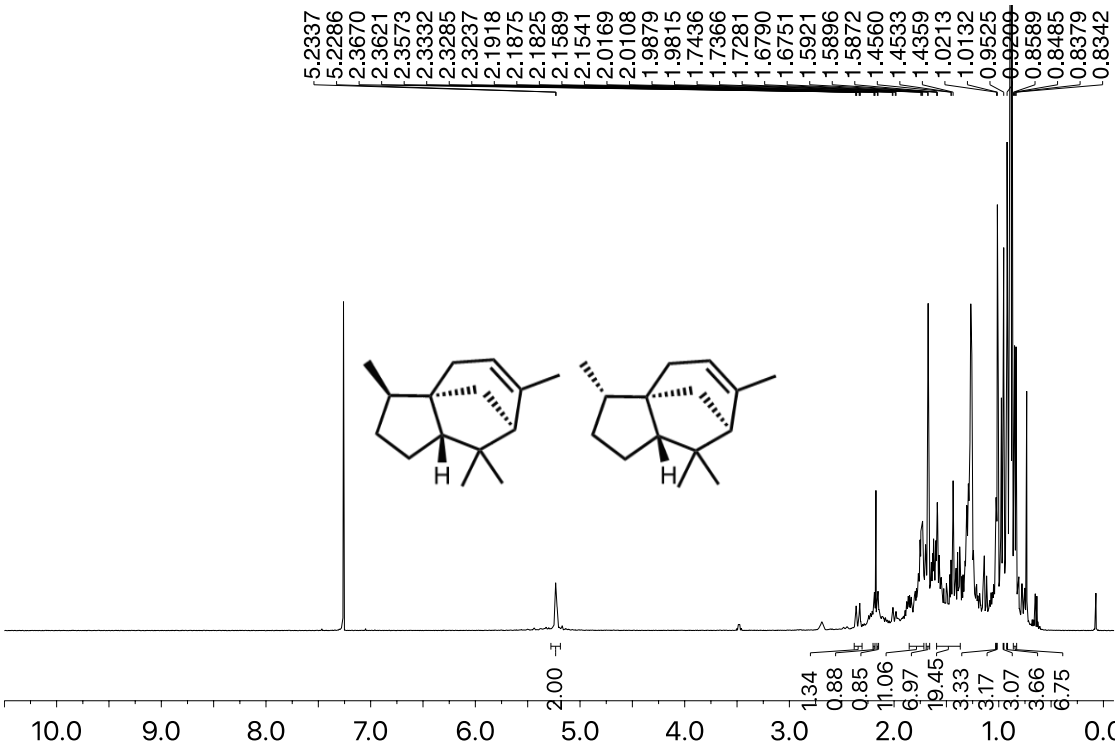


Figure 2.32 ^{1}H NMR (400 MHz, $CDCl_3$) of compound **2.7**/**2.8**.

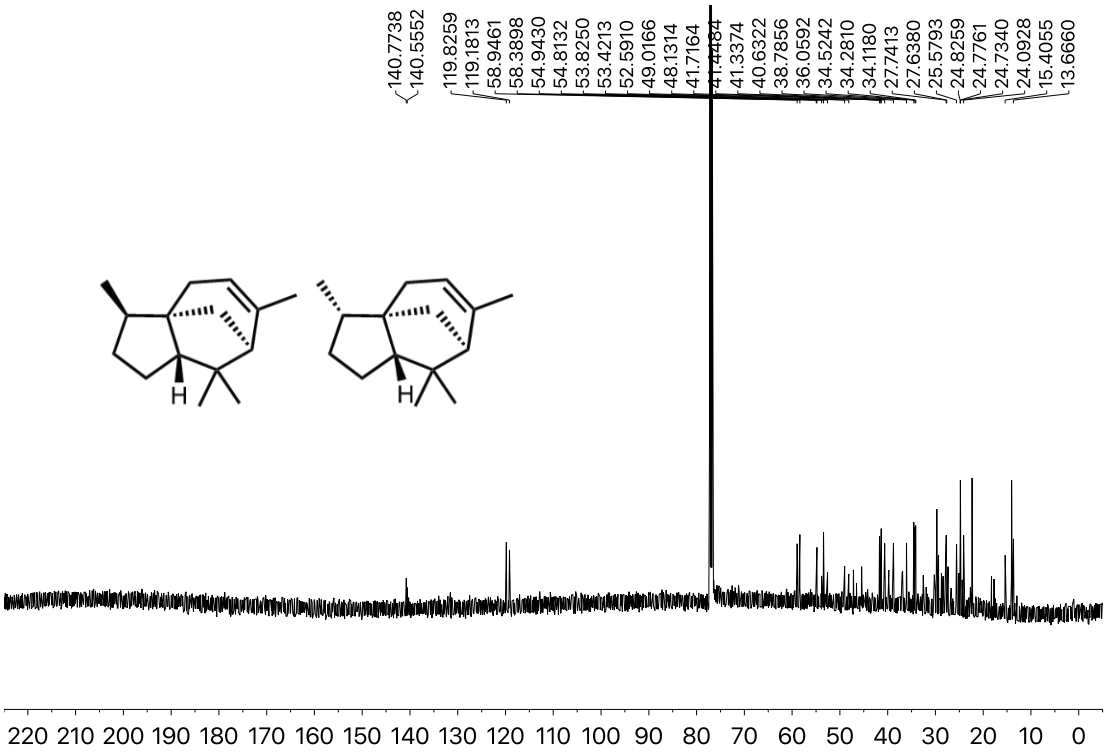


Figure 2.33 ^{13}C NMR (126 MHz, $CDCl_3$) of compounds **2.7/2.8**.

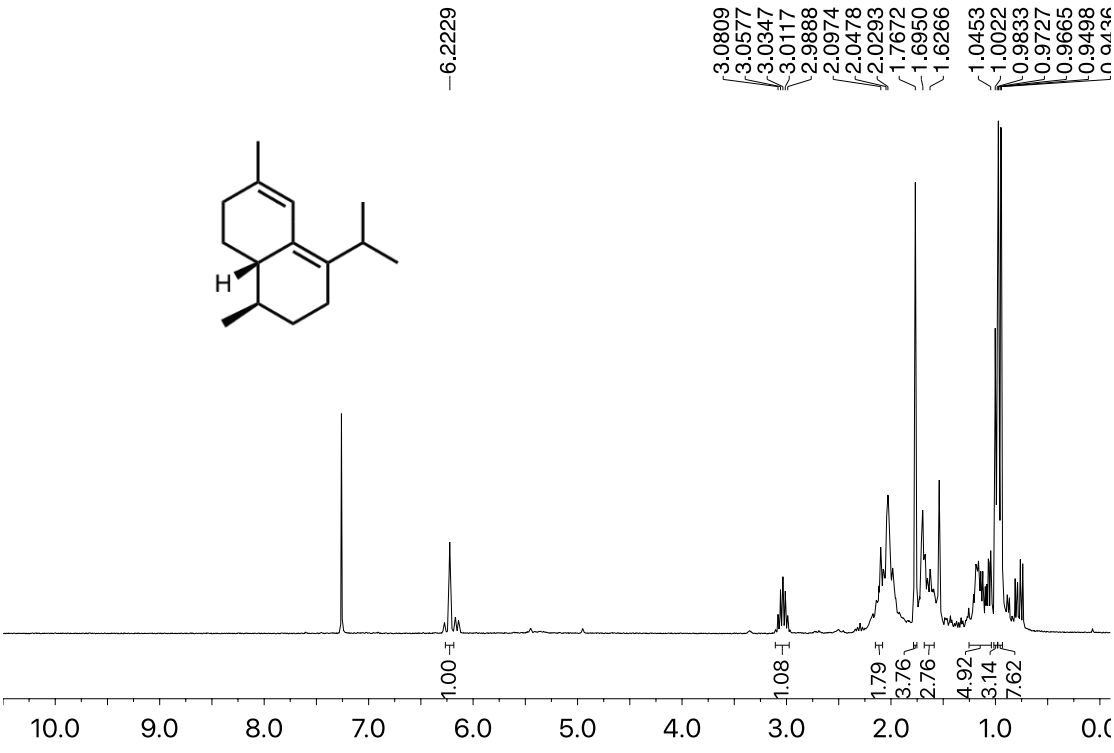


Figure 2.34 ^{1}H NMR (500 MHz, $CDCl_3$) of compound **2.9**.

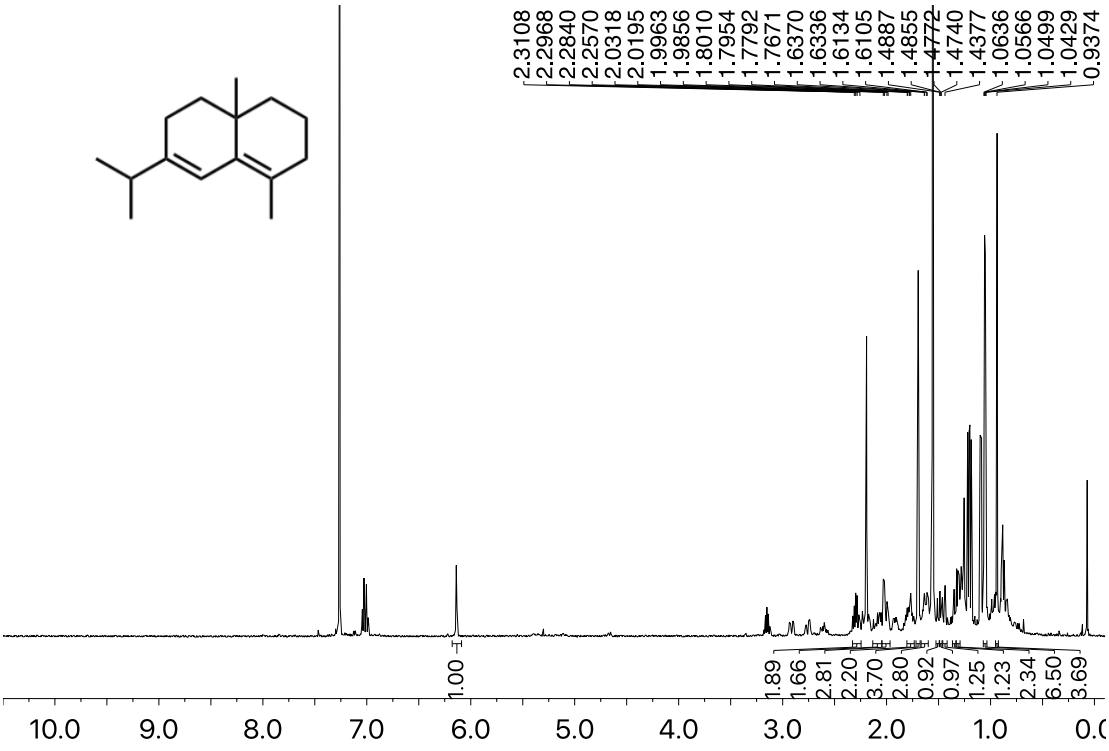


Figure 2.35 ^{1}H NMR (500 MHz, $CDCl_3$) of compound **2.4**.

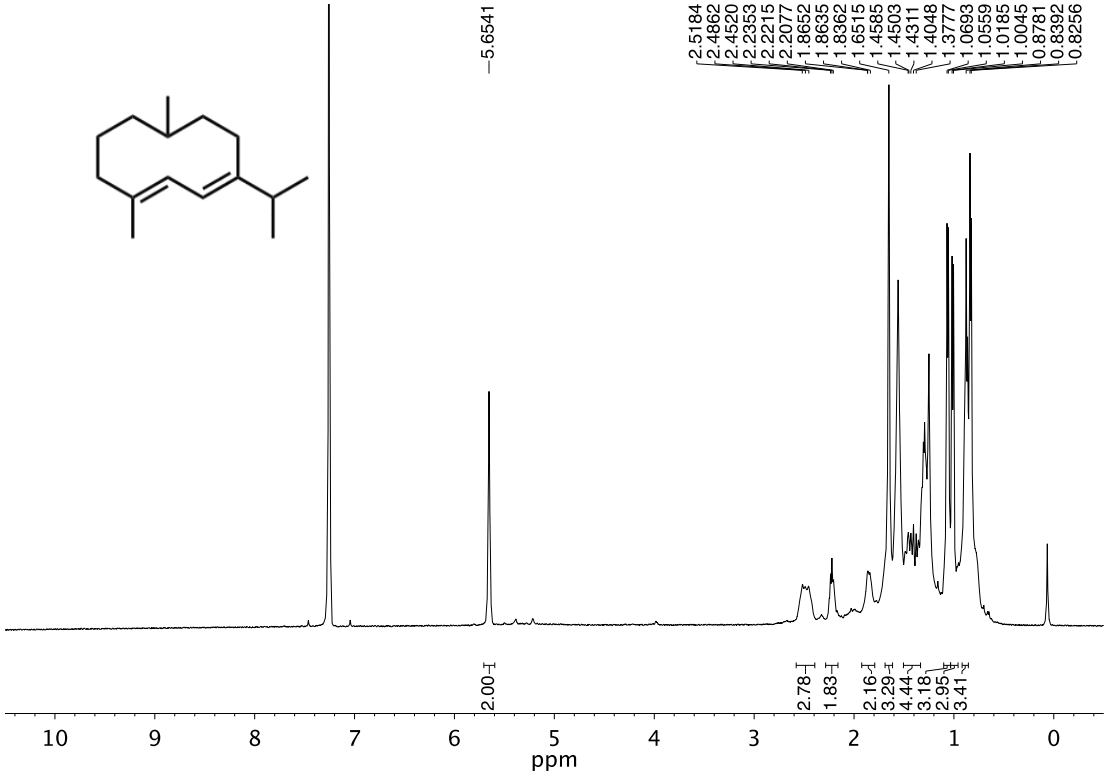


Figure 2.36 ^{1}H NMR (500 MHz, $CDCl_3$) of compound **2.22**.

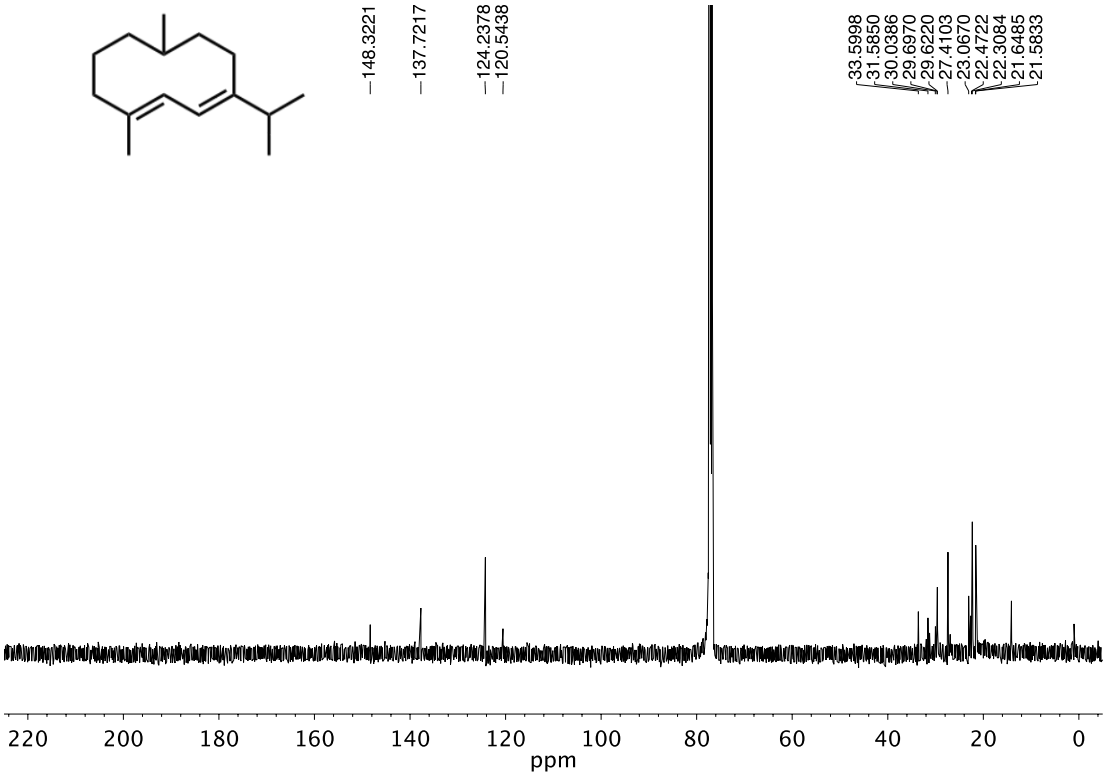


Figure 2.37 ^{13}C NMR (126 MHz, $CDCl_3$) of compound **2.22**.

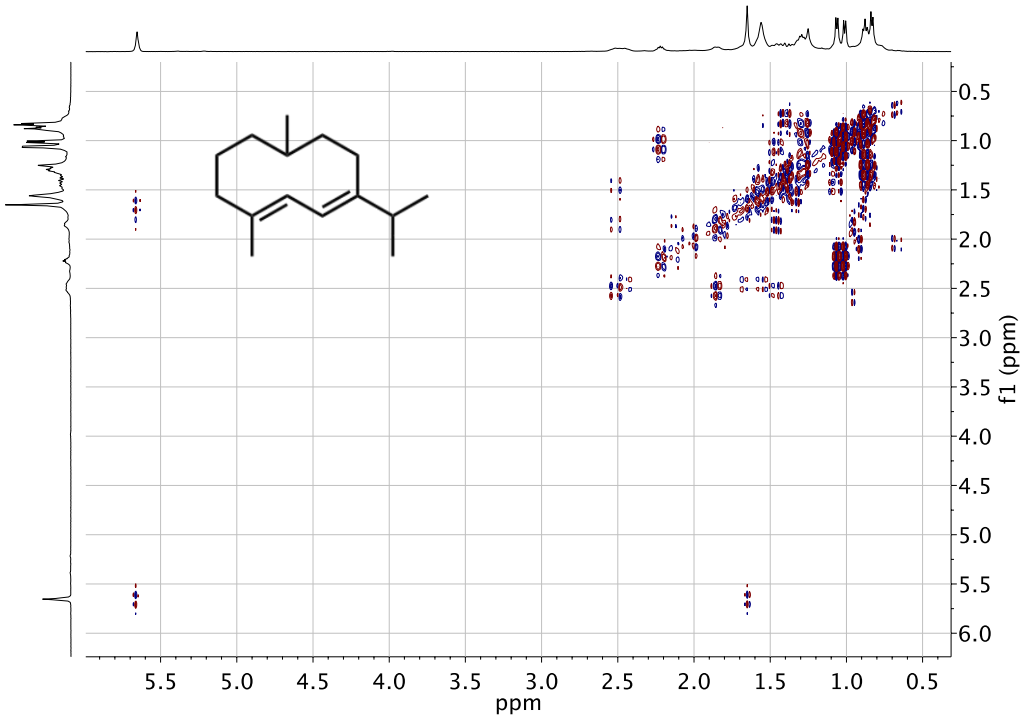


Figure 2.38 COSY NMR (500 MHz, $CDCl_3$) of compound **2.22**.

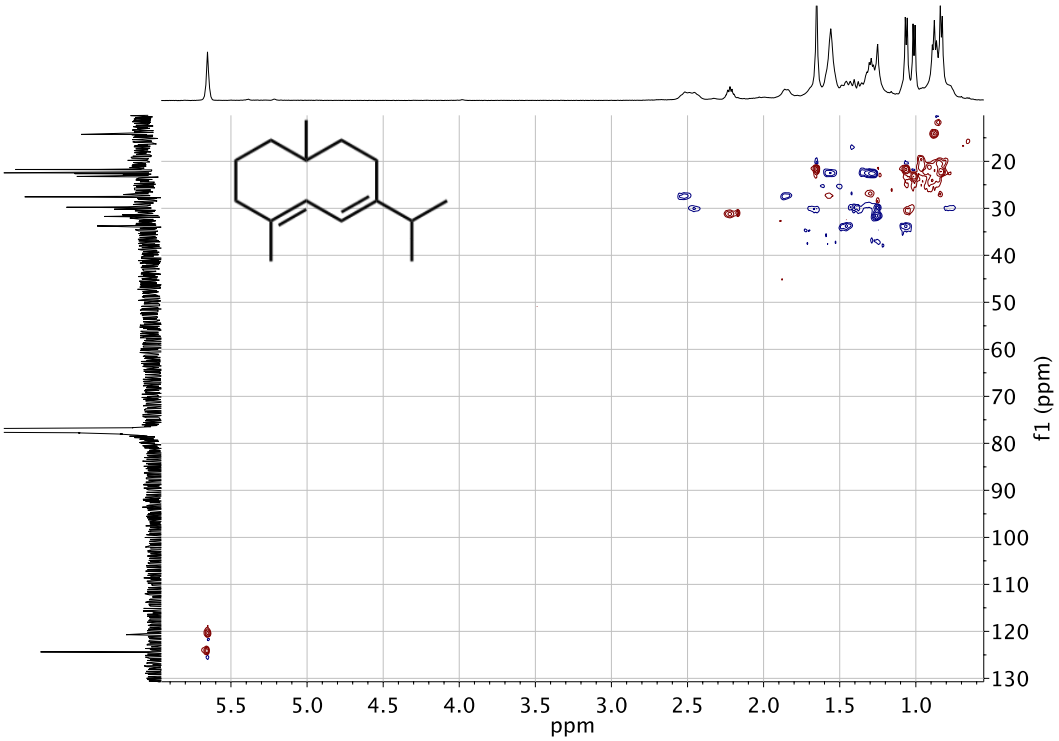


Figure 2.39 HSQC NMR (500 MHz, $CDCl_3$) of compound **2.22**.

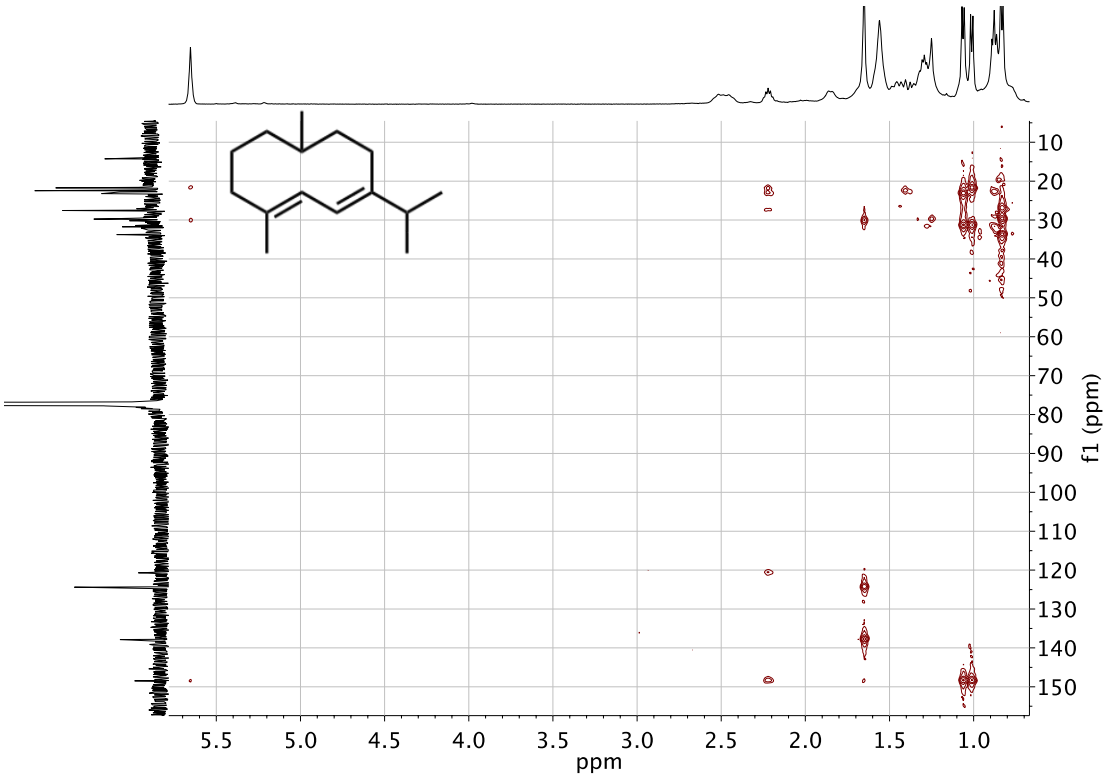


Figure 2.40 HMBC NMR (500 MHz, $CDCl_3$) of compound **2.22**.

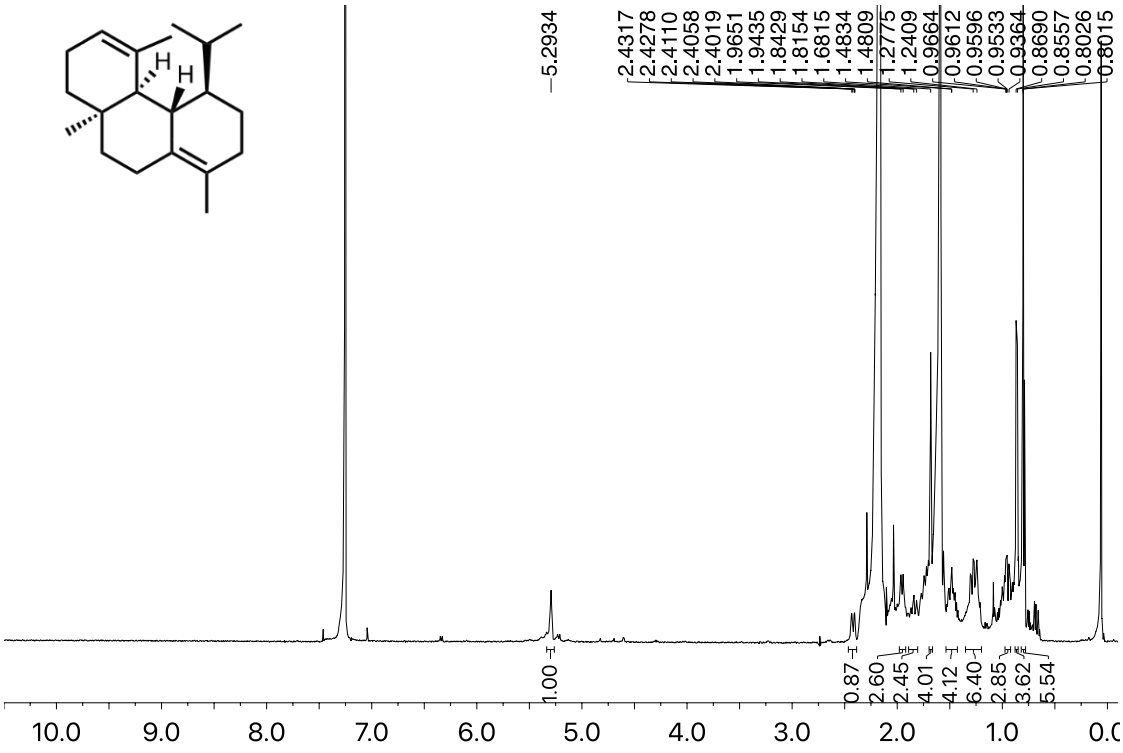


Figure 2.41 ^{1}H NMR (500 MHz, $CDCl_3$) of compound **2.11.**

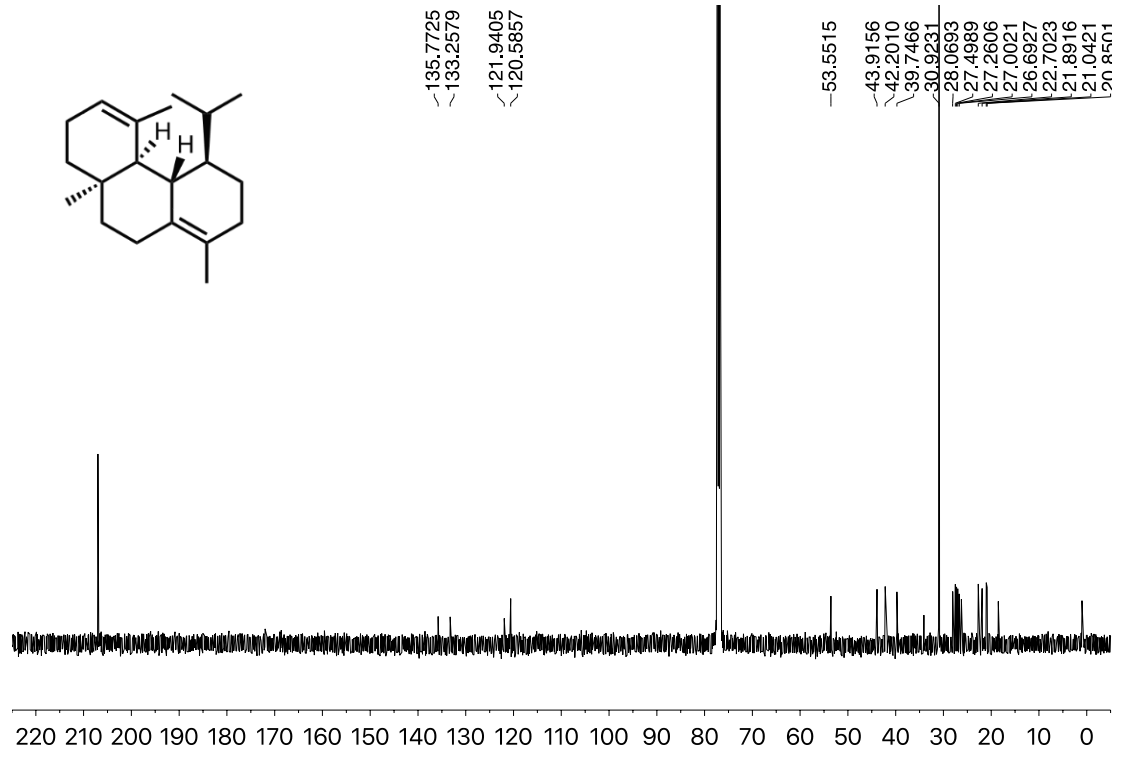


Figure 2.42 ^{13}C NMR (126 MHz, $CDCl_3$) of compound **2.11**.

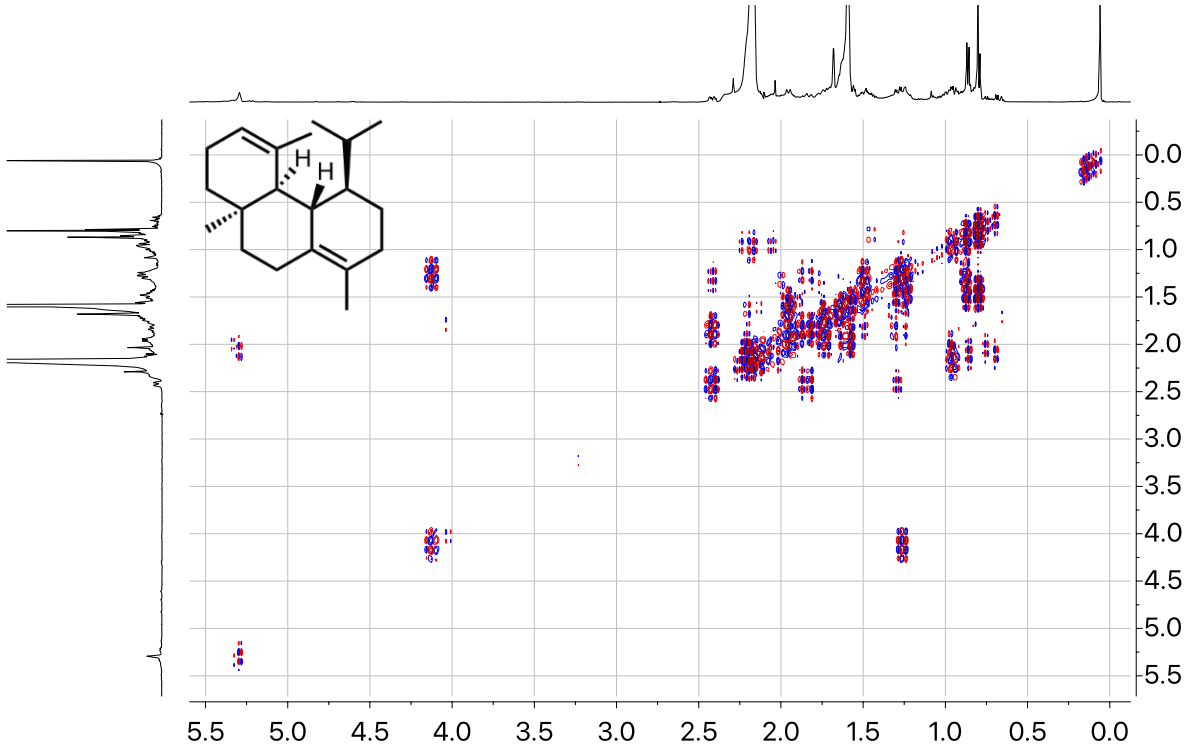


Figure 2.43 COSY NMR (500 MHz, $CDCl_3$) of compound **2.11**.

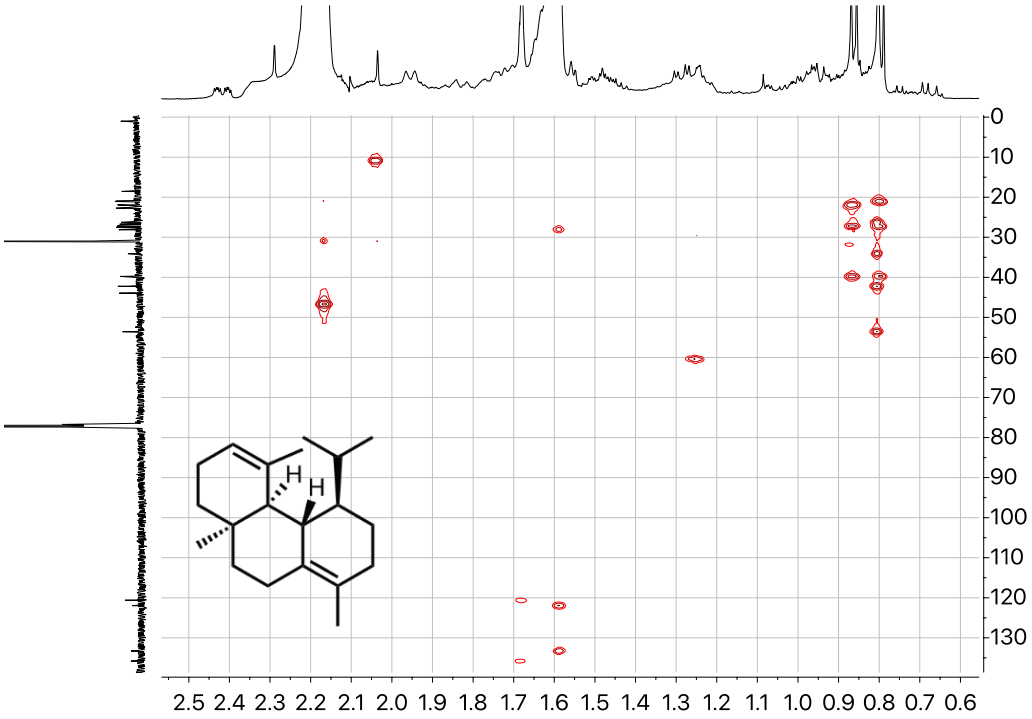


Figure 2.44 HMBC NMR (500 MHz, $CDCl_3$) of compound **2.11**.

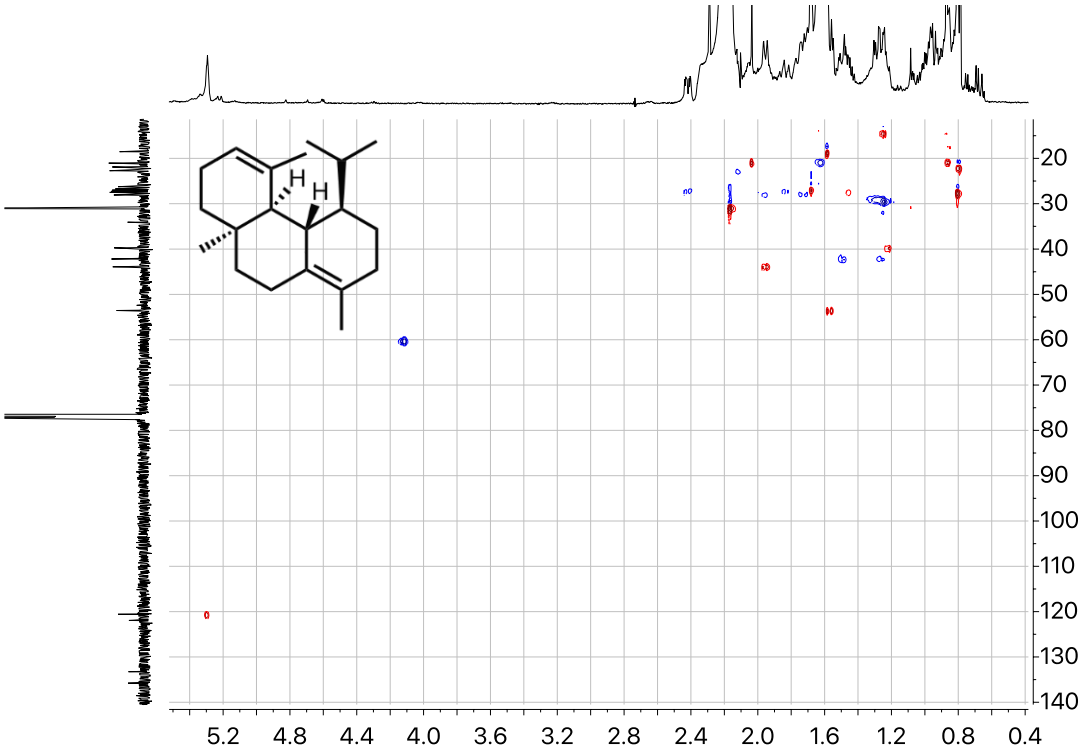


Figure 2.45 HSQC NMR (500 MHz, $CDCl_3$) of compound **2.11**.

2.7.7 Evaluation of Diterpenes by MicroED

2.7.7.1 MicroED procedure

Crude sample extract from cyclization of **2.10** were prepared by working up reaction according to procedure outlined in section 2.7.5 but before purification. The salt contaminant was found after dissolving the concentrated mixture in EtOH and gently heating with a heat gun until homogeneous. This mixture was allowed to evaporate over a period of days until particles were observed. Using a micropipette, 2 μL of suspension onto a pure carbon 200 mesh Cu grid and blotted with a kimwipe. This process was repeated twice and the TEM grid was inserted at room temperature.

Silicate particles were found after working up and performing $AgNO_3$ silica gel fractionation according to procedure found in section 2.7.5. The concentrated mixture was dissolved in EtOH and heated gently with a heat gun until homogeneous and allowed to slowly evaporate over a period of days. Using a micropipette, 2 μL of suspension onto a pure carbon 200 mesh Cu grid and blotted with a kimwipe. This process was repeated twice and the TEM grid was inserted at room temperature.

HPLC purified sample **2.11** was evaluated by crystallization from $CDCl_3$ solvent in an NMR tube over a period of days. A pure carbon 200 mesh Cu grid was dropped into the NMR tube and shaken together with the sample. The grid was removed and gently tapped while held with tweezers to remove excess powder. The sample was initially evaluated at room temperature and found to be beam sensitive, so additional collection took place under cryogenic conditions.

Data was collected on a Thermo Fisher Talos F200C transmission electron microscope operating with an accelerating voltage of 200keV, corresponding to an electron wavelength of 0.0251 Å. Electron diffraction data was collected using a Thermo Fisher CetaD camera.

Screening the TEM grid for microcrystals was performed at 2600x magnification in imaging mode. Crystals selected for data collection were isolated by a selected area aperture. Data was collected by taking images of the diffraction patterns generated by a continuously rotating crystal integrated continuously at a rate of 3 seconds per frame. This rotation was performed at a rate of 0.3° per second with a minimum and maximum tilt range of −65° to +65°. Crystals selected for data collection were isolated by a selected area aperture to reduce the background noise contributions and calibrated to eucentric height to stay in the aperture over the entire tilt range. Samples collected at cryogenic conditions were placed onto a Gatan 626 cryo holder. Slow cooling the sample includes inserting room temperature Gatan 626 cryo holder and cooling to cryogenic temperatures after insertion into the TEM. Plunge frozen samples were frozen in liquid nitrogen, placed onto a liquid nitrogen cooled Gatan 626 cryo holder, and inserted and maintained at cryogenic temperature for the duration of data collection on the electron microscope. All diffraction data was processed using the XDS suite of programs.[13–15] Structure were solved *ab initio* by direct methods in SHELXT or SHELXD. [16–7]

2.7.8 Supplementary Notes & References

(1) Kuprat, M.; Lehmann, M.; Shulz, A; Villinger, A. Synthesis of pentafluorophenyl silver by means of Lewis acid catalysis: Structure of silver solvent complexes. *Organometallics* **2010**, *29*, 1421–1427.

(2) L'Heureux, A.; Beaulieu, F.; Bennetti, C.; Bill, D. R.; Clayton, S.; LaFlamme, F.; Mirmehrabi, M.; Tadayon, S.; Tovell D; Couturier, M. Aminodifluorosulfinium salts: selective fluorination reagents with enhanced thermal stability and ease of handling. *J. Org. Chem.* **2010**, *75*, 3401–3411.

(3) Sladojevich, F.; Arlow, S. I.; Tang, P.; Ritter, T. Late-stage deoxyfluorination of alcohols with PhenoFluor. *J. Am. Chem. Soc.* **2013**, *135*, 2470–2473.

(4) Dryzhakov, M.; Richmond, E.; Li, G.; Moran, J. Catalytic $B(C_6F_5)_3 \bullet H_2O$-promoted defluorinative functionalization of tertiary aliphatic fluorides *J. Fluorine Chem.* **2017**, *193*, 45–51.

(5) Kim, J.; Matsuyama, S.; Suzuki, T. Deuterated analogues of 4,8-dimethyldecanal, the aggregation pheromone of *Tribolium castaneum*: synthesis and pheromonal activity *J. Label Compd. Radiopharm.* **2004**, *47*, 921–934.

(6) Kaihara, K.; Toyomi, K.; Numata, H.; Ohfune, Y.; Shinada, T. Structure–activity relationship of novel juvenile hormone, $JHSB_3$, isolated from the stink bug, *Plautia stali Tetrahedron* **2012**, *68*, 106–113.

(7) Su, C.; Hopson, R.; Williard, P. G. Isotopically enriched ^{13}C diffusion-ordered NMR spectroscopy: analysis of methyllithium *J. Org. Chem.* **2013**, *78*, 11733–11746.

(8) Yadav, J. S.; Thirupathaiah, B.; Srihari, P. A concise stereoselective total synthesis of (+)-Artemisinin *Tetrahedron* **2010**, *66*, 2005–2009.

(9) Horton, M.; Pattenden, G. Bicyclo[3.3.0]octenones in synthesis. A new synthesis of (±)-cedrene using sequential inter- and intra-molecular Michael reactions *J. Chem. Soc. Perkin Trans. 1* **1984**, *0*, 811–817.

(10) Ngo, K.; Brown, G. D. Synthesis of sesquiterpene allylic alcohols and sesquiterpene dienes from *Cupressus bakeri* and *Chamaecyparis obtusa J. Chem. Soc. Perkin Trans. 1* **2000**, *0*, 189–194.

(11) Fukuzawa, A.; Aye, M.; Takaya, Y.; Masamune, T.; Murai, A. A sesquiterpene alcohol from the red alga *Laurencia nipponica Phytochemistr*y **1990**, *7*, 2337–2339.

(12) Dauben, W. G.; Hubbell, J. P.; Oberhansli, P. & Thiessen, W. E. Acid-catalyzed cyclization of cembrene and isocembrol. *J. Org. Chem.* **1979**, *44*, 669–673.

(13) Kabsch, W. *Acta Cryst.* **2010**, *D66*, 125–132.

(14) Kabsch, W. *Acta Cryst.* **2010,** *D66*, 133–144.

(15) Hattne, J., *et al. Acta Cryst.* ***2015***, *71*, 353–360.

(16) Sheldrick, G. M. A short history of SHELX. *Acta Cryst.* **2008**, *A64*, 112–122.

(17) Sheldrick, G. M. *Acta Cryst.* **2015** *A71*, 3–8.

(18) Sheldrick, G. M. *Acta Cryst.* **2015**, *C71*, 3–8.

(19) Hübschle, C. B., Sheldrick, G. M. & Dittrich, B. *J. Appl. Cryst.* **2011,** *44*, 1281–1284.

CHAPTER THREE

Small Molecule Structural Determination Utilizing Microcrystal Electron Diffraction

Jessica E. Burch, Austin Smith, Seb Caille, Shawn D. Walker, Ryan Wurz, Victor Cee, Jose Rodriguez, D. Gostovic, Kyle Quasdorf, Hosea M. Nelson *In Preparation.*

Brian J. Curtis, Lee Joon Kim, Chester J. J. Wrobel, James M. Eagan, Rubin A. Smith, Jessica E. Burch, Henry H. Le, Alexander B. Artyukhin, Hosea M. Nelson, Frank C. Schroeder *Org. Lett.* **2020**, *22*, 6724–6728.

3.1 Abstract

The examples and work outlined in this chapter showcase the application of microcrystal electron diffraction (microED) towards pharmaceutical and natural product compounds, demonstrating its potential as a routine technique for structural analysis of small molecules. Fifteen small molecule structures were solved from compounds within active pharmaceutical pipelines, often without the need for cryogenic data collection or crystallization attempts. In addition, the structure of secondary metabolites from *Caenorhabditis elegans* nematodes were determined utilizing a combination of microED and synthesis.

3.2 Introduction

Structural analysis is a fundamental part of rationalizing chemical behavior and developing hypotheses that fuel innovation in the chemical sciences. Many of the commonly employed methods to elucidate the structure of small molecules, such as infrared spectroscopy, UV-vis, and NMR spectroscopy, rely on inference.[1] While a combination of these spectroscopic techniques can be sufficient to confidently assign structure, there are cases in the literature of well-studied natural products, and even pharmaceutical compounds in clinical trials, that were

initially misassigned.[2,3] This can lead to challenges with intellectual property and years of research effort spent synthesizing a different compound than intended.

Methods to provide unambiguous structural solutions are limited. X-ray crystallography has long been regarded as the gold standard for structural elucidation but is limited in practice by stringent crystal quality and size requirements, as described in detail in Chapter 1.[4] For samples that fail to produce crystals of sufficient quality for single crystal X-ray analysis, X-ray powder diffraction and NMR crystallography have been developed as alternative analytical techniques. These techniques may remain ineffective at providing assignments in the case of highly complex compounds.[5–10] X-ray free-electron laser (XFEL) crystallography can resolve complex species, but with only a handful of XFEL facilities in existence, its use for routine data analysis is impractical.[11] The complexity of chemical pipelines from modern pharmaceutical programs are increasing (Figure 3.1), making the development of novel analytical methods suitable for

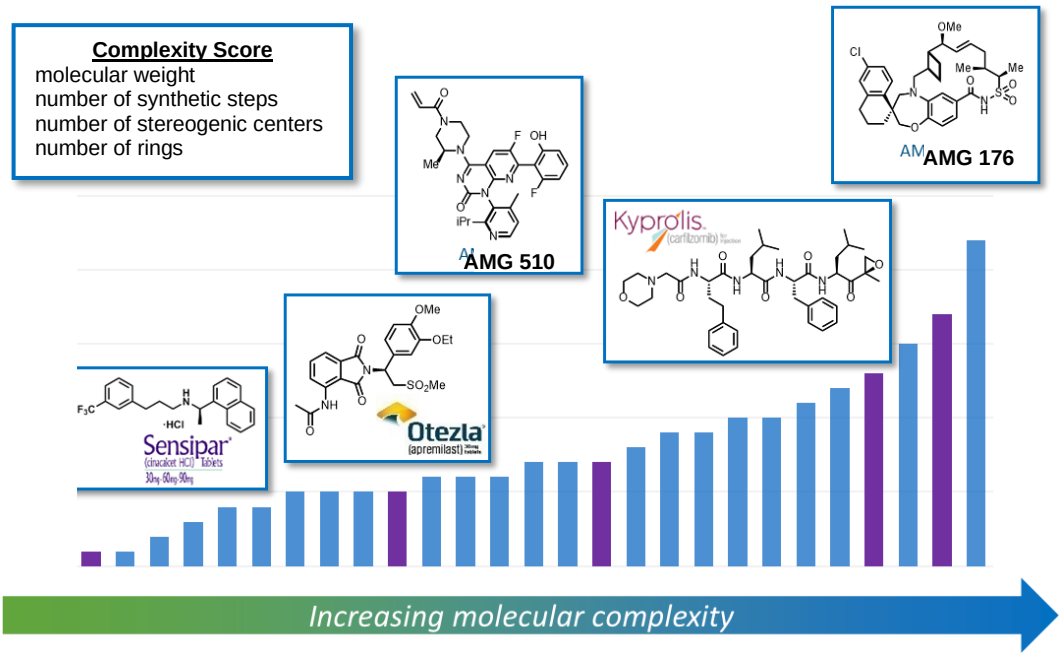


Figure 3.1 Molecular complexity score ranking the pipeline of Amgen's small molecule pharmaceutical programs.

providing routine, unambiguous structural assignment from challenging molecules critical for the future of drug development.[12]

Microcrystal electron diffraction (microED) has recently increased in popularity as a crystallographic technique capable of structural determination utilizing microcrystals that may be present in even seemingly amorphous powders.[13] A number of reports have compared structural solutions of identical species obtained by both single crystal X-ray crystallography and microED, demonstrating the ability for microED to provide data equivalent in accuracy to the current gold standard in solid state structural assignment.[13] MicroED has been used to elucidate structures spanning a wide range of chemical space, from proteins, materials, and MOFs, to small organic molecules, complex natural products, and organometallic species.[14–20] Furthermore, recent work determining the absolute configuration of a nanocrystalline pharmaceutical compound suggests a promising future in which microED can be utilized to determine absolute stereochemistry.[21] In addition to its promise as a novel, routine crystallographic technique, microED is capable of detecting polymorphology and crystalline impurities on the nanomolar scale.[22] This sensitivity is particularly attractive for industrial applications, where detection of minor impurities and crystal polymorphology could offer commercial value.[23]

3.3 MicroED of Pharmaceutical Compounds

The purpose of this study was to evaluate the practicality of using microED as a tool to routinely determine chemical structures of pharmaceutically relevant compounds. Thirty pharmaceutical samples were selected for evaluation by microED, spanning a range of chemical complexity from both medicinal and process pharmaceutical sectors. A subset of samples were isolated directly following routine chromatographic purification with no attempts to promote crystallinity, while another subset were isolated through purification by recrystallization.

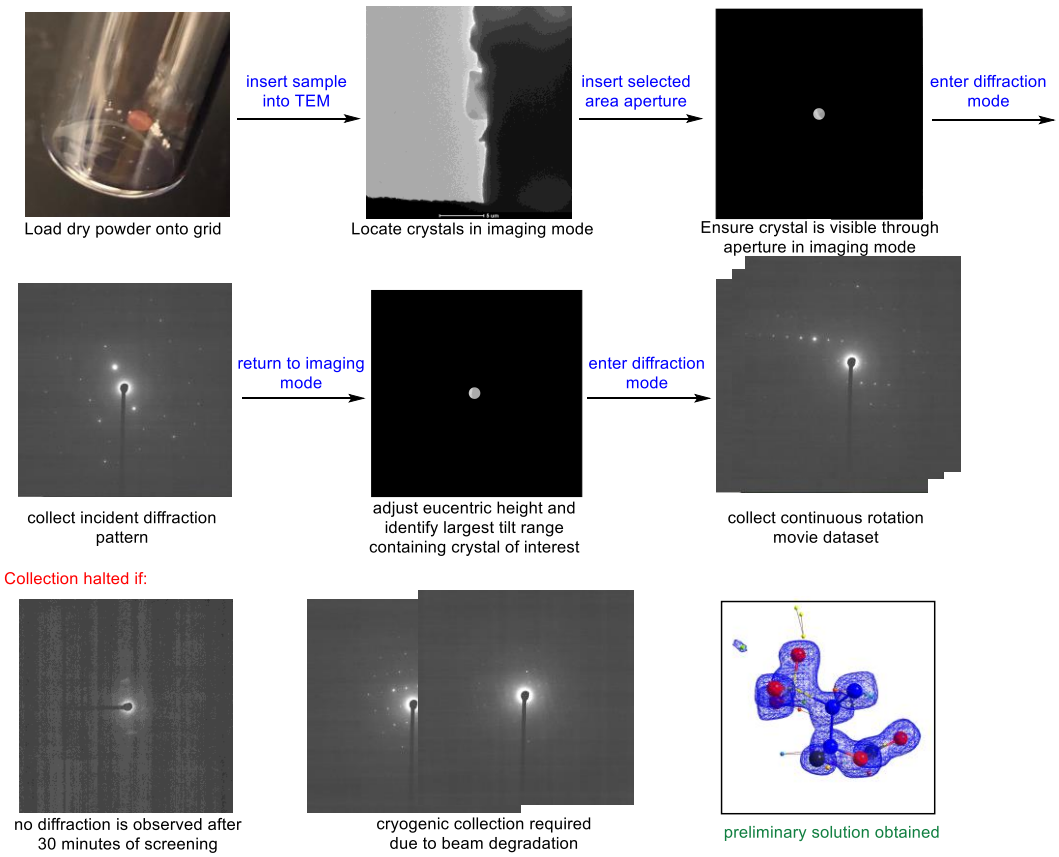


Figure 3.2 Representative MicroED data collection workflow.

Initial efforts began by rapidly screening all thirty samples at room temperature to evaluate crystallinity and collect diffraction datasets during the initial screening, if possible. Samples were prepared by transferring milligram quantities of dry powder as received into a dram vial and manually grinding with a glass pipette (Figure 3.2). A pure carbon 200 mesh Cu grid was dropped into the vial and shaken together with the sample. The grid was removed and gently tapped while held with tweezers to remove excess powder. The sample was placed onto a single tilt holder and inserted into a Thermo Fisher Scientific Talos F200C transmission electron microscope equipped with a Ceta-D detector operating at an accelerating voltage of 200 keV.

During this initial screening period, active microscope time was limited to one hour or less per sample. To screen for crystallinity, particles were located on the grid in imaging mode at 2600x

magnification. After identifying a particle of interest, a diffraction pattern was recorded by isolating a region of the particle using a selected area aperture and entering parallel-illuminated diffraction mode utilizing the low dose software on the Thermo Fisher microscope user interface.

A single image of the diffraction pattern was taken on a Thermo Fisher Scientific Ceta-D camera. If user inspection of the diffraction pattern suggested that the particle was mono-crystalline and provided <1.2 Å resolution diffraction, the microscope was returned to imaging mode. The eucentric height, defined to be the point at which the sample does not move laterally as the crystal is rotated, is achieved through finely adjusting the stage height to ensure the crystal would remain within the selected area aperture throughout a tilt series. The maximum tilt range is ± 65°, but neighboring crystals and grid bars might prevent collection of a full 130° dataset. After making these adjustments and returning to diffraction mode, a continuously rotating electron diffraction movie was collected by rotating the stage at a rate of 0.3° s^{-1}. The Ceta-D CMOS 4k x 4k camera was operated using rolling shutter mode and continuously integrated at a rate of 3 seconds per frame with data binning by 2 to produce 2k x 2k images. Diffraction movies were saved as SER files.

Given the ability to collect multiple movies in a matter of minutes, we wanted the ability for a single user to simultaneously collect and process data. To facilitate this, we developed a Python script that interacts with existing programs utilized for processing microED data to allow for automated conversion and indexing, inspired by similar automation developed for serial rotation electron diffraction.[24–27] Every movie collected for this study was successfully indexed and scaled using this strategy, taking approximately 1-2 minutes to process each dataset. The development and use of automated data processing pipelines for microED is discussed in detail in Chapter 4.

Once one or more processed movies are obtained, a user can either directly solve the data by using the SHELX software suite, or quickly merge multiple pre-processed datasets using XSCALE before submitting to SHELX. [28–30] By following this screening method, six of the thirty compounds produced *ab initio* preliminary solutions in an hour or less per sample (Figure 3.3). After obtaining these preliminary solutions from SHELXT or SHELXD, the data was refined using SHELXL within ShelXle.[31]

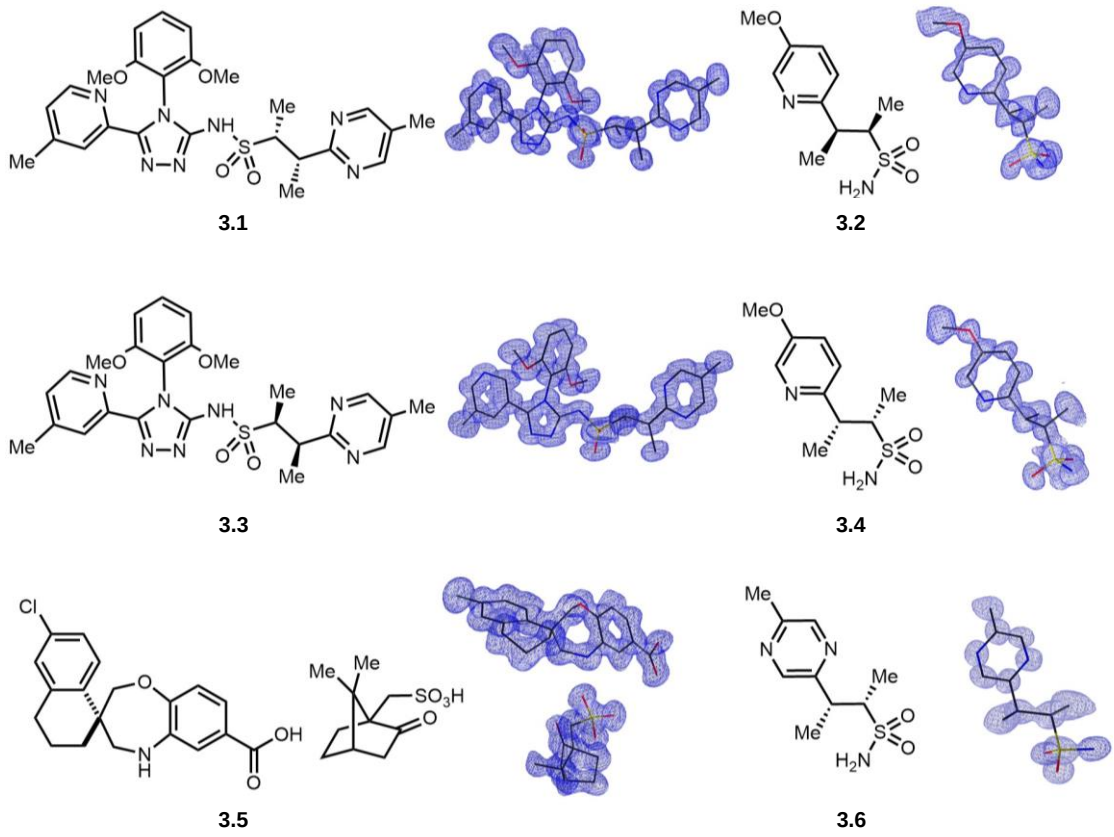


Figure 3.3 Structures in which a preliminary microED solution was obtained in under one hour each. Structures 3.1 and 3.3 have two molecules within their asymmetric unit. Hydrogens omitted for clarity.

Nine datasets were recorded for enantiomeric pair **3.1** and **3.3** (Figure 3.3), and for each compound two to three of these datasets were merged to provide the structural solutions. Both isomers crystallized in the monoclinic space group $P2_1$ with similar unit cell parameters. The

asymmetric unit of each enantiomer contained two molecules, and overlaying these molecules using PyMol software demonstrated conformational differences that break their crystallographic equivalency.[32] Enantiomeric pair **3.2** and **3.4** were also obtained during the rapid screening stage, crystallizing in orthorhombic space group $P2_12_12$ with nearly identical unit cell parameters. Three to four datasets were collected for each enantiomer, but the structure of each enantiomer was resolved using a single dataset. It is not trivial to assign relative stereochemistry by 2D NMR for many of the compounds in Figure 3.3, in which stereogenic centers residing on acyclic portions of the molecule can be difficult to assign due to free rotation.

Co-crystallized species **3.5** was obtained in space group $P2_1$ after collecting five datasets and merging four for the final solution. For this analyte, considerable rotational disorder was observed at room temperature for the camphor sulfonic acid (CSA) moiety. This structure assigns connectivity of the spirocyclic compound, but the CSA rotational disorder prevents unambiguous assignment of stereochemistry. Interestingly, the enantiomer of this compound was also studied, but required cryogenic data collection due to its beam sensitivity.[33] Compound **3.6** was analyzed after purification by crystallization, while its enantiomer, (**3.12**, Figure 3.8), was studied as a powder isolated without any crystallization attempts. Only one dataset was collected to generate a single movie solution of crystallized enantiomer **3.6** (Figure 3.3) in $P2_12_12_1$. In contrast, non-crystallized enantiomer **3.12** (Figure 3.8) required multiple hours of screening to locate monocrystalline domains.

A subset of samples subjected to the initial screening at room temperature provided high resolution incident diffraction, but rapid truncation of diffraction resolution was observed over the duration of the continuous rotation movie, presumably due to radiation damage (Figure 3.4). Depending on the severity of the beam sensitivity, high quality diffraction data can often still be

obtained, but may require extensive data collection and trial-and-error merging of datasets until an optimal combination is achieved. This can be a time- and labor-intensive process. As an alternative strategy, samples can be cooled to cryogenic temperatures within the microscope to reduce beam sensitivity (Figure 3.4a–3.4b vs. 3.4c–3.4d). Cryogenic screening was avoided except in the case of these highly beam sensitive samples due to the time required to cool and subsequently warm the cryogenic specimen holder between samples.

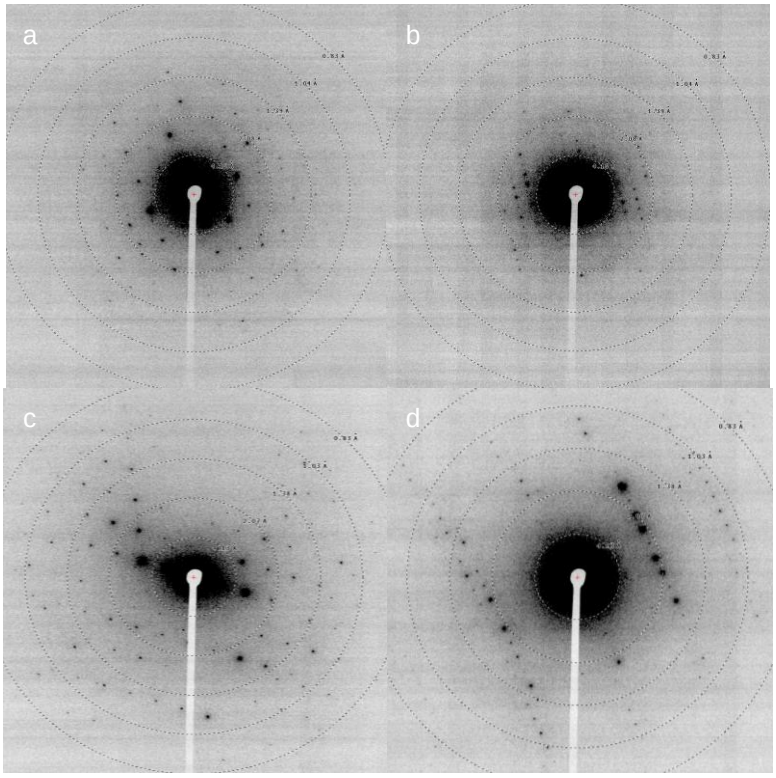


Figure 3.4 Diffraction resolution loss. Room temperature TEM diffraction movie frame 1 (a) and frame 50 (b) of **3.9** vs. cryogenic diffraction movie frame 1 (c) and frame 50 (d) of **3.9**, highlighting attenuation of resolution loss with reduced temperature.

To prepare the next batch of samples, new dry powder grids were prepared as described previously. These grids were placed onto a 626 Gatan cryo holder and inserted into a Talos F200C TEM at room temperature. After insertion but before opening the column valves, the cryo holder

was cooled with liquid nitrogen to -177 °C. The temperature was monitored using a Gatan 1905 Temperature Controller. After stabilizing at cryogenic temperatures, particles were again screened using selected area diffraction. An additional four compounds (**3.7**–**3.10**, Figure 3.5) were solved by this method, with a maximum time of three hours from powder to preliminary structure per sample. It is important to note that these cryogenically cooled samples did not require time-intensive vitrification and cryo transfer processes, still making this strategy a convenient method for rapid structural elucidation.

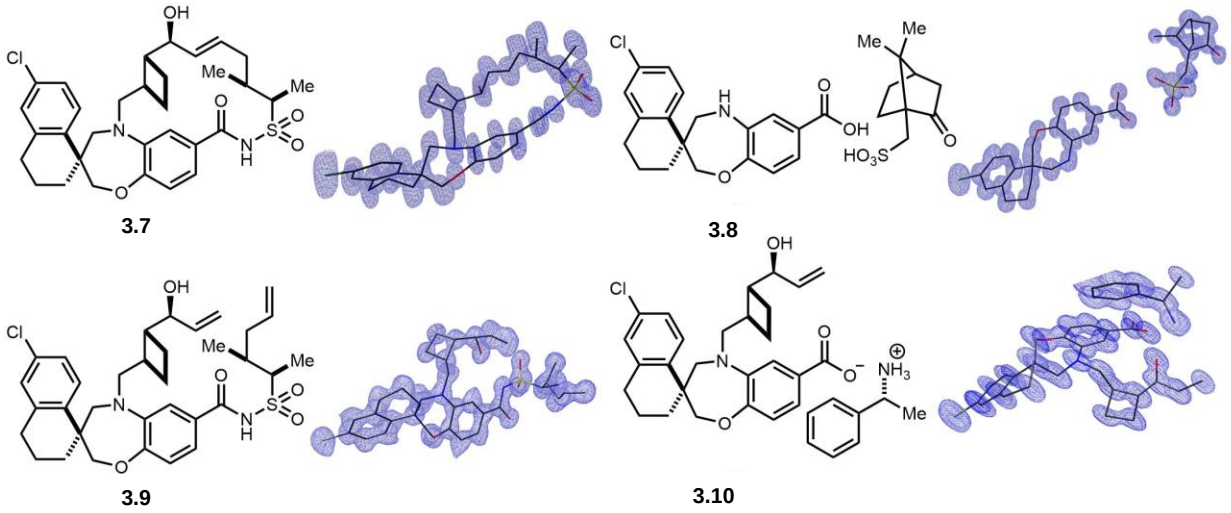


Figure 3.5 Small molecule structures solved by microED at cryogenic temperatures in under three hours each. Hydrogens omitted for clarity. Crystals of species **3.7** had two molecules in their asymmetric unit.

Samples that initially looked promising by incident diffraction, but were unable to provide sufficient data at room temperature, had a marked improvement in diffraction resolution at cryogenic temperatures. Minor variations in the unit cell axes could be observed between room temperature data and cryogenic data, but no phase transitions were observed. Remarkably, the structure of compound **3.7** (Figure 3.5) containing a 16-membered ring, was able to be solved by SHELXD in P_1 after merging three out of four generated datasets. While salt species **3.8** did not

provide a solution at room temperature, collection of four datasets (and merging of three) at cryogenic temperatures provided a solution in $P2_1$ with less rotational disorder at the camphor sulfonic acid molecule than its enantiomeric counterpart, **3.5** (Figure 3.3). To resolve ambiguity of the disordered CSA, **3.5** was subjected to additional cryogenic screening. Collection of four and merging of two datasets provided a solution with reduced disorder. It is critical to note that the cryogenic structure of **3.5** led to unambiguous assignment of the stereochemistry at the CSA, producing the expected enantiomeric structure to **3.8**; in contrast, relying upon the disordered CSA in the room temperature structure would lead to misassignment of stereochemistry of the spirocycle. The error values for these two structures are similar, with R = 0.1198, GooF = 1.239 for the room temperature structure, and R = 0.1127, GooF = 1.143 for the cryogenic structure. The primary differences between these structures are seen in the preliminary solution, obtained with no human input other than molecular formula (Figure 3.6). In the preliminary structures, the cryogenic structure is missing two atoms in the CSA; the room temperature preliminary solution is missing five. Importantly, the missing atoms in the room temperature structure include those relevant to assigning stereochemistry.

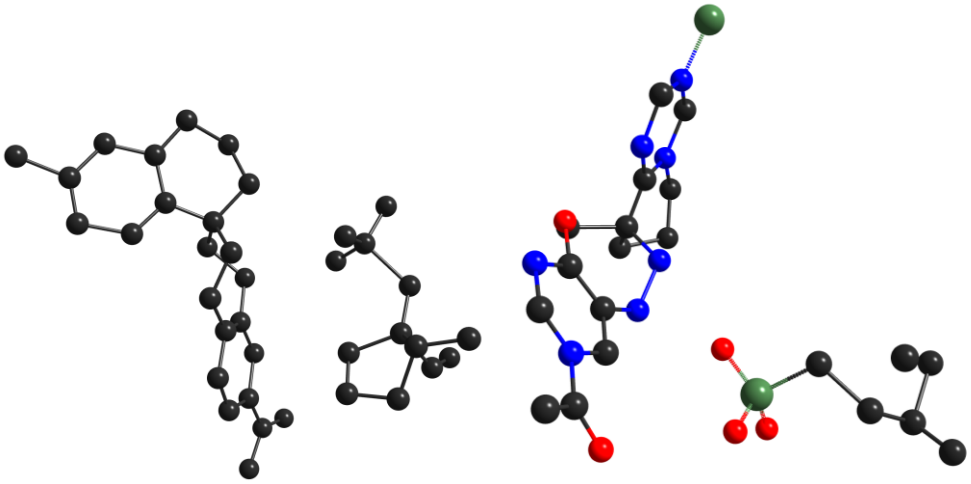


Figure 3.6 Preliminary solutions of **3.5** from cryogenic collection (left) and room temperature collection (right).

The other difference between these structures is seen in the thermal ellipsoid representations (Figure 3.7). While the spirocycle is well-resolved in both structures, the thermal ellipsoids of the CSA are drastically reduced with the cryogenic temperatures. We believe this serves as an excellent example as to why it is critical for microED reports to not only include standard crystallographic statistics, but preliminary structural solutions and refined structure files.

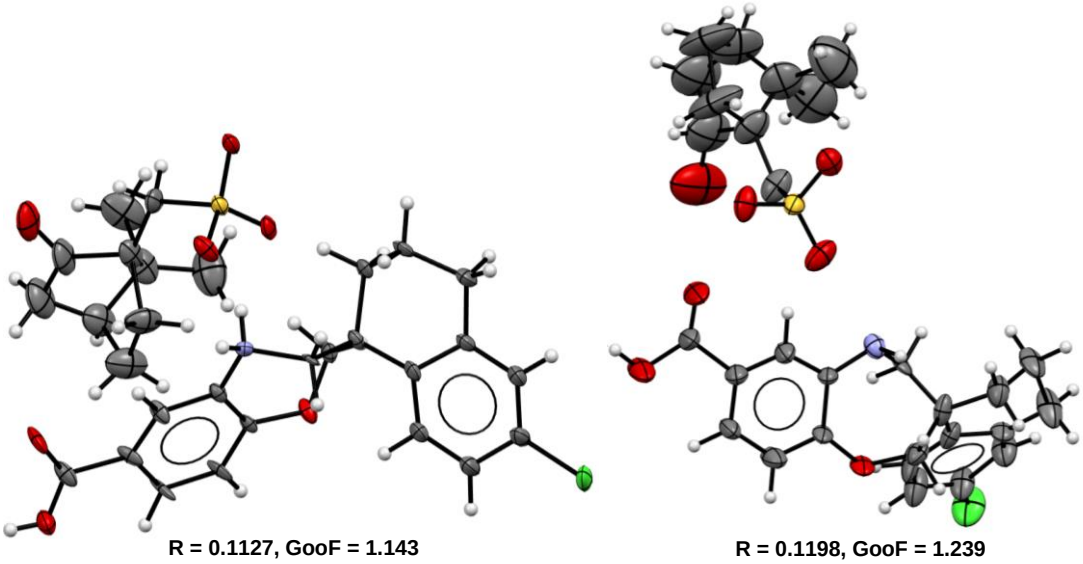


Figure 3.7 Refined structures of **3.5** from cryogenic collection (left) and room temperature collection (right). Thermal ellipsoids shown at 30% probability.

Diene **3.9** (Figure 3.5) yielded a preliminary solution in $P2_12_12_1$ after generating and merging of three datasets. Amine salt **3.10** required collection of seven and merging of three movies to generate a solution in space group $P2_12_12_1$. Each of these samples produced a preliminary structure in three hours or less.

The remaining eighteen samples (**3.11**–**3.15** Figure 3.8, and Figure 3.9) were either too polycrystalline to easily obtain a structure, had poor diffraction resolution, or provided no diffraction at room temperature. In the case of polycrystalline compound **3.12** (Figure 3.8), additional hours spent carefully locating monocrystalline domains within a largely polycrystalline

sample ultimately provided the structure at room temperature. From eight datasets collected, two were merged to provide a solution in space group C2. Other low resolution and polycrystalline samples were subjected to additional screening, but ultimately crystallization attempts were required to provide their solutions.

To crystallize, ~1 mg of powder was placed into 6 x 50 mm borosilicate culture tubes purchased from VWR. Samples were dissolved in approximately 500 uL of solvent and allowed to slowly evaporate at room temperature. Higher boiling solvents were evaporated from open containers, while low boiling solvents required placing the culture tube inside an empty dram vial with a slightly loosened cap. If the initial solvent failed to produce crystals after fully evaporating, the amorphous samples were re-dissolved in the same culture tube with a new solvent mixture. Evaporation occurred until precipitation was observed. Sample crystallization time spanned from overnight to 3 days. Compound **3.14** was crystallized from diethyl ether. The crystals were placed onto a grid as a dry powder, plunge frozen in liquid nitrogen, and transferred into the TEM while the holder was maintained at cryogenic temperatures due to the presumed volatility of the sample. Microcrystals of enantiomeric pair **3.11** and **3.13** were obtained from slow evaporation from a 50/50 mixture of MeCN and H_2O and dried under reduced pressure. Microcrystals of sample **3.15** were generated by slow evaporation from a mixture of H_2O and DMSO (10% v/v). The crystals were blotted with a kimwipe and dried under reduced pressure to remove excess solvent before being inserted into the TEM.

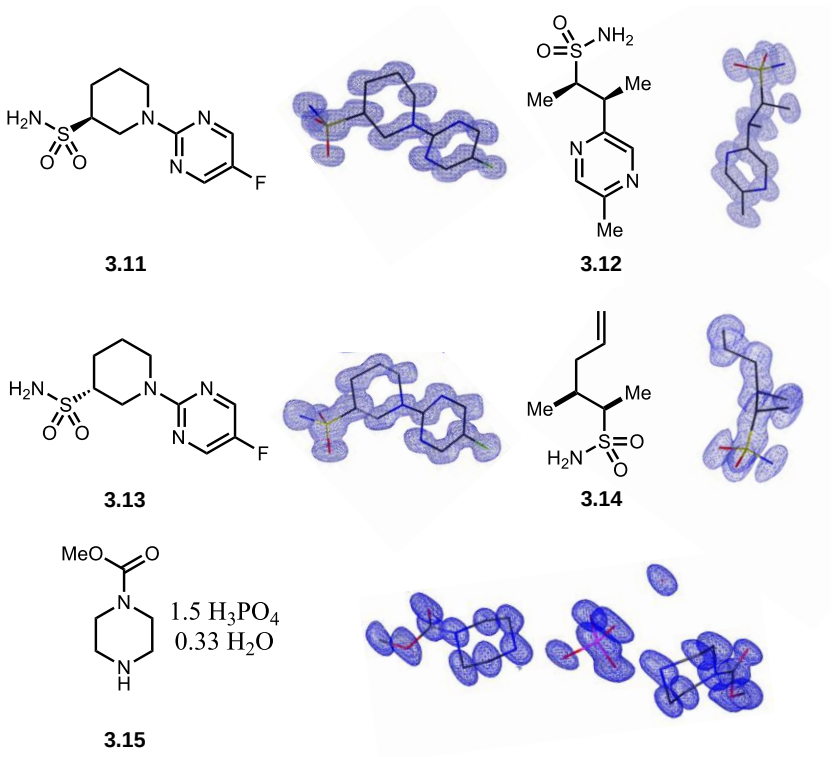


Figure 3.8 Structures obtained via microED with extensive screening and recrystallization. **3.12** Obtained by locating monocrystalline species on a largely polycrystalline grid. **3.11**, **3.13**–**3.15** Samples recrystallized to obtain structure. **3.11** and **3.13** contain four molecules within their asymmetric unit. **3.15** partial view of asymmetric unit. Hydrogens omitted for clarity.

Eleven structures were obtained directly from the samples as received, four of those compounds having been isolated without any attempts to crystallize during purification. The structural elucidation of four additional compounds were achieved through recrystallization screening. For example, sulfonamide pair **3.11** and **3.13** (Figure 3.8) were re-crystallized from MeCN/H_2O and provided solutions in $P2_12_12$. While screening recrystallized **3.13**, multiple similar unit cells were encountered, suggesting polymorphism. The structure of the presumed polymorph was not able to be determined and was not detected in the original powder sample.

Structure **3.15** is another example of a process sample that provided structural elucidation but was not necessarily representative of the original sample composition. While the presence of piperazine is maintained at levels lower than 5000 ppm, this impurity was found in significant amounts in the structure obtained from recrystallization. This highlights a limitation in gaining information about the bulk sample from recrystallized samples analyzed by microED.

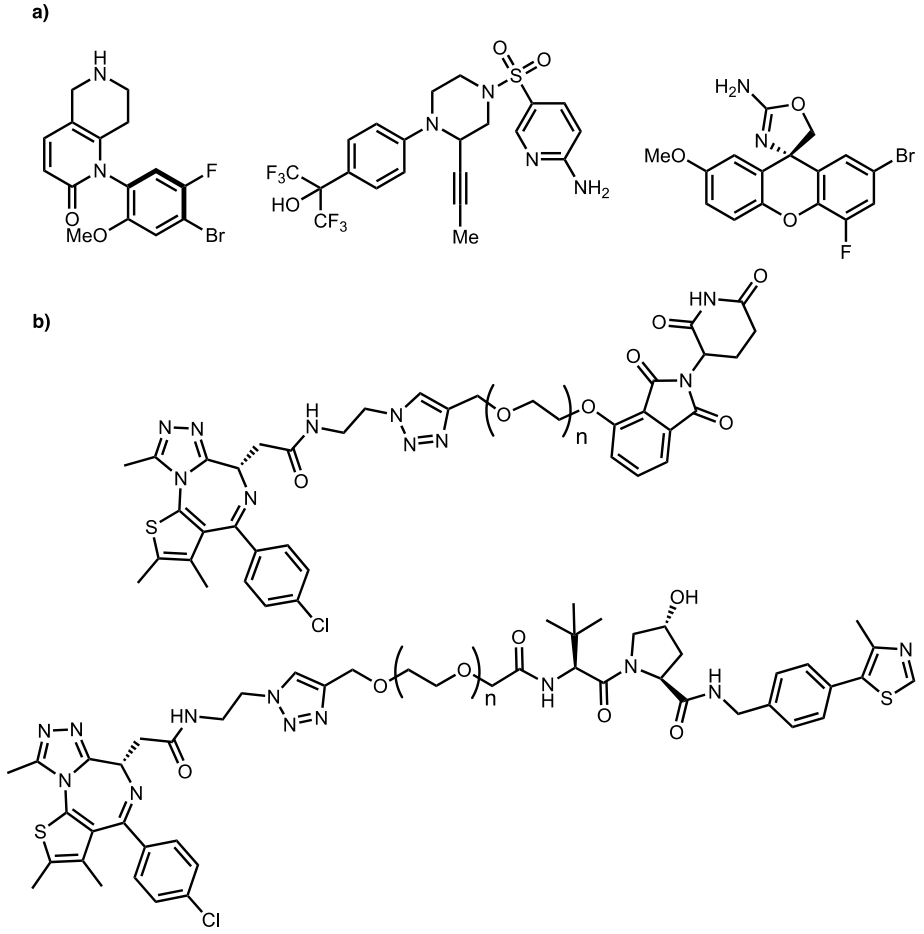


Figure 3.9 Samples that failed to generate structures. (a) Enantiomeric pairs (only one enantiomer drawn)

(b) PROTAC, n = 0–4

The remaining fifteen compounds failed to produce structures in our hands in a timely manner. These samples are comprised of three small molecule enantiomeric pairs (Figure 3.9a) and nine proteolysis targeting chimera (PROTAC) compounds (Figure 3.9b). While most medicinal and process small molecule samples can be evaluated as dry powders, complex species

like PROTACs are assumed to benefit from evaluation in the frozen hydrated state, as is described in more detail in Chapter 1. Extensive crystallization and vitrification screening is a potential avenue for producing microcrystals from these highly complex species, but was beyond the scope of this rapid-timeline study in which fifteen of thirty compounds were solved using approximately forty hours of TEM time and seventy hours of automated and user-driven data processing.

3.4 Pharmaceutical Atropisomerism

Atropisomerism is a type of axial chirality that arises from hindered rotation about a single bond (Figure 3.10). If the barrier to rotation is sufficiently high, typically above 20 kcal/mol, these conformers can be isolated separately.[34] There are examples of FDA-approved drugs and

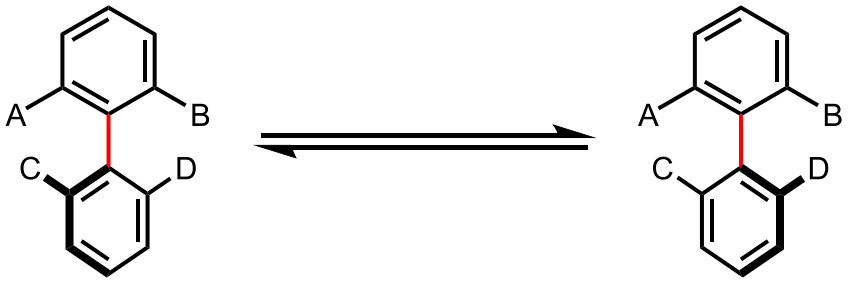


Figure 3.10 Schematic of atropisomerism.

experimental compounds that possess atropisomerism, and the differing conformations of these species have been shown to play an important role in their biological activity.[34,35]

One of these experimental atropisomeric compounds is GS-6207 (Figure 3.11) developed by Gilead Sciences. This compound is a potent, long-acting HIV-1 capsid inhibitor that possesses antiviral activity against both wild-type and multidrug-resistant HIV variants.[36,37] GS-6207 is currently undergoing clinical trials evaluating its use in combination with other antiretroviral agents for treatment of HIV.[38] Interestingly, this compound possesses two atropisomers, **3.16** and

3.17 that can readily interconvert in solution at room temperature; however, as a crystalline sodium salt, these species do not interconvert under ambient conditions.[36]

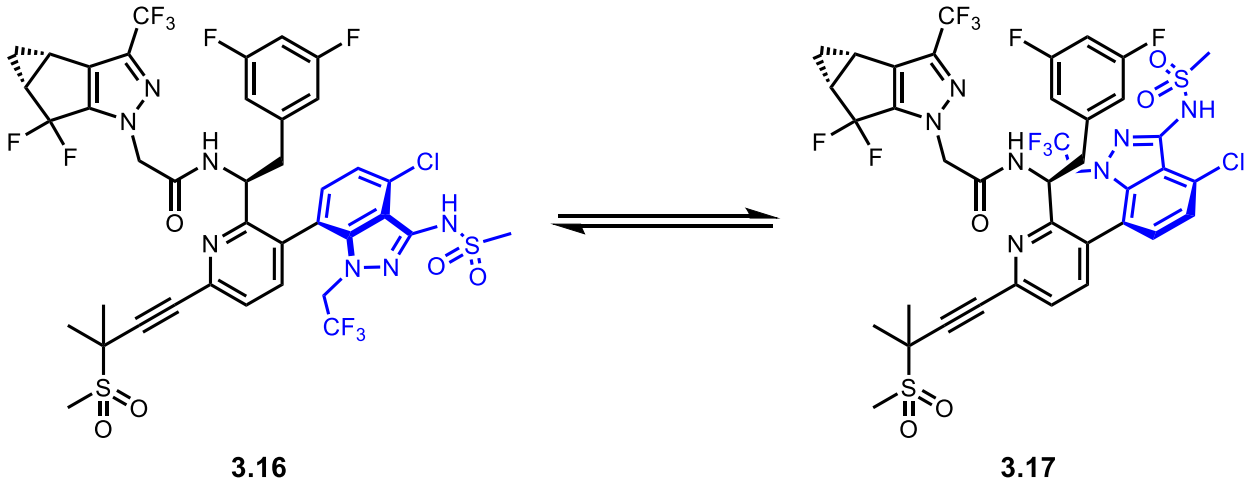


Figure 3.11 Atropisomers of **GS-6207.**

In this study, we sought to unambiguously confirm the identity of a single atropisomer of GS-6207 isolated as a sodium salt. Efforts to obtain a structure *via* single crystal X-ray crystallography were undertaken, but proved to be unsuccessful. MicroED has been demonstrated to be a powerful alternative to X-ray crystallography, particularly in cases where crystal size or morphology prevent structural elucidation by X-ray crystallography. Analysis of GS-6207 microcrystals utilizing microED led to unambiguous determination of atropisomer identity.

The sodium salt of GS-6207 was analyzed by loading dry powder onto TEM grid and plunging into liquid nitrogen. A preliminary solution was obtained by merging three datasets from three separate crystals with no user input other than molecular formula utilizing SHELXD.[24–26,28–31] This high resolution (0.95 Å) and high completeness (95.7%) structure in $P2_12_12_1$ unambiguously confirms the identity of the isolated atropisomer as the sodium salt of **3.17** (Figure 3.12). To confirm that this structure is representative of the bulk material, a simulated powder X-ray diffraction pattern was generated from the refined microED structure and overlayed with the

experimental XPRD of the bulk material.[39] The agreement between these PXRD patterns confirms identification of the bulk sample as sodium salt of **3.17**.

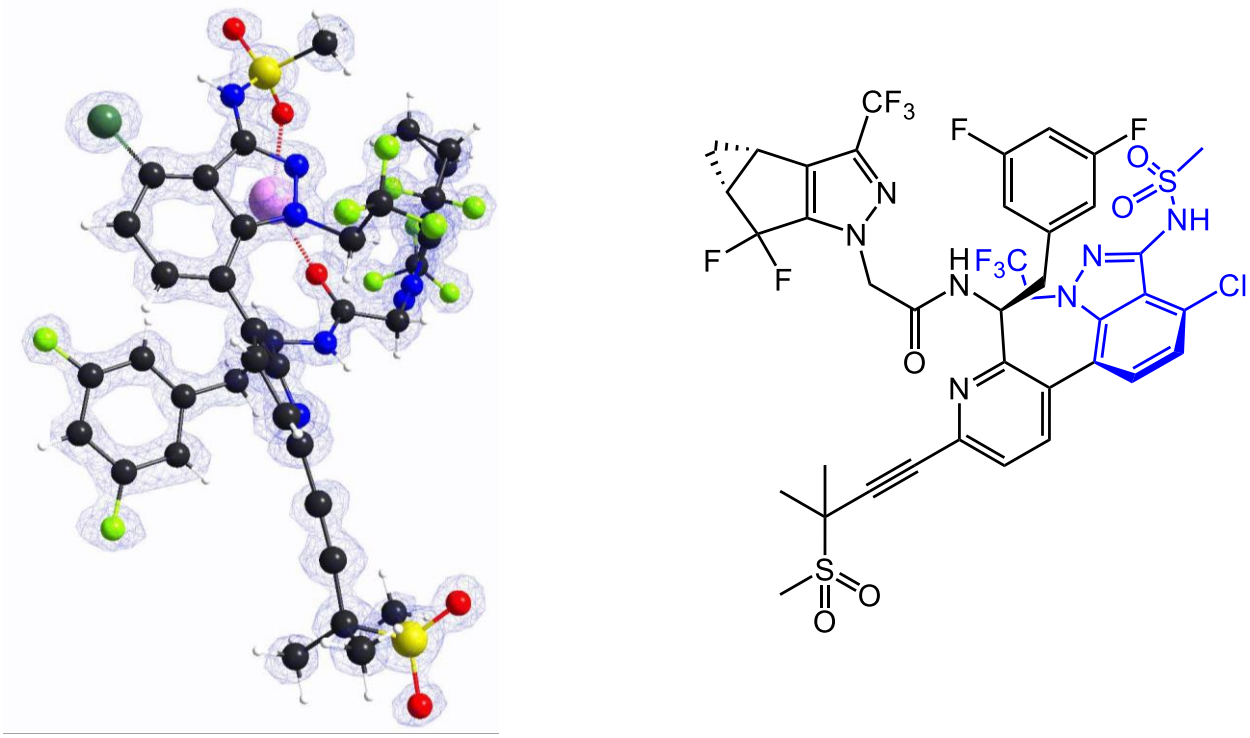


Figure 3.12 *Ab initio* microED solution of GS-6207 sodium salt (left), confirming atropisomer identity as **3.17** (right).

3.5 MicroED of Natural Products

The nematodes *Caenorhabditis elegans* and *Pristionchus pacificus* are simple but powerful model systems for human physiology and biochemistry.[40–43] Studies of these organisms have led to the discovery of an extensive network of small molecules that play a central role in the regulation of aging, behavior, and development.[44,45] These signaling molecules are biochemically derived from assembly of simple building blocks from all major primary metabolic pathways, including nucleoside metabolism, demonstrating a broad biological importance.

During metabolomic studies of mutant *C. elegans* and *P. pacificus*, the Schroeder group and others identified a series of unusual purine nucleoside and gluconucleosides metabolites (Figure 3.13).[40,44,45] Further studies revealed the presence of related gluconucleoside natural products

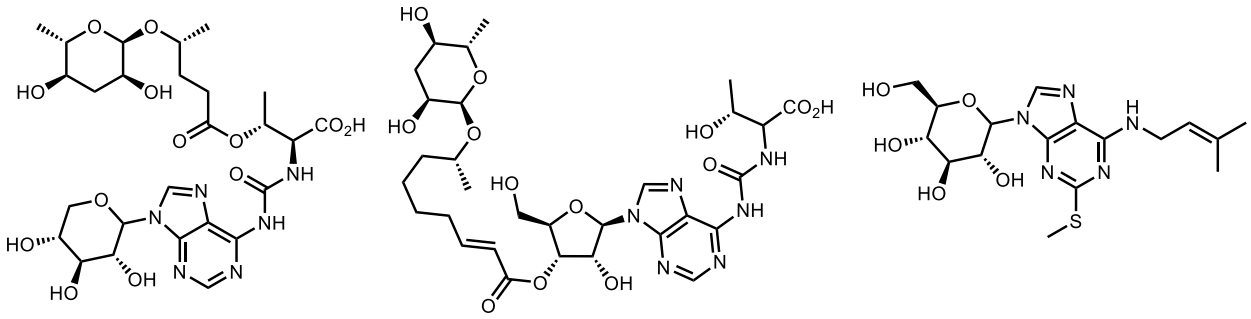


Figure 3.13 Unusual purine nucleoside and gluconucleosides from *C. elegans* and other nematodes.

that incorporate a uric acid moiety. This is notable due to the known biosynthetic generation of uric acid from purine degradation, while there is an underrepresentation of this motif in reported natural products isolated from animals.[46–50] Interestingly, uric acid supplementation was shown to increase the life span in *C. elegans*, and these long-lived mutants showed an increase in receptors related to insulin signaling and production of uric acid-derived gluconucleosides uglas#1 (**3.18**, Figure 3.14) and uglas#11 (**3.19**) relative to ascaroside ascr#1 (**3.20**).[51–53]

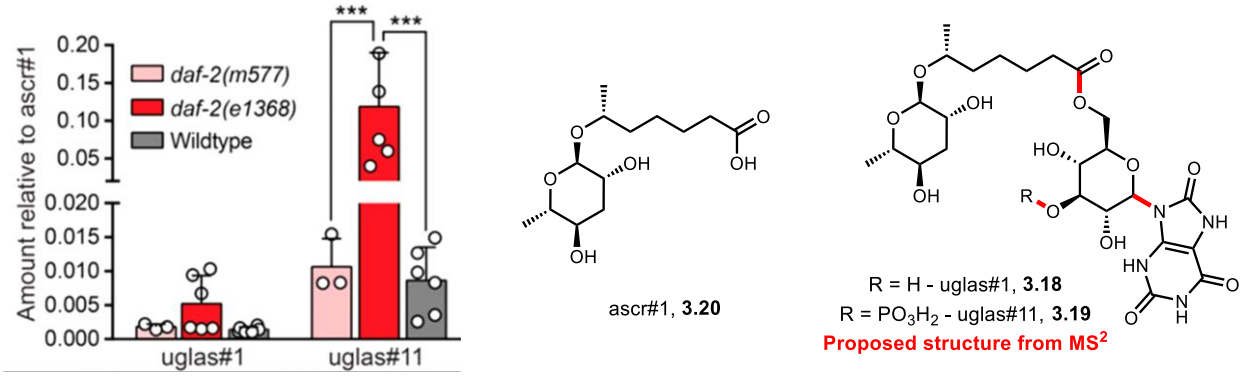


Figure 3.14 Upregulation of **3.18** and **3.19** in long-lived mutant *C. elegans* compared to **3.20** and their proposed structures from MS^2 studies.

The proposed structure of uglas#1 (**3.18**) from mass spectrometry studies includes uric acid, glucose, and an ascaroside moiety, but the connectivity of these components was unable to be determined due to insufficient quantities of isolable metabolite for studies by NMR.[54] To determine the connectivity of these species, the synthesis of uglas#1 was undertaken by our collaborators in the Schroeder laboratory. Glycosylation of uric acid (**3.21**, Figure 3.15) with protected glucose **3.22** produced a mixture of three constitutional isomers, **3.23**, **3.24**, and **3.25**. Deprotection and comparison of these products to *C. elegans* extract confirmed that the major synthetic glycosylation product, gluric#1 (**3.23**), was present as a metabolite in *C. elegans*. Unfortunately, 2D NMR was unable to assign N-linkage due to broadening of important HMBC correlations. To confirm the identity of this metabolite, we performed microED studies. MicroED has recently been demonstrated to be a powerful tool for small molecule structure elucidation, but had remained largely unproven in natural product applications. Synthetic gluric#1 (**3.23**), isolated

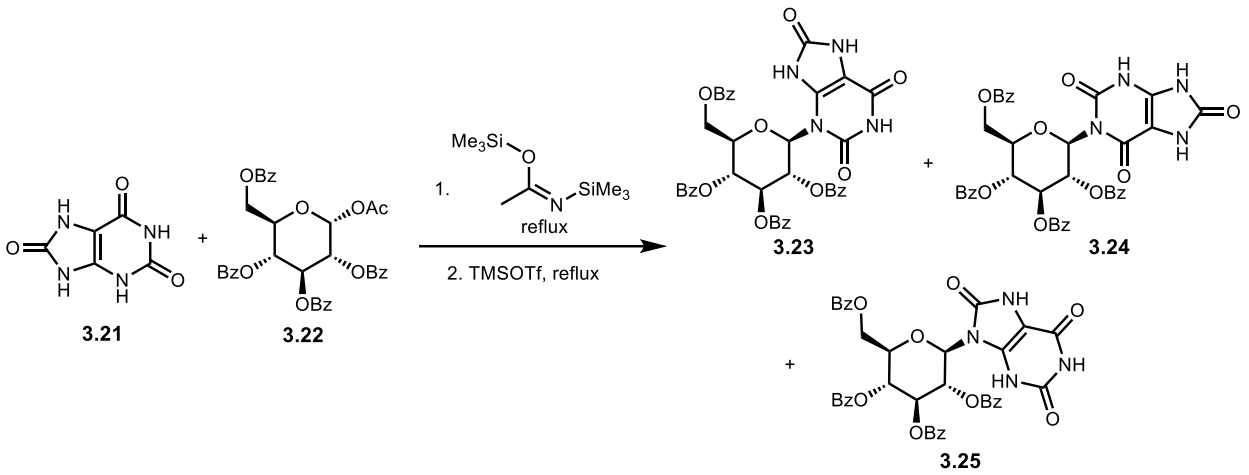


Figure 3.15 Synthesis of a mixture of glycosylated uric acid-derivatives *en route* to uglas#1.

by reversed-phase flash chromatography as a white powder, was deposited on a Quantifoil holey-carbon TEM grid. Initial electron micrographs of the samples showed numerous microcrystalline domains, including prisms 1−3 μm long. Continuous rotation selected area diffraction data were collected from 104 crystals. Merging of four data sets provided a direct methods solution from high resolution (1.0 Å), high-completeness (87%) data in P_1.

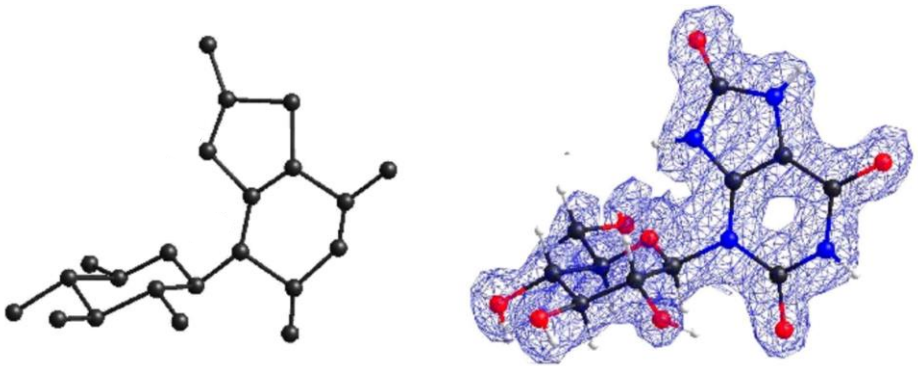


Figure 3.16 Preliminary solution (left) and refined structure (right) of gluric#1 **3.23**.

Importantly, without refinement or any user input besides molecular formula, the initial structural solution obtained from SHELXD confirms glycosylation regioselectivity, despite lacking some atoms on the sugar fragment (Figure 3.16).[28–31] Refinement of this structure

allowed for placement of all atoms of gluric#1 (**3.23**), providing for unambiguous confirmation of the structure.

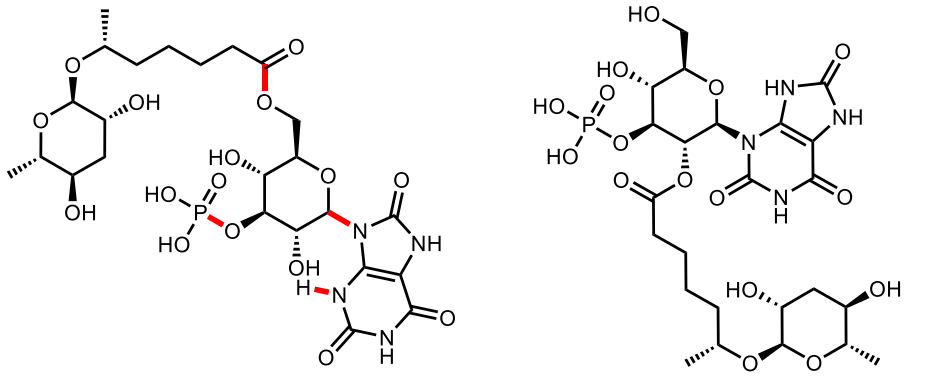


Figure 3.17 Proposed and revised structure of **3.19**.

Having resolved this structural ambiguity, the Schroeder group continued their synthetic effort and ultimately confirmed the revised structure of the uric acid-derived gluconucleoside uglas#11 (Figure 3.17), as is described in detail in our published work.[20]

3.6 Conclusion

In conclusion, the studies described in this chapter highlight the emerging technique of microED as a promising tool capable of routine unambiguous structural elucidation of small molecules comprising a range of chemical complexity. While study of biological samples *via* microED typically require time-consuming vitrification and cryo transfer processes, we demonstrated that many small molecule compounds provided diffraction data sufficient for obtaining preliminary solutions within minutes to a few hours from room temperature or non-solvated cryogenic conditions. Additionally, microED addresses challenges relevant to natural products isolation and total synthesis, in which complex molecules may not be able to be resolved confidently using NMR studies, and it is often not feasible to grow crystals large enough for single crystal X-ray crystallography.

3.7 Notes and References

(1) Günther, H. NMR Spectroscopy: Basic Principles, Concepts and Applications in Chemistry; *John Wiley & Sons: New York*, **2013**.

(2) Chhetri, B. K.; Lavoie, S.; Sweeney-Jones, A. M.; Kubanek, J. Recent trends in the structural revision of natural products. *Nat. Prod. Rep.* **2018**, *35* 514−531.

(3) Jacob, N. T.; Lockner, J. W.; Kravchenko, V. V.; Janda, K. D. Pharmacophore reassignment for induction of the immunosurveillance cytokine TRAIL. *Angew. Chem. Int. Ed.* **2014**, *53*, 6628−6631.

(4) Dunitz, J. D. X-ray Analysis and the Structure of Organic Molecules; *Verlag: Zürich*, **1995**.

(5) Florence, A.; Johnston, A.; Fernandes, P.; Shankland, K.; Stevens, H. N. E.; Osmundsen, S.; Mullen, A. B. Powder study of hydrochlorothiazide form II. *Acta Cryst., Sect. E: Struct. Rep. Online*. **2005**, *61*, 2798−2800.

(6) Harris, K. D. M. Structure solution from powder X-ray diffraction data by genetic algorithm techniques, applied to organic materials generated as polycrystalline products from solid state processes. *Mat. Man. Proc.* **2009**, *24*, 293−302.

(7) Miclaus, M.; Grosu, I. G.; Filip, X.; Tripon, C.; Filip, C. Optimizing structure determination from powders of crystalline organic solids with high molecular flexibility: the case of lisinopril dehydrate. *CrystEngComm*. **2014**, *16*, 299−303.

(8) Luedeker, D.; Gossmann, R.; Langer, K.; Brunklaus, G. Crystal Engineering of Pharmaceutical Co-crystals: “NMR Crystallography” of Niclosamide Co-crystals. *Cryst. Growth Des.* **2016**, *16*, 3087−3100.

(9) Hope, M. A.; Nakamura, T.; Ahlawat, P.; Mishra, A.; Cordova, M.; Jahanbakhshi, F.; Mladenović, M.; Runjhun, R.; Merten, L.; Hinderhofer, A.; Carlsen, B. I.; Kubicki, D. J.;

Gershoni-Poranne, R.; Schneeberger, T.; Carbone, L. C.; Liu, Y.; Zakeeruddin, S. M.; Lewinski, J.; Hagfeldt, A.; Schreiber, F.; Rothlisberger, U.; Grätzel, M.; Milić, J. V.; Emsley, L. Nanoscale phase segregation in supramolecular p-templating for hyrbide perovskite photovoltaics from NMR crystallography. *J. Am. Chem. Soc.* **2021**, *143*, 1529−1538.

(10) David, W. I. F.; Shankland, K. Structure determination from powder diffraction data. *Acta Crystallogr., Sect. A: Found. Crystallogr.* **2008**, *64*, 52−64.

(11) Kao, C.-C. Challenges and opportunities for the next decade of XFELs. *Nat. Rev. Phys.* **2020**, *2*, 340−341.

(12) Caille, S.; Cui, S.; Faul, M. M.; Mennen, S. M.; Tedrow, J. S.; Walker, S. D. Molecular complexity as a driver for chemical process innovation in the pharmaceutical industry. *J. Org. Chem.* **2019**, *84*, 4583–4603.

(13) (a) Gemmi, M.; Mugnaioli, E.; Gorelik, T. E.; Kolb, U.; Palatinus, L.; Boullay, P.; Hovmöller, S.; Abrahams, J. P. 3D Electron Diffraction: The Nanocrystallography Revolution. *ACS Cent. Sci.* **2019**, *5*, 1315–1329. (b) Jones, C. G.; Martynowycz, M. W.; Hattne, J.; Fulton, T. J.; Stoltz, B. M.; Rodriguez, J. A.; Nelson, H. M.; Gonen, T. The CryoEM Method MicroED as a Powerful Tool for Small Molecule Structure Determination. *ACS Cent. Sci.* **2018**, *4*, 1587–1592.

(14) Mugnaioli, E.; Lanza, A. E.; Bortolozzi, G.; Righi, L.; Merlini, M.; Cappello, V.; Marini, L.; Athanassiou, A.; Gemmi, M. Electron Diffraction on Flash-Frozen Cowlesite Reveals the Structure of the First Two-Dimensional Natural Zeolite. *ACS Cent. Sci.* **2020**, *6*, 1578–1586.

(15) Clabbers, M. T. B.; Hongyi, X. Microcrystal electron diffraction in macromolecular and pharmaceutical structure determination. *Drug Discovery Today: Technologies*. **2020**, *In Press*.

(16) Jones, C. G.; Asay, M.; Kim, L. J.; Kleinsasser, J. F.; Saha, A.; Fulton, T. J.; Berkley, K. R.; Cascio, D.; Malyutin, A. G.; Conley, M. P.; Stoltz, B. M.; Lavallo, V.; Rodríguez, J. A.; Nelson,

H. M. Characterization of Reactive Organometallic Species via MicroED. *ACS Centr. Sci.* **2019**, *5*, 1507–1513.

(17) Das, P. P.; Perez, A. G.; Galanis, A. S.; Nicolopoulos, S. Structural Characterization of Beam Sensitive Pharmaceutical Compounds Using 3D Electron Diffraction-Micro-ED at Low Dose with Pixelated Detectors. *Microscopy and Microanalysis*. **2020**, *26*, 1522–1522.

(18) Kim, L. J.; Xue, M.; Li, X.; Xu, Z.; Paulson, E.; Mercado, B. Q.; Nelson, H. M.; Herzon, S. Structural Revision of the Lomaiviticins *J. Am. Chem. Soc.* **2021**, *143*, 6578–6585.

(19) Samkian, A.; Kiel, G. R.; Jones, C. G.; Bergman, H.; Oktawiec, J.; Nelson, H. M.; Tilley, T. D. Elucidation of Diverse Solid-State Packing in a Family of Electron-Deficient Expanded Helicenes via Microcrystal Electron Diffraction (MicroED) *Angew. Chem. Int. Ed.* **2020**, *5*, 2493–2499.

(20) Curtis, B. J.; Kim, L. J.; Wrobel, C. J. J.; Eagen, J. M.; Smith, R. A.; Burch, J. E.; Le, H. H.; Artyukhin, A. B.; Nelson, H. M.; Schroeder, F. C. Identification of Uric Acid Gluconucleoside–Ascaroside Conjugates in Caenorhabditis elegans by Combining Synthesis and MicroED *Org. Lett.* **2020**, *22*, 6724–6728.

(21) Brázda, P.; Palatinus, L.; Babor, M. Electron diffraction determines molecular absolute configuration in a pharmaceutical nanocrystal. *Science*. **2019**, *364*, 667–669.

(22) Kim, L. J.; Ohashi, M.; Zhang, Z.; Tan, D.; Asay, M.; Cascio, D.; Rodriguez, J. A.; Tang, Y.; Nelson, H. M. Prospecting for natural products by genome mining and microcrystal electron diffraction. *Nat. Chem. Biol.* **2021**, *17*, 872−877.

(23) Lee, A. Y.; Erdemir, D.; Myerson, A. S. Crystal polymorphism in chemical process development. *Annu. Rev. Chem. Biomol. Eng.* **2011**, *2*, 259−280.

(24) Kabsch, W. Xds. *Acta Crystallogr.* **2010**, *D66*, 125–132.

(25) Kabsch, W. Integration, scaling, space-group assignment and post-refinement. *Acta Crystallogr*. **2010,** *D66*, 133–144.

(26) Hattne, J.; Reyes, F. E.; Nannenga, B. L.; Shi, D.; de la Cruz, M. J.; Leslie, A. G. W.; Gonen, T. MicroED data collection and processing. *Acta Crystallogr.*, Sect. A: Found. Adv. **2015**, *71*, 353−360.

(27) Wang, B.; Zou, X.; Smeets, S. Automated serial rotation electron diffraction combined with cluster analysis: an efficient multi-crystal workflow for structure determination *IUCrJ*. **2019**, *6*, 854–867.

(28) Sheldrick, G. M. A short history of SHELX. *Acta Cryst.* **2008**, *A64*, 112–122.

(29) Sheldrick, G. M. SHELXT – Integrated space-group and crystal-structure determination. *Acta Cryst.* **2015** *A71*, 3–8.

(30) Sheldrick, G. M. Crystal structure refinement with SHELXL. *Acta Cryst.* **2015**, *C71*, 3–8.

(31) Hübschle, C. B., Sheldrick, G. M. & Dittrich, B. ShelXle: A Qt graphical user interface for SHELXL. *J. Appl. Cryst.* **2011,** *44*, 1281–1284.

(32) Delano, W. The PyMOL Molecular Graphics System (Schrodinger LLC). http://www.pymol.org.

(33) Hattne, J.; Shi, D.; Glynn, C.; Zee, C.-T.; Gallagher-Jones, M.; Martynowycz, M. W.; Rodriguez, J. A.; Gonen, T. Analysis of Global and Site-Specific Radiation Damage in Cryo-EM *Structure* **2018**, *5*, 759–766.

(34) Toenjes, S. T.; Gustafson, J. L. Atropisomerism in medicinal chemistry: challenges and opportunities *Future Med. Chem.* **2018**, *10*, 409–422.

(35) Zask, A.; Murphy, J.; Ellestad, G. A. Biological stereoselectivity of atropisomeric natural products and drugs *Chirality* **2013**, *25*, 265–274.

(36) Bester, S. M. *et al*. Structural and mechanistic bases for a potent HIV-1 capsid inhibitor *Science* **2020**, *370*, 360–364.

(37) Link, J. O. *et al* Clinical targeting of HIV capsid protein with a long-acting small molecule *Nature* **2020**, *584*, 614–618.

(38) Gilead Sciences, Study to evaluate the safety and efficacy of lenacapavir in combination with an optimized background regimen in heavily treatment experienced participants living with HIV-1 infection with multidrug resistance (CAPELLA). Identifier: NCT04150068. https://clinicaltrials.gov/ct2/show/NCT04150068.

(39) Macrae, C. F.; Sovago, I.; Cottrell, S. J.; Galek, P. T. A.; McCabe, P.; Pidcock, E.; Platings, M.; Shields, G. P.; Stevens, J. S.; Towler, M.; Wood, P. A. Mercury 4.0: from visualization to analysis, design, and production. *J. Appl. Cryst.* **2020**, *53*, 226–235.

(40) Bose, N.; Ogawa, A.; von Reuss, S. H.; Yim, J. J.; Ragsdale, E. J.; Sommer, R. J.; Schroeder, F. C. Complex small-molecule architectures regulate phenotypic plasticity in a nematode *Angew. Chem.* **2012**, *124*, 12606−12611.

(41) Kaletta, T.; Hengartner, M. O. Finding function in novel targets: C. elegans as a model organism. *Nat. Rev. Drug Discovery* **2006**, *5*, 387−399.

(42) Bumbarger, D. J.; Riebesell, M.; Rödelsperger, C.; Sommer, R. J. System-wide rewiring underlies behavioral differences in predatory and bacterial-feeding nematodes. *Cell* **2013**, *152*, 109−119.

(43) Bento, G.; Ogawa, A.; Sommer, R. J. Co-option of the hormone-signalling module dafachronic acid–DAF-12 in nematode evolution. *Nature* **2010**, *466*, 494−497.

(44) Von Reuss, S. H.; Schroeder, F. C. Combinatorial chemistry in nematodes: modular assembly of primary metabolism-derived building blocks. *Nat. Prod. Rep.* **2015**, *32*, 994−1006.

(45) Butcher, R. A. Natural products as chemical tools to dissect complex biology in C. elegans. *Curr. Opin. Chem. Biol.* **2019**, *50*, 138−144.

(46) Hatfield, D.; Rinehart, R. R.; Forrest, H. S. 166. 3-Ribosyluric acid. Part II. Isolation of the corresponding nucleotide from beef blood. *J. Chem. Soc.* **1963**, 899−902.

(47) Forrest, H. S.; Hatfield, D.; Lagowski, J. M. 199. Uric acid riboside. Part I. Isolation and reinvestigation of the structure. *J. Chem. Soc.* **1961**, 963−968.

(48) Hatfield, D.; Forrest, H. S. Biosynthesis of 3-ribosyluric acid (uric acid riboside). *Biochim. Biophys. Acta* **1962**, *62*, 185−187.

(49) Lohrmann, R.; Lagowski, J. M.; Forrest, H. S. 81. 3-Ribosyluric acid. Part III. Unambiguous syntheses of 3-ribosyluric acid and related compounds. *J. Chem. Soc.* **1964**, 451−459.

(50) Balaban, R. S.; Nemoto, S.; Finkel, T. Mitochondria, oxidants, and aging. *Cell* **2005**, *120*, 483−495.

(51) Wan, Q. L.; Fu, X.; Dai, W.; Yang, J.; Luo, Z.; Meng, X.; Liu, X.; Zhong, R.; Yang, H.; Zhou, Q. Uric acid induces stress resistance and extends the life span through activating the stress response factor DAF-16/FOXO and SKN-1/NRF2. *Aging* **2020**, *12*, 2840−2856.

(52) Kenyon, C. J. The genetics of ageing. *Nature* **2010**, *464*, 504−512.

(53) Patel, D. S.; Garza-Garcia, A.; Nanji, M.; McElwee, J. J.; Ackerman, D.; Driscoll, P. C.; Gems, D. Clustering of genetically defined allele classes in the Caenorhabditis elegans DAF-2 insulin/IGF-1 receptor. *Genetics* **2008**, *178*, 931−946.

(54) Artyukhin, A. B.; Zhang, Y. K.; Akagi, A. E.; Panda, O.; Sternberg, P. W.; Schroeder, F. C. Metabolomic “dark matter” dependent on peroxisomal β-oxidation in Caenorhabditis elegans. *J. Am. Chem. Soc.* **2018**, *140*, 2841− 2852.

3.8 Experimental Section

3.8.1 Materials and Methods

Samples prepared according to previously disclosed procedures outlined in **Table 3.1** below. Data was collected on a Thermo Fisher Talos F200C transmission electron microscope operating with an accelerating voltage of 200keV, corresponding to an electron wavelength of 0.0251 Å. Electron diffraction data was collected using a Thermo Fisher CetaD camera. Screening the TEM grid for microcrystals was performed at 2600x magnification in imaging mode. Crystals selected for data collection were isolated by a selected area aperture. Data was collected by taking images of the diffraction patterns generated by a continuously rotating crystal integrated continuously at a rate of 3 seconds per frame. This rotation was performed at a rate of 0.3° per second with a minimum and maximum tilt range of −65° to +65°. Crystals selected for data collection were isolated by a selected area aperture to reduce the background noise contributions and calibrated to eucentric height to stay in the aperture over the entire tilt range. Samples collected at cryogenic conditions were placed onto a Gatan 626 cryo holder. Slow cooling the sample includes inserting room temperature Gatan 626 cryo holder and cooling to cryogenic temperatures after insertion into the TEM. Plunge frozen samples were frozen in liquid nitrogen, placed onto a liquid nitrogen cooled Gatan 626 cryo holder, and inserted and maintained at cryogenic temperature for the duration of data collection on the electron microscope. All diffraction data was processed using the XDS suite of programs as controlled by a custom Python automation script.[1–3] Structure were solved *ab initio* by direct methods in SHELXT or SHELXD and refined with SHELXL using ShelXle.[4–7] Thermal parameters were refined anisotropically for all non-hydrogen atoms. Hydrogen atoms were assigned using the riding model.

Compound	Purification by Crystallization	Compound Source and References
3.1	Yes	Medicinal Chemistry[8]
3.2	No	Medicinal Chemistry[8,9]
3.3	Yes	Medicinal Chemistry[8]
3.4	No	Medicinal Chemistry[8,9]
3.5	Yes	Process Chemistry[10]
3.6	Yes	Medicinal Chemistry[8,9,11–15]
3.7	Yes	Process Chemistry[10]
3.8	Yes	Process Chemistry[10]
3.9	No	Process Chemistry[10]
3.10	Yes	Process Chemistry[10]
3.11	No*	Medicinal Chemistry[11,13]
3.12	No	Medicinal Chemistry[8,9,11–15]
3.13	No*	Medicinal Chemistry[11,14]
3.14	Yes*	Process Chemistry[3]
3.15	Yes*	Process Chemistry[16]
Figure 5a	No	Medicinal Chemistry[17–19]
Figure 5b	No	PROTACs[20]

Table 3.1 Source of thirty pharmaceutical compounds analyzed in this study. * = samples were recrystallized for the purpose of obtaining a crystal structure.

3.8.2 Room Temperature TEM Screening Procedure

Milligram to sub-milligram quantities of dry powder were placed into a dram vial as received and manually ground with a glass pipette. A pure carbon 200 mesh Cu grid or lacey carbon Cu grid was placed inside of the vial and gently shaken together with the powder to "dry load" the grid (Figure 3.19). The grid was removed with Dumont straight self-closing tweezers and the tweezers were gently tapped against a lab bench while holding the grid to shake off excess powder. This sample was clipped into a single tilt holder and inserted into a well-aligned Thermo Fisher Scientific Talos F200C transmission electron microscopy operating at an accelerating voltage of 200keV.

After achieving suitable pressure, the column valves were opened and the grid was manually scanned at 2600x magnification in imaging mode. To screen for crystallinity, an incident diffraction pattern was recorded by isolating a region of the particle using a selected area aperture and entering parallel-illuminated diffraction mode utilizing the low dose software on the Thermo Fisher microscope user interface.

A single image of the diffraction pattern was taken on a Thermo Fisher Scientific Ceta-D camera. If user inspection of the diffraction pattern suggested that the particle was monocrystalline and provided <1.2 Å resolution diffraction, the eucentric height of the sample was finely adjusted in imaging mode to ensure the crystal would remain within the selected area aperture throughout

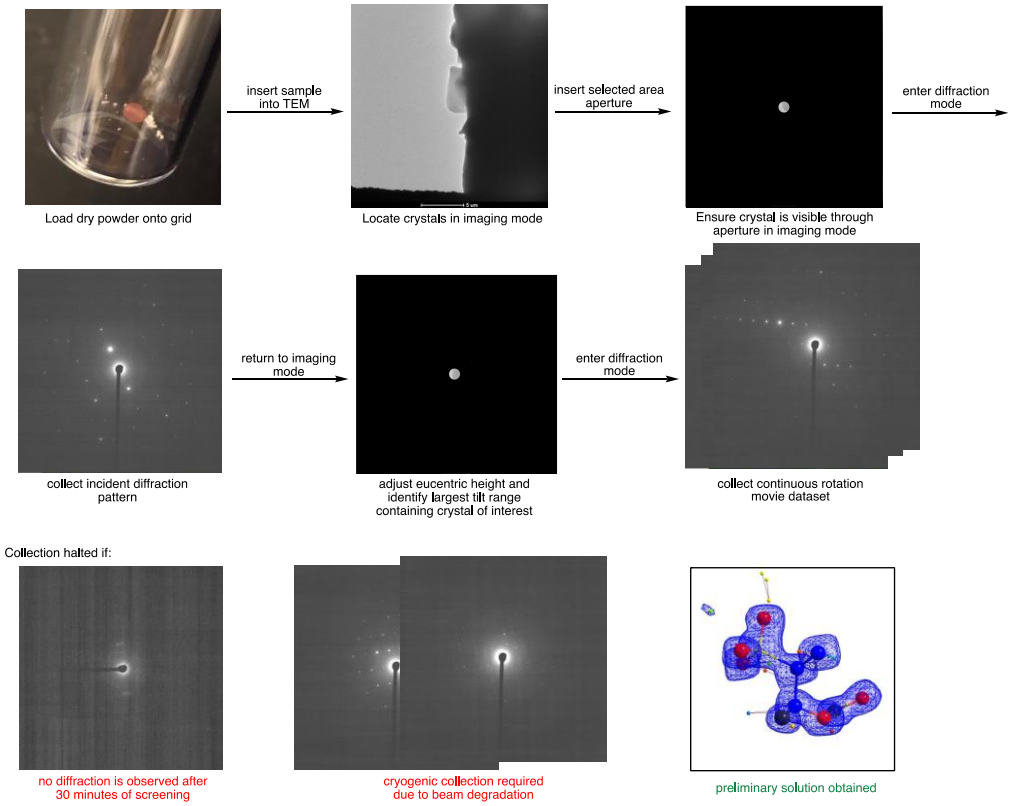


Figure 3.18 Representative data collection workflow.

a tilt series with a maximum tilt range of ± 65°. Upon returning to diffraction mode at eucentric height, a continuously rotating electron diffraction movie was collected. The stage was rotated at a rate of 0.3° s^{-1} and a detector distance of 960mm. The Ceta-D CMOS 4k x 4k camera was operated using rolling shutter mode and continuously integrated at a rate of 3 seconds per frame with binning by 2 to produce 2k x 2k images. Diffraction movies were saved as SER files. Movies were saved with a standardized naming format and processed using the automated data workflow

described herein while additional movies were collected. These processed movies were manually re-indexed to different space groups and/or merged with other datasets as needed until preliminary solutions were obtained.

Screening was halted if no diffraction was observed after 30 minutes, the sample visibly lost resolution over the course of a single movie, or a preliminary solution with >90% of expected atoms was obtained.

3.8.3 Room Temperature Screening Crystal Structures

3.8.3.1 (2R,3S)-N-(4-(2,6-dimethoxyphenyl)-5-(5-methylpyridin-3-yl)-4H-1,2,4-triazol-3-yl)-3-(5-methylpyrimidin-2-yl)butane-2-sulfonamide (3.1).

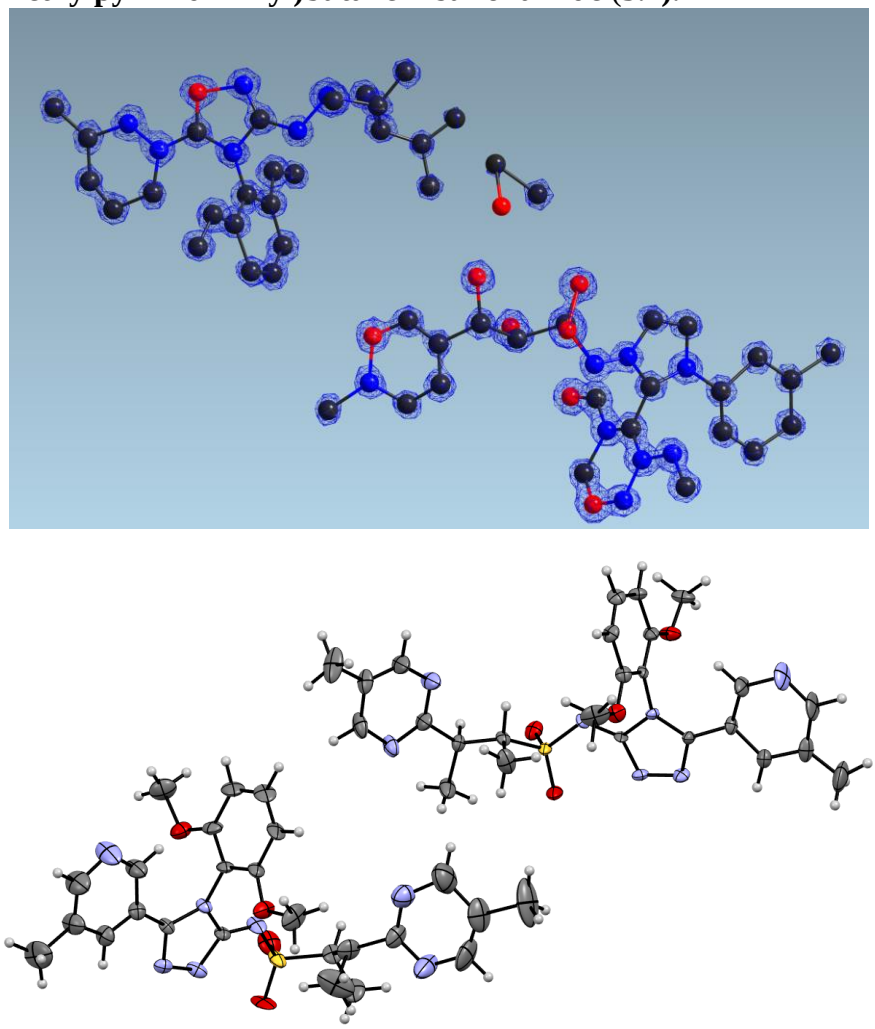


Figure 3.19 MicroED crystal structure of **3.1**. Initial direct methods solution of **3.1** (top) with electron density map (F_{obs}) contoured at 1.41 e $Å^{-3}$ and ORTEP diagram of refined **3.1** (bottom). Thermal ellipsoids shown as shaded octants at 30% probability.

Crystal data and structure refinement for 3.1.

Empirical formula	$C_{25}H_{29}N_7O_4S$

Formula weight	523.61

Data Collection

Type of instrument	Talos F200C
Wavelength	0.0215 Å
Data collection temperature	294(2) K
Unit cell dimensions	a = 9.3100(10)
	b = 20.490(2)
	c = 12.650(4)
	β = 108.42
Volume	2289.5(8)
Z	2
Crystal system	Monoclinic
Space group	$P2_1$
Density (calculated)	1.515 Mg/m^3
F(000)	100
Measured reflections	6341
Reflections with I > 2σ(*I*)	3849
Resolution	0.90 Å
Completeness	81.4%
Index ranges	$10 \leq h \leq -10$, $23 \leq k \leq$
	-24, $13 \leq l \leq -13$

Structure Solution and Refinement

Structure solution program	SHELXT (Uson & Sheldrick, 1999)
Primary solution method	Direct methods
Secondary solution method	Difference Fourier map
Hydrogen placement	Geometric positions
Structure refinement program	SHELXL-2018/3 (Sheldrick, 2018)
Refinement method	Full matrix least-squares on F^2
Data / restraints / parameters	6341 / 1069 / 668

Treatment of hydrogen atoms	Riding
Goodness-of-fit on F^2	1.243
Final R indices [I>2s(I), 975 reflections]	R1 = 0.1228, *w*R2 = 0.2924
R indices (all data)	R1 = 0.1678, *w*R2 = 0.3187
Type of weighting scheme used	Sigma
Weighting scheme used	$w=1/s^2(Fo^2)$
Max shift/error	0.044
Average shift/error	0.000
Largest diff. peak and hole	0.19 and -0.12 e.Å^{-3}

Special Refinement Details

Refinement of F^2 against ALL reflections. The weighted R-factor (wR) and goodness of fit (S) are based on F^2, conventional R-factors (R) are based on F, with F set to zero for negative F^2. The threshold expression of $F^2 > 2s(F^2)$ is used only for calculating R-factors(gt) etc. and is not relevant to the choice of reflections for refinement. R-factors based on F^2 are statistically about twice as large as those based on F, and R-factors based on ALL data will be even larger.

All esds (except the esd in the dihedral angle between two l.s. planes) are estimated using the full covariance matrix. The cell esds are taken into account individually in the estimation of esds in distances, angles and torsion angles; correlations between esds in cell parameters are only used when they are defined by crystal symmetry. An approximate (isotropic) treatment of cell esds is used for estimating esds involving l.s. planes.

3.8.3.2 (2R,3S)-3-(5-methoxypyridin-2-yl)butane-2-sulfonamide (3.2).

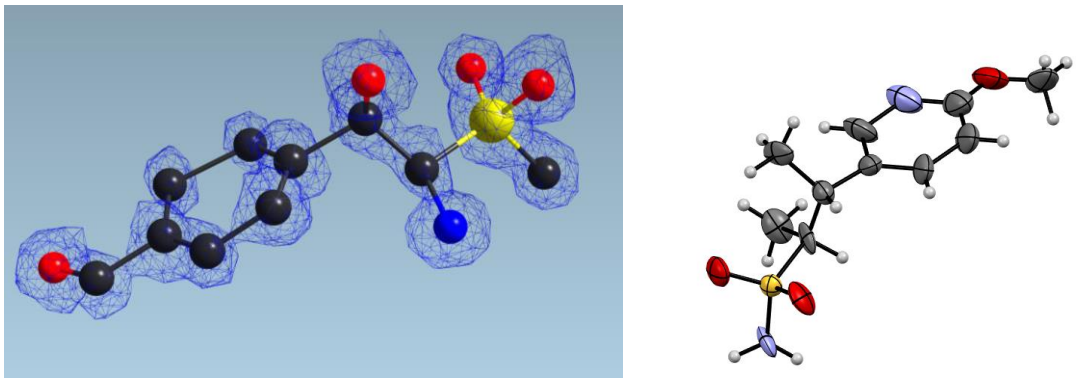


Figure 3.20 MicroED crystal structure of **3.2.** Initial direct methods solution of **3.2** (left) with electron density map (F_{obs}) contoured at 1.03 e $Å^{-3}$ and ORTEP diagram of refined **3.2** (right). Thermal ellipsoids shown as shaded octants at 30% probability.

Crystal data and structure refinement for 3.2.

Empirical formula	$C_{25}H_{29}N_7O_4S$
Formula weight	523.61

Data Collection

Type of instrument	Talos F200C
Wavelength	0.0215 Å
Data collection temperature	294(2) K
Unit cell dimensions	a = 22.830(4)
	b = 6.810(10)
	c = 6.980(2)
Volume	1085.2(4)
Z	1
Crystal system	Orthorhombic
Space group	$P2_12_12$
Density (calculated)	1.495 Mg/m^3
F(000)	103
Measured reflections	3816

Reflections with I > 2σ(*I*)	870
Resolution	0.95 Å
Completeness	82.9%
Index ranges	$7 \leq h \leq -7$, $25 \leq k \leq -25$, $7 \leq l \leq -7$

Structure Solution and Refinement

Structure solution program	SHELXT (Uson & Sheldrick, 1999)
Primary solution method	Direct methods
Secondary solution method	Difference Fourier map
Hydrogen placement	Geometric positions
Structure refinement program	SHELXL-2018/3 (Sheldrick, 2018)
Refinement method	Full matrix least-squares on F^2
Data / restraints / parameters	1360 / 135 / 146
Treatment of hydrogen atoms	Riding
Goodness-of-fit on F^2	1.480
Final R indices [I>2s(I), 975 reflections]	R1 = 0.1347, *w*R2 = 0.3138
R indices (all data)	R1 = 0.1779, *w*R2 = 0.3349
Type of weighting scheme used	Sigma
Weighting scheme used	$w=1/s^2(Fo^2)$
Max shift/error	0.051
Average shift/error	0.009
Largest diff. peak and hole	0.20 and -0.18 e.Å^{-3}

Special Refinement Details

Refinement of F^2 against ALL reflections. The weighted R-factor (wR) and goodness of fit (S) are based on F^2, conventional R-factors (R) are based on F, with F set to zero for negative F^2. The threshold expression of $F^2 > 2s(F^2)$ is used only for calculating R-factors(gt) etc. and is not relevant to the choice of reflections for refinement. R-factors based on F^2 are statistically about twice as large as those based on F, and R-factors based on ALL data will be even larger.

All esds (except the esd in the dihedral angle between two l.s. planes) are estimated using the full covariance matrix. The cell esds are taken into account individually in the estimation of esds in distances, angles and torsion angles; correlations between esds in cell parameters are only

used when they are defined by crystal symmetry. An approximate (isotropic) treatment of cell esds is used for estimating esds involving l.s. planes.

3.8.3.3 (2S,3R)-N-(4-(2,6-dimethoxyphenyl)-5-(5-methylpyridin-3-yl)-4H-1,2,4-triazol-3-yl)-3-(5-methylpyrimidin-2-yl)butane-2-sulfonamide (3.3).

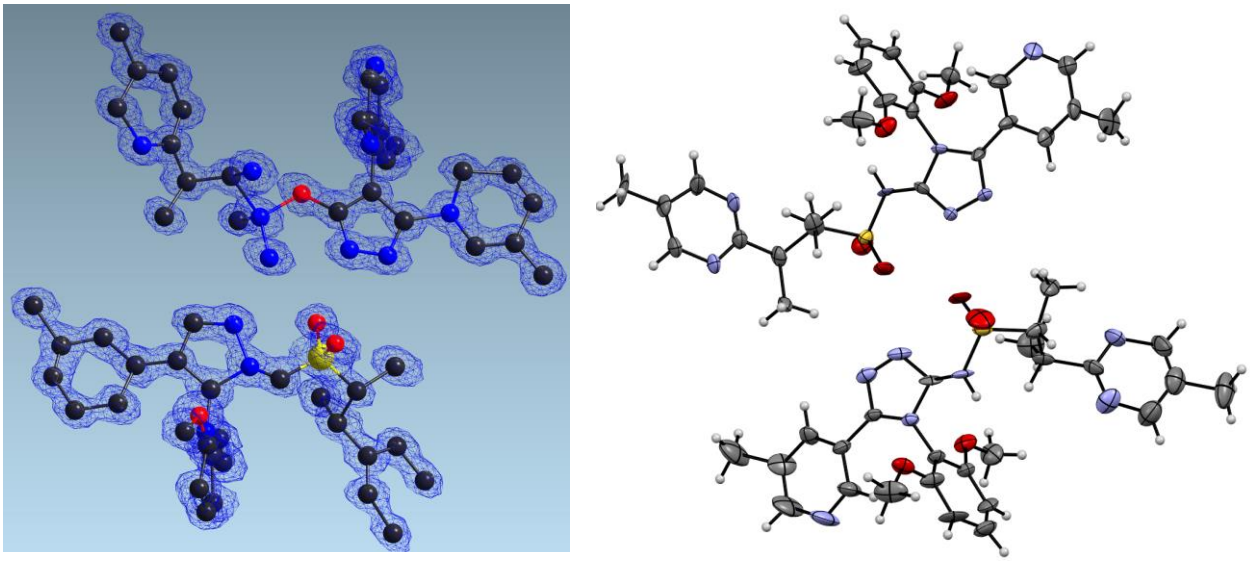


Figure 3.21 MicroED crystal structure of **3.3.** Initial direct methods solution of **3.3** (left) with electron density map (F_{obs}) contoured at 1.03 e Å^{-3} and ORTEP diagram of refined **3.3** (right). Thermal ellipsoids shown as shaded octants at 30% probability.

Crystal data and structure refinement for 3.3.

Empirical formula	$C_{25}H_{29}N_7O_4S$
Formula weight	523.61

Data Collection

Type of instrument	Talos F200C
Wavelength	0.0215 Å
Data collection temperature	294(2) K
Unit cell dimensions	a = 9.3100(10)
	b = 20.450(2)
	c = 12.680(4)
	β = 108.40
Volume	2290.7(8)
Z	2

Crystal system	Monoclinic
Space group	$P2_1$
Density (calculated)	1.518 Mg/m^3
F(000)	100
Measured reflections	5615
Reflections with $I > 2\sigma(I)$	3218
Resolution	0.90 Å
Completeness	85.5%
Index ranges	$10 \leq h \leq -10$, $21 \leq k \leq -21$, $14 \leq l \leq -14$

Structure Solution and Refinement

Structure solution program	SHELXT (Uson & Sheldrick, 1999)
Primary solution method	Direct methods
Secondary solution method	Difference Fourier map
Hydrogen placement	Geometric positions
Structure refinement program	SHELXL-2018/3 (Sheldrick, 2018)
Refinement method	Full matrix least-squares on F^2
Data / restraints / parameters	5615 / 677 / 668
Treatment of hydrogen atoms	Riding
Goodness-of-fit on F^2	1.226
Final R indices [I>2s(I), 975 reflections]	R1 = 0.1294, *w*R2 = 0.3063
R indices (all data)	R1 = 0.1777, *w*R2 = 0.3408
Type of weighting scheme used	Sigma
Weighting scheme used	$w=1/s^2(Fo^2)$
Max shift/error	0.016
Average shift/error	0.000
Largest diff. peak and hole	0.20 and -0.14 $e.Å^{-3}$

Special Refinement Details

Refinement of F^2 against ALL reflections. The weighted R-factor (wR) and goodness of fit (S) are based on F^2, conventional R-factors (R) are based on F, with F set to zero for negative

F^2. The threshold expression of $F^2 > 2s(F^2)$ is used only for calculating R-factors(gt) etc. and is not relevant to the choice of reflections for refinement. R-factors based on F^2 are statistically about twice as large as those based on F, and R-factors based on ALL data will be even larger.

All esds (except the esd in the dihedral angle between two l.s. planes) are estimated using the full covariance matrix. The cell esds are taken into account individually in the estimation of esds in distances, angles and torsion angles; correlations between esds in cell parameters are only used when they are defined by crystal symmetry. An approximate (isotropic) treatment of cell esds is used for estimating esds involving l.s. planes.

3.8.3.4 (2S,3R)-3-(5-methoxypyridin-2-yl)butane-2-sulfonamide (3.4)

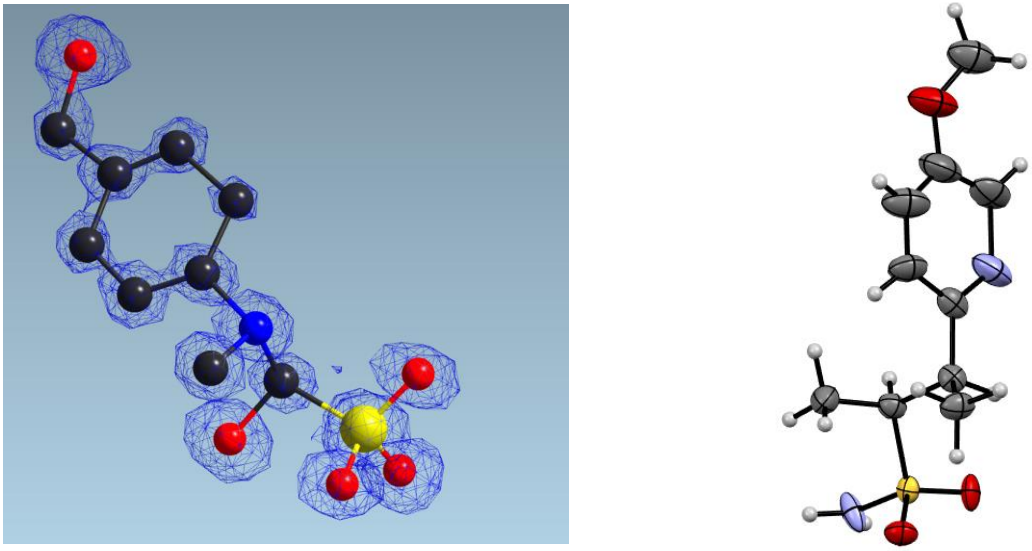


Figure 3.22 MicroED crystal structure of **3.4.** Initial direct methods solution of **3.4** (left) with electron density map (F_{obs}) contoured at 1.03 e $Å^{-3}$ and ORTEP diagram of refined **3.4** (right). Thermal ellipsoids shown as shaded octants at 30% probability.

Crystal data and structure refinement for 3.4.

Empirical formula	$C_{10}H_{16}N_2O_3S$
Formula weight	244.31

Data Collection

Type of instrument	Talos F200C
Wavelength	0.0215 Å
Data collection temperature	294(2) K
Unit cell dimensions	a = 6.840(2)
	b = 22.820(4)

	c = 6.9800(10)
Volume	1089.5(4)
Z	4
Crystal system	Orthorhombic
Space group	$P2_12_12$
Density (calculated)	1.489 Mg/m^3
F(000)	103
Measured reflections	1212
Reflections with I > 2σ(*I*)	649
Resolution	0.95 Å
Completeness	89.4%
Index ranges	24 ≤ h ≤ -24, 7 ≤ k ≤ -7, 7 ≤ l ≤ -7

Structure Solution and Refinement

Structure solution program	SHELXT (Uson & Sheldrick, 1999)
Primary solution method	Direct methods
Secondary solution method	Difference Fourier map
Hydrogen placement	Geometric positions
Structure refinement program	SHELXL-2018/3 (Sheldrick, 2018)
Refinement method	Full matrix least-squares on F^2
Data / restraints / parameters	1212 / 136 / 134
Treatment of hydrogen atoms	Riding
Goodness-of-fit on F^2	1.414
Final R indices [I>2s(I), 975 reflections]	R1 = 0.1706, *w*R2 = 0.3583
R indices (all data)	R1 = 0.2370, *w*R2 = 0.3910
Type of weighting scheme used	Sigma
Weighting scheme used	$w=1/s^2(Fo^2)$
Max shift/error	0.030
Average shift/error	0.000
Largest diff. peak and hole	0.18 and -0.18 $e.Å^{-3}$

Special Refinement Details

Refinement of F^2 against ALL reflections. The weighted R-factor (wR) and goodness of fit (S) are based on F^2, conventional R-factors (R) are based on F, with F set to zero for negative F^2. The threshold expression of $F^2 > 2s(F^2)$ is used only for calculating R-factors(gt) etc. and is not relevant to the choice of reflections for refinement. R-factors based on F^2 are statistically about twice as large as those based on F, and R-factors based on ALL data will be even larger.

All esds (except the esd in the dihedral angle between two l.s. planes) are estimated using the full covariance matrix. The cell esds are taken into account individually in the estimation of esds in distances, angles and torsion angles; correlations between esds in cell parameters are only used when they are defined by crystal symmetry. An approximate (isotropic) treatment of cell esds is used for estimating esds involving l.s. planes.

3.8.3.5 (R)-6'-chloro-3',4,4',5-tetrahydro-2H,2'H-spiro[benzo[b][1,4]oxazepine-3,1'-naphthalene]-7-carboxylic acid ((1R,4S)-7,7-dimethyl-2-oxobicyclo[2.2.1]heptan-1-yl)methansulfonate (3.5)

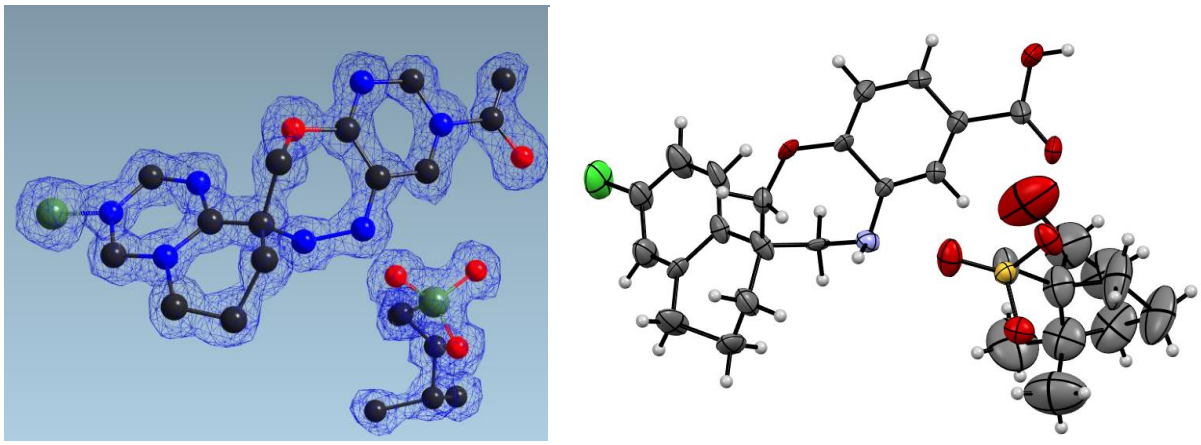


Figure 3.23 MicroED crystal structure of **3.5**. Initial direct methods solution of **3.5** (left) with electron density map (F_{obs}) contoured at 1.03 e $Å^{-3}$ and ORTEP diagram of refined **3.5** (right). Thermal ellipsoids shown as shaded octants at 30% probability.

Crystal data and structure refinement for 3.5.

Empirical formula	$C_{29}H_{34}NO_7SCl$
Formula weight	574.07

Data Collection

Type of instrument	Talos F200C
Wavelength	0.0215 Å
Data collection temperature	294(2) K
Unit cell dimensions	a = 10.5200(10)

	b = 10.220(2)
	c = 12.660(4)
	β = 110.33
Volume	1276.4(5)
Z	2
Crystal system	Monoclinic
Space group	$P2_1$
Density (calculated)	1.494 Mg/m^3
F(000)	99
Measured reflections	2438
Reflections with I > 2σ(*I*)	1379
Resolution	1.0 Å
Completeness	95.7%
Index ranges	$9 \leq h \leq -9$, $10 \leq k \leq -10$, $12 \leq l \leq -12$

Structure Solution and Refinement

Structure solution program	SHELXT (Uson & Sheldrick, 1999)
Primary solution method	Direct methods
Secondary solution method	Difference Fourier map
Hydrogen placement	Geometric positions
Structure refinement program	SHELXL-2018/3 (Sheldrick, 2018)
Refinement method	Full matrix least-squares on F^2
Data / restraints / parameters	2438 / 612 / 357
Treatment of hydrogen atoms	Riding
Goodness-of-fit on F^2	1.239
Final R indices [I>2s(I), 975 reflections]	R1 = 0.1198, *w*R2 = 0.2721
R indices (all data)	R1 = 0.1823, *w*R2 = 0.3103
Type of weighting scheme used	Sigma
Weighting scheme used	$w=1/s^2(Fo^2)$
Max shift/error	0.061

Average shift/error	0.001
Largest diff. peak and hole	0.13 and -0.13 e.Å^{-3}

Special Refinement Details

Refinement of F^2 against ALL reflections. The weighted R-factor (wR) and goodness of fit (S) are based on F^2, conventional R-factors (R) are based on F, with F set to zero for negative F^2. The threshold expression of $F^2 > 2s(F^2)$ is used only for calculating R-factors(gt) etc. and is not relevant to the choice of reflections for refinement. R-factors based on F^2 are statistically about twice as large as those based on F, and R-factors based on ALL data will be even larger.

All esds (except the esd in the dihedral angle between two l.s. planes) are estimated using the full covariance matrix. The cell esds are taken into account individually in the estimation of esds in distances, angles and torsion angles; correlations between esds in cell parameters are only used when they are defined by crystal symmetry. An approximate (isotropic) treatment of cell esds is used for estimating esds involving l.s. planes.

Significant disorder of the CSA moiety prevents assignment of absolute stereochemistry.

3.8.3.6 (2S,3R)-3-(5-methylpyrazin-2-yl)butane-2-sulfonamide (3.6)

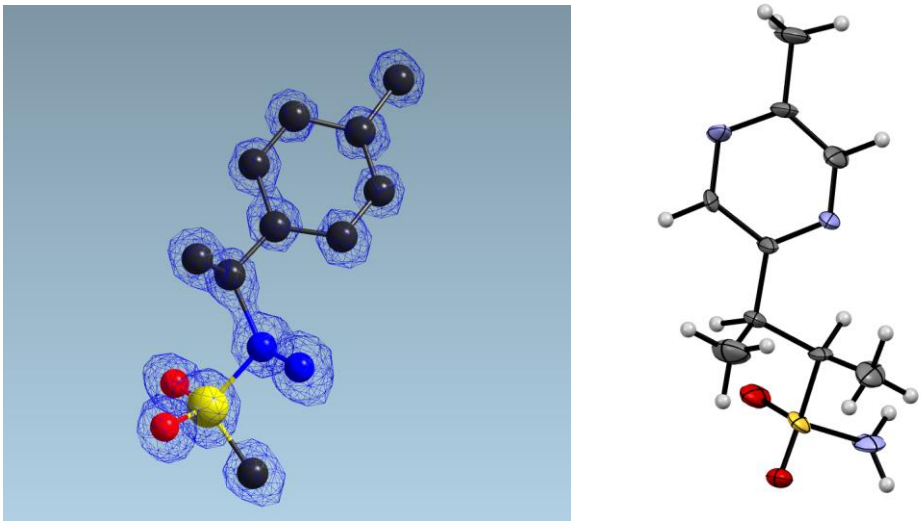


Figure 3.24 MicroED crystal structure of **3.6**. Initial direct methods solution of **3.6** (left) with electron density map (F_{obs}) contoured at 1.41 e Å^{-3} and ORTEP diagram of refined **3.6** (right). Thermal ellipsoids shown as shaded octants at 30% probability.

Crystal data and structure refinement for 3.6.

Empirical formula	$C_9H_{15}N_3O_2S$
Formula weight	229.30

Data Collection

Type of instrument	Talos F200C
Wavelength	0.0215 Å
Data collection temperature	294(2) K
Unit cell dimensions	a = 7.4500(10)
	b = 8.130(2)
	c = 16.240(4)
Volume	983.6(4)
Z	4
Crystal system	Orthorhombic
Space group	$P2_12_12_1$
Density (calculated)	1.548 Mg/m^3
F(000)	46
Measured reflections	1652
Reflections with $I > 2\sigma(I)$	975
Resolution	0.80 Å
Completeness	82.6%
Index ranges	$8 \leq h \leq -8$, $9 \leq k \leq$
	-9, $19 \leq l \leq -19$

Structure Solution and Refinement

Structure solution program	SHELXT (Uson & Sheldrick, 1999)
Primary solution method	Direct methods
Secondary solution method	Difference Fourier map
Hydrogen placement	Geometric positions
Structure refinement program	SHELXL-2018/3 (Sheldrick, 2018)
Refinement method	Full matrix least-squares on F^2
Data / restraints / parameters	1652 / 127 / 137
Treatment of hydrogen atoms	Riding
Goodness-of-fit on F^2	1.267

Final R indices [I>2s(I), 975 reflections]	R1 = 0.1808, *w*R2 = 0.4048
R indices (all data)	R1 = 0.2167, *w*R2 = 0.4398
Type of weighting scheme used	Sigma
Weighting scheme used	$w=1/s^2(Fo^2)$
Max shift/error	0.079
Average shift/error	0.001
Largest diff. peak and hole	0.21 and -0.38 e.Å^{-3}

Special Refinement Details

Refinement of F^2 against ALL reflections. The weighted R-factor (wR) and goodness of fit (S) are based on F^2, conventional R-factors (R) are based on F, with F set to zero for negative F^2. The threshold expression of $F^2 > 2s(F^2)$ is used only for calculating R-factors(gt) etc. and is not relevant to the choice of reflections for refinement. R-factors based on F^2 are statistically about twice as large as those based on F, and R-factors based on ALL data will be even larger.

All esds (except the esd in the dihedral angle between two l.s. planes) are estimated using the full covariance matrix. The cell esds are taken into account individually in the estimation of esds in distances, angles and torsion angles; correlations between esds in cell parameters are only used when they are defined by crystal symmetry. An approximate (isotropic) treatment of cell esds is used for estimating esds involving l.s. planes.

3.8.4 Cryogenic TEM Screening Procedure

Milligram to sub-milligram quantities of dry powder were placed into a dram vial as received and manually ground with a glass pipette. A pure carbon 200 mesh Cu grid or lacey carbon Cu grid was placed inside of the vial and gently shaken together with the powder to “dry load” the grid. The grid was removed with Dumont straight self-closing tweezers and the tweezers were gently tapped against a lab bench while holding the grid to shake off excess powder. This sample was clipped into a Gatan 626 cryo holder at room temperature and inserted into a well-aligned Thermo Fisher Scientific Talos F200C transmission electron microscopy operating at an accelerating voltage of 200keV. After successful insertion, the cryo holder was cooled with liquid nitrogen until reaching a stable temperature of ~–177 °C. After achieving

stable temperature and low vacuum pressure, incident diffraction screening and movie collection were performed as described in **Supporting Information Section 2**. Screening was halted after 3 hours, or if a preliminary solution with >90% of expected atoms was obtained.

3.8.5 Crystal Structures of Cryogenically Cooled Samples

3.8.5.1 (R)-6'-chloro-3',4,4',5-tetrahydro-2H,2'H-spiro[benzo[b][1,4]oxazepine-3,1'-naphthalene]-7-carboxylic acid ((1R,4S)-7,7-dimethyl-2-oxobicyclo[2.2.1]heptan-1-yl)methansulfonate (3.5a)

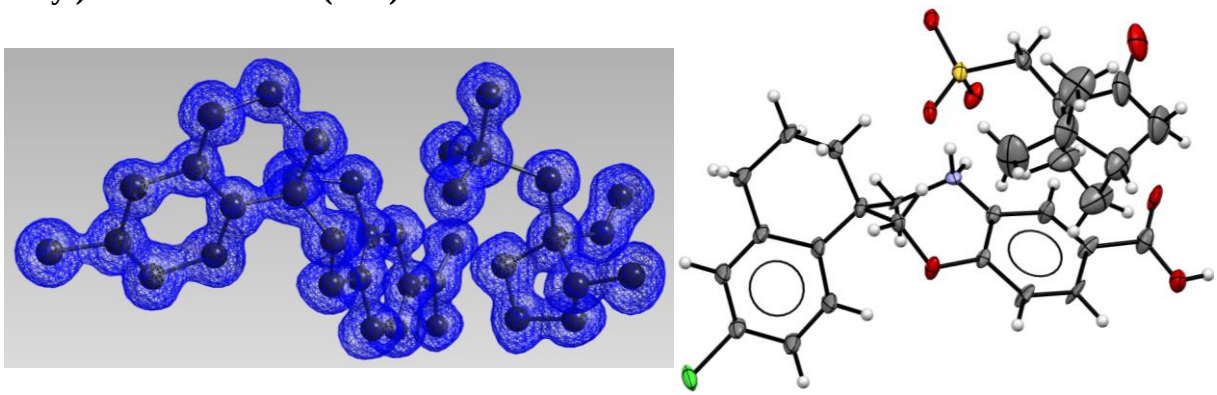


Figure 3.25 MicroED crystal structure of **3.5a**. Initial direct methods solution of **3.5a** (left) with electron density map (F_{obs}) contoured at 1.03 e $Å^{-3}$ and ORTEP diagram of refined **3.5a** (right). Thermal ellipsoids shown as shaded octants at 30% probability.

Crystal data and structure refinement for 3.5a.

Empirical formula	$C_{29}H_{34}NO_7SCl$
Formula weight	574.07

Data Collection

Type of instrument	Talos F200C
Wavelength	0.0215 Å
Data collection temperature	294(2) K
Unit cell dimensions	a = 10.4700(10)
	b = 10.260(2)
	c = 12.440(4)
	β = 109.92

Volume	1256.4(5)
Z	2
Crystal system	Monoclinic
Space group	$P2_1$
Density (calculated)	1.523 Mg/m^3
F(000)	99
Measured reflections	2474
Reflections with I > 2σ(*I*)	1787
Resolution	1.0 Å
Completeness	92.6%
Index ranges	$10 \leq h \leq -10$, $10 \leq k \leq -10$, $12 \leq l \leq -12$

Structure Solution and Refinement

Structure solution program	SHELXD (Uson & Sheldrick, 1999)
Primary solution method	Direct methods
Secondary solution method	Difference Fourier map
Hydrogen placement	Geometric positions
Structure refinement program	SHELXL-2018/3 (Sheldrick, 2018)
Refinement method	Full matrix least-squares on F^2
Data / restraints / parameters	2474 / 703 / 354
Treatment of hydrogen atoms	Riding
Goodness-of-fit on F^2	1.143
Final R indices [I>2s(I), 975 reflections]	R1 = 0.1127, *w*R2 = 0.2643
R indices (all data)	R1 = 0.1456, *w*R2 = 0.2863
Type of weighting scheme used	Sigma
Weighting scheme used	$w=1/s^2(Fo^2)$
Max shift/error	0.000
Average shift/error	0.000
Largest diff. peak and hole	0.17 and -0.16 $e.Å^{-3}$

Special Refinement Details

Refinement of F^2 against ALL reflections. The weighted R-factor (wR) and goodness of fit (S) are based on F^2, conventional R-factors (R) are based on F, with F set to zero for negative F^2. The threshold expression of $F^2 > 2s(F^2)$ is used only for calculating R-factors(gt) etc. and is not relevant to the choice of reflections for refinement. R-factors based on F^2 are statistically about twice as large as those based on F, and R-factors based on ALL data will be even larger.

All esds (except the esd in the dihedral angle between two l.s. planes) are estimated using the full covariance matrix. The cell esds are taken into account individually in the estimation of esds in distances, angles and torsion angles; correlations between esds in cell parameters are only used when they are defined by crystal symmetry. An approximate (isotropic) treatment of cell esds is used for estimating esds involving l.s. planes.

3.8.5.2 (1S,3'R,6'R,7'S,8'E,11'S,12'R)-6-chloro-7'-hydroxy-11',12'-dimethyl-3,4-dihydro-2H,15'H-spiro[naphthalene-1,22'-[20]oxa[13]thia[1,14]diazatetracyclo [14.7.2.$0^{3,6}$.$0^{19,24}$]pentacosa[8,16,18,24]tetraen]-15'-one 13',13'-dioxide (3.7)

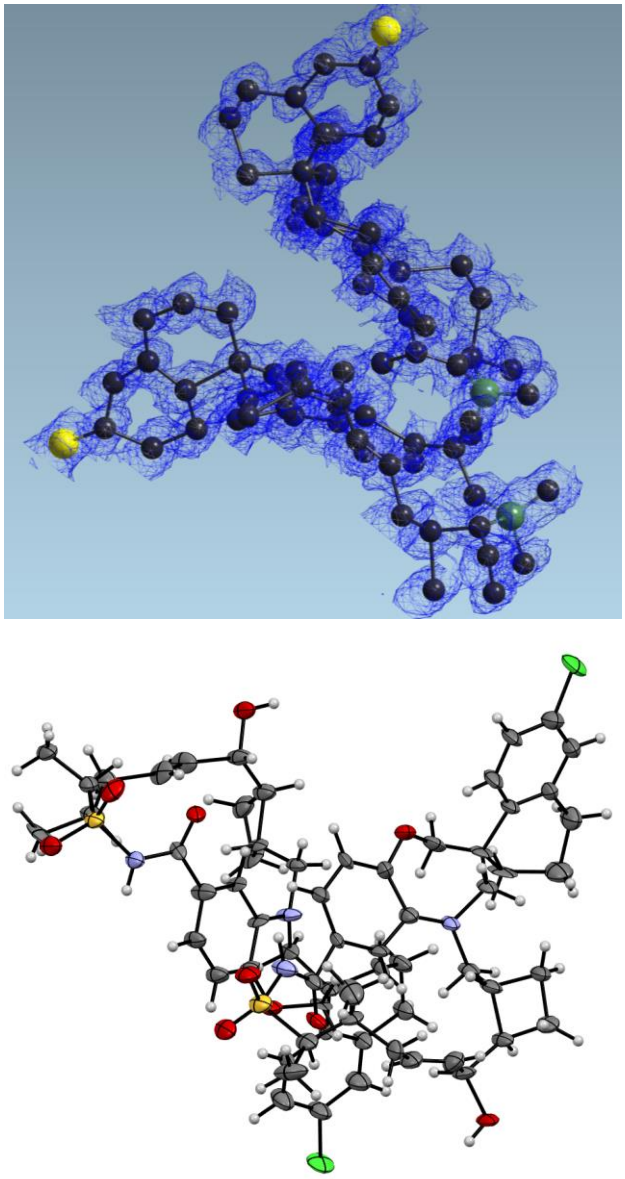


Figure 3.26 MicroED crystal structure of **3.7.** Initial direct methods solution of **3.7** (top) with electron density map (F_{obs}) contoured at 1.01 e $Å^{-3}$ and ORTEP diagram of refined **3.7** (bottom). Thermal ellipsoids shown as shaded octants at 30% probability.

Crystal data and structure refinement for 3.7.

Empirical formula	$C_{32}H_{39}N_2O_5SCl$
Formula weight	599.16

Data Collection

Type of instrument	Talos F200C
Wavelength	0.0215 Å
Data collection temperature	96(4) K
Unit cell dimensions	a = 11.3400(10)
	b = 11.340(2)
	c = 12.500(4)
	α = 73.74
	β = 69.36
	γ = 71.13
Volume	1398.2(5)
Z	2
Crystal system	Monoclinic
Space group	P_1
Density (calculated)	1.423 Mg/m^3
F(000)	243
Measured reflections	6522
Reflections with I > 2σ(*I*)	3782
Resolution	0.90 Å
Completeness	82.5%
Index ranges	$12 \leq h \leq -12$, $12 \leq k \leq$
	-12, $13 \leq l \leq -13$

Structure Solution and Refinement

Structure solution program	SHELXD (Uson & Sheldrick, 1999)

Primary solution method	Direct methods
Secondary solution method	Difference Fourier map
Hydrogen placement	Geometric positions
Structure refinement program	SHELXL-2018/3 (Sheldrick, 2018)
Refinement method	Full matrix least-squares on F^2
Data / restraints / parameters	6522 / 1253 / 740
Treatment of hydrogen atoms	Riding
Goodness-of-fit on F^2	1.242
Final R indices [I>2s(I), 975 reflections]	R1 = 0.1465, *w*R2 = 0.3107
R indices (all data)	R1 = 0.1954, *w*R2 = 0.3439
Type of weighting scheme used	Sigma
Weighting scheme used	$w=1/s^2(Fo^2)$
Max shift/error	0.083
Average shift/error	0.000
Largest diff. peak and hole	0.25 and -0.26 e.Å^{-3}

Special Refinement Details

Refinement of F^2 against ALL reflections. The weighted R-factor (wR) and goodness of fit (S) are based on F^2, conventional R-factors (R) are based on F, with F set to zero for negative F^2. The threshold expression of $F^2 > 2s(F^2)$ is used only for calculating R-factors(gt) etc. and is not relevant to the choice of reflections for refinement. R-factors based on F^2 are statistically about twice as large as those based on F, and R-factors based on ALL data will be even larger.

All esds (except the esd in the dihedral angle between two l.s. planes) are estimated using the full covariance matrix. The cell esds are taken into account individually in the estimation of esds in distances, angles and torsion angles; correlations between esds in cell parameters are only used when they are defined by crystal symmetry. An approximate (isotropic) treatment of cell esds is used for estimating esds involving l.s. planes.

3.8.5.3 (S)-6'-chloro-3',4,4',5-tetrahydro-2H,2'H-spiro[benzo[b][1,4]oxazepine-3,1'-naphthalene]-7-carboxylic acid ((1S,4R)-7,7-dimethyl-2-oxobicyclo[2.2.1]heptan-1-yl)methansulfonate (3.8)

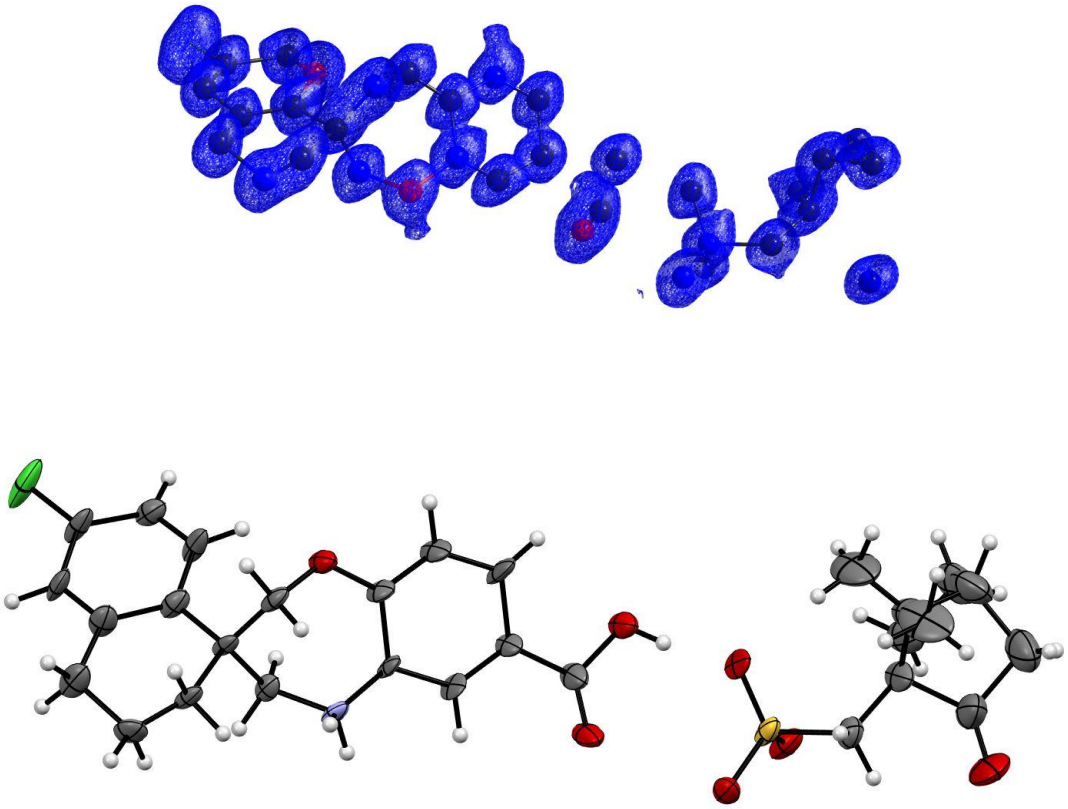


Figure 3.27 MicroED crystal structure of **3.8.** Initial direct methods solution of **3.8** (top) with electron density map (F_{obs}) contoured at 1.03 e $Å^{-3}$ and ORTEP diagram of refined **3.8** (bottom). Thermal ellipsoids shown as shaded octants at 30% probability.

Crystal data and structure refinement for 3.8.

Empirical formula	$C_{29}H_{34}NO_7SCl$
Formula weight	576.08

Data Collection

Type of instrument	Talos F200C
Wavelength	0.0215 Å
Data collection temperature	96(4) K
Unit cell dimensions	a = 10.6900(10)
	b = 10.220(2)

	c = 12.680(4)
	β = 111.22
Volume	1291.4(5)
Z	2
Crystal system	Monoclinic
Space group	$P2_1$
Density (calculated)	1.482 Mg/m^3
F(000)	99
Measured reflections	3824
Reflections with I > 2σ(*I*)	2409
Resolution	0.85 Å
Completeness	85.2%
Index ranges	12 ≤ h ≤ -12, 11 ≤ k ≤ -11, 14 ≤ l ≤ -14

Structure Solution and Refinement

Structure solution program	SHELXT (Uson & Sheldrick, 1999)
Primary solution method	Direct methods
Secondary solution method	Difference Fourier map
Hydrogen placement	Geometric positions
Structure refinement program	SHELXL-2018/3 (Sheldrick, 2018)
Refinement method	Full matrix least-squares on F^2
Data / restraints / parameters	3824 / 377 / 354
Treatment of hydrogen atoms	Riding
Goodness-of-fit on F^2	1.244
Final R indices [I>2s(I), 975 reflections]	R1 = 0.1415, *w*R2 = 0.3278
R indices (all data)	R1 = 0.1794, *w*R2 = 0.3523
Type of weighting scheme used	Sigma
Weighting scheme used	$w=1/s^2(Fo^2)$
Max shift/error	0.044
Average shift/error	0.002

Largest diff. peak and hole	0.15 and -0.16 e.Å^{-3}

Special Refinement Details

Refinement of F^2 against ALL reflections. The weighted R-factor (wR) and goodness of fit (S) are based on F^2, conventional R-factors (R) are based on F, with F set to zero for negative F^2. The threshold expression of $F^2 > 2s(F^2)$ is used only for calculating R-factors(gt) etc. and is not relevant to the choice of reflections for refinement. R-factors based on F^2 are statistically about twice as large as those based on F, and R-factors based on ALL data will be even larger.

All esds (except the esd in the dihedral angle between two l.s. planes) are estimated using the full covariance matrix. The cell esds are taken into account individually in the estimation of esds in distances, angles and torsion angles; correlations between esds in cell parameters are only used when they are defined by crystal symmetry. An approximate (isotropic) treatment of cell esds is used for estimating esds involving l.s. planes.

3.8.5.4 (S)-6'-chloro-5-(((1R,2R)-2-((S)-1-hydroxyallyl)cyclobutyl)methyl)-N-(((2R,3S)-3-methylhex-5-en-2-yl)sulfonyl)-3',4,4',5-tetrahydro-2H,2'H-spiro[benzo[B][1,4] oxazepine-3,1'-naphthalene]-7-carboxamide (3.9)

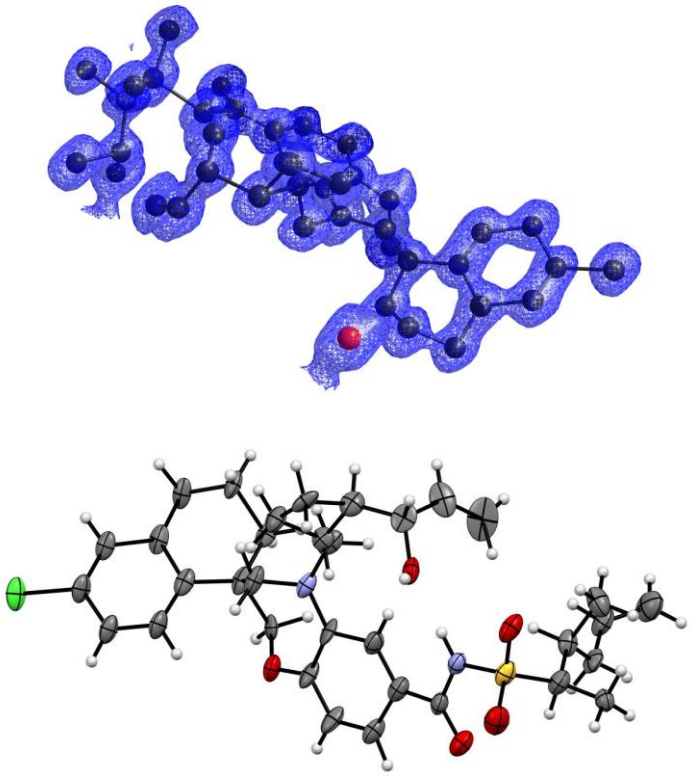


Figure 3.28 MicroED crystal structure of **3.9.** Initial direct methods solution of **3.9** (top) with electron density map (F_{obs}) contoured at 1.03 e $Å^{-3}$ and ORTEP diagram of refined **3.9** (bottom). Thermal ellipsoids shown as shaded octants at 30% probability.

Crystal data and structure refinement for 3.9.

Empirical formula	$C_{34}H_{43}N_2O_5SCl$
Formula weight	626.26

Data Collection

Type of instrument	Talos F200C
Wavelength	0.0215 Å
Data collection temperature	96(4) K
Unit cell dimensions	a = 10.5200(10)
	b = 15.050(2)

	c = 17.020(4)
Volume	2694.7(8)
Z	4
Crystal system	Orthorhombic
Space group	$P2_12_12_1$
Density (calculated)	1.546 Mg/m^3
F(000)	105
Measured reflections	2393
Reflections with I > 2σ(*I*)	1774
Resolution	1.0 Å
Completeness	83.1%
Index ranges	$10 \leq h \leq -10$, $15 \leq k \leq -15$, $15 \leq l \leq -14$

Structure Solution and Refinement

Structure solution program	SHELXD (Uson & Sheldrick, 1999)
Primary solution method	Direct methods
Secondary solution method	Difference Fourier map
Hydrogen placement	Geometric positions
Structure refinement program	SHELXL-2018/3 (Sheldrick, 2018)
Refinement method	Full matrix least-squares on F^2
Data / restraints / parameters	2393 / 748 / 390
Treatment of hydrogen atoms	Riding
Goodness-of-fit on F^2	1.399
Final R indices [I>2s(I), 975 reflections]	R1 = 0.1111, *w*R2 = 0.2457
R indices (all data)	R1 = 0.1460, *w*R2 = 0.2596
Type of weighting scheme used	Sigma
Weighting scheme used	$w=1/s^2(Fo^2)$
Max shift/error	0.045
Average shift/error	0.000
Largest diff. peak and hole	0.15 and -0.13 $e.Å^{-3}$

Special Refinement Details

Refinement of F^2 against ALL reflections. The weighted R-factor (wR) and goodness of fit (S) are based on F^2, conventional R-factors (R) are based on F, with F set to zero for negative F^2. The threshold expression of $F^2 > 2s(F^2)$ is used only for calculating R-factors(gt) etc. and is not relevant to the choice of reflections for refinement. R-factors based on F^2 are statistically about twice as large as those based on F, and R-factors based on ALL data will be even larger.

All esds (except the esd in the dihedral angle between two l.s. planes) are estimated using the full covariance matrix. The cell esds are taken into account individually in the estimation of esds in distances, angles and torsion angles; correlations between esds in cell parameters are only used when they are defined by crystal symmetry. An approximate (isotropic) treatment of cell esds is used for estimating esds involving l.s. planes.

3.8.5.5 (R)-1-phenylethan-1-aminium (S)-6'-chloro-5-(((1R,2R)-2-((S)-1-hydroxyallyl) cyclobutyl)methyl)-3',4,4',5-tetrahydro-2H,2'H-spiro[benzo[b][1,4]oxazepine-3,1'-naphthalene]-7-carboxylate (3.10)

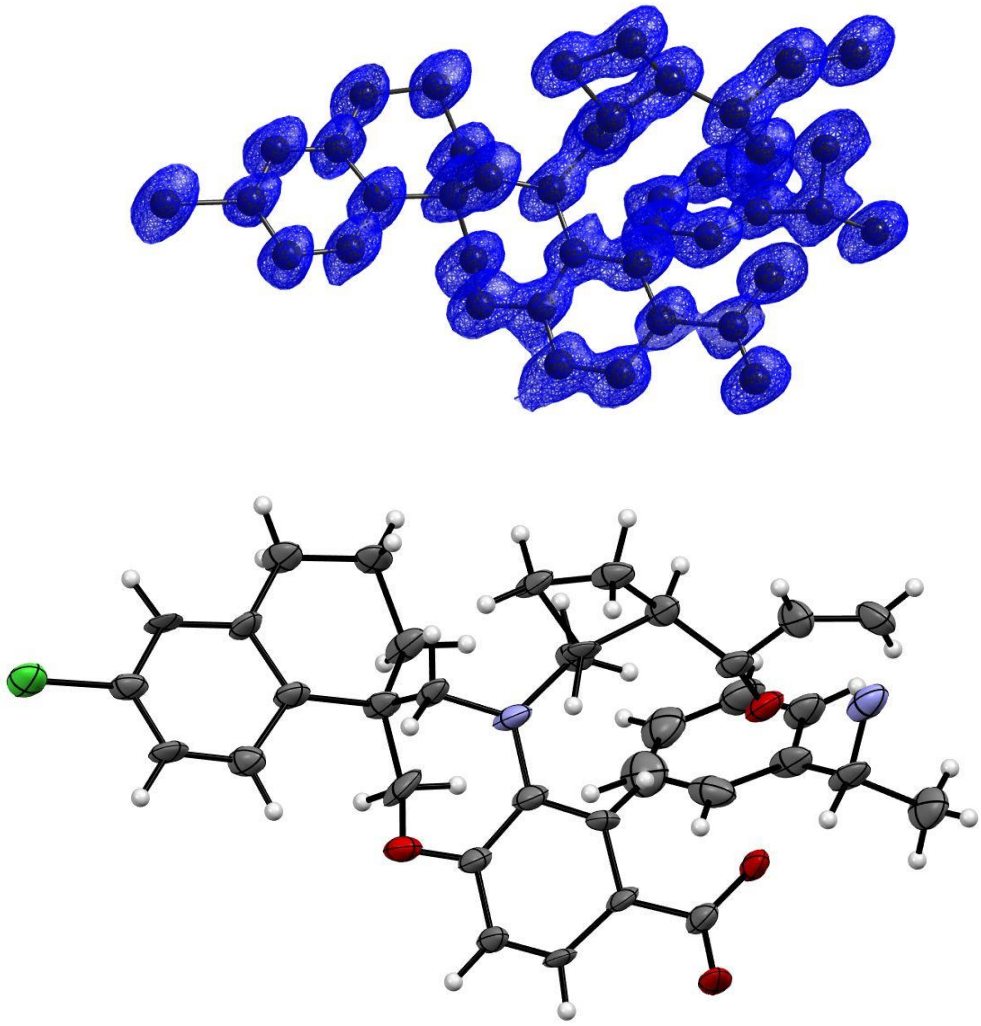


Figure 3.29 MicroED crystal structure of **3.10.** Initial direct methods solution of **3.10** (top) with electron density map (F_{obs}) contoured at 1.41 e $Å^{-3}$ and ORTEP diagram of refined **3.10** (bottom). Thermal ellipsoids shown as shaded octants at 30% probability.

Crystal data and structure refinement for 3.10.

Empirical formula	$C_{35}H_{36}N_2O_4Cl$
Formula weight	584.11

Data Collection

Type of instrument	Talos F200C
Wavelength	0.0215 Å
Data collection temperature	96(4) K
Unit cell dimensions	a = 7.9800(10)
	b = 11.730(2)
	c = 28.850(4)
Volume	2700.5(7)
Z	4
Crystal system	Orthorhombic
Space group	$P2_12_12_1$
Density (calculated)	1.437 Mg/m^3
F(000)	103
Measured reflections	3263
Reflections with I > 2σ(*I*)	2351
Resolution	0.85 Å
Completeness	68.6%
Index ranges	$9 \leq h \leq -9$, $13 \leq k \leq$
	-13, $25 \leq l \leq -25$

Structure Solution and Refinement

Structure solution program	SHELXD (Uson & Sheldrick, 1999)
Primary solution method	Direct methods
Secondary solution method	Difference Fourier map
Hydrogen placement	Geometric positions
Structure refinement program	SHELXL-2018/3 (Sheldrick, 2018)
Refinement method	Full matrix least-squares on F^2
Data / restraints / parameters	3263 / 390 / 380
Treatment of hydrogen atoms	Riding
Goodness-of-fit on F^2	1.469

Final R indices [I>2s(I), 975 reflections]	R1 = 0.1206, *w*R2 = 0.2740
R indices (all data)	R1 = 0.1536, *w*R2 = 0.2834
Type of weighting scheme used	Sigma
Weighting scheme used	$w=1/s^2(Fo^2)$
Max shift/error	0.026
Average shift/error	0.000
Largest diff. peak and hole	0.17 and -0.13 e.Å^{-3}

Special Refinement Details

Refinement of F^2 against ALL reflections. The weighted R-factor (wR) and goodness of fit (S) are based on F^2, conventional R-factors (R) are based on F, with F set to zero for negative F^2. The threshold expression of $F^2 > 2s(F^2)$ is used only for calculating R-factors(gt) etc. and is not relevant to the choice of reflections for refinement. R-factors based on F^2 are statistically about twice as large as those based on F, and R-factors based on ALL data will be even larger.

All esds (except the esd in the dihedral angle between two l.s. planes) are estimated using the full covariance matrix. The cell esds are taken into account individually in the estimation of esds in distances, angles and torsion angles; correlations between esds in cell parameters are only used when they are defined by crystal symmetry. An approximate (isotropic) treatment of cell esds is used for estimating esds involving l.s. planes.

3.8.6 Additional Screening and Recrystallization of Samples

Sample **3.12** was screened in the same manner as described in **Supporting Information Section 2** for ~4 additional hours at room temperature to locate monocrystalline domains in a largely polycrystalline sample.

Crystallization of **3.11**, **3.13**, **3.14**, and **3.15** was performed by placing ~1 mg of powder as received into 6 x 50 mm borosilicate culture tubes purchased from VWR. Samples were dissolved in approximately 500 uL of solvent and allowed to slowly evaporate at room temperature. Higher boiling solvents were evaporated from open containers, while low boiling solvents required placing the culture tube inside an empty dram vial with a slightly loosened cap. If the initial solvent failed to produce a solid after fully evaporating based on visual inspection, the amorphous samples were re-dissolved in the same culture tube with a new solvent mixture. Evaporation occurred until precipitation was observed. Sample crystallization time spanned from overnight to 3 days. **3.11** and **3.13** were obtained from slow evaporation from a 50/50 mixture of MeCN and H_2O. The crystals were dried under reduced pressure and screened at cryogenic temperatures as outlined in **Supporting Information Section 4**.

3.15 was generated from slow evaporation from H_2O with a small amount of DMSO. The crystals were blotted with a kimwipe and dried under reduced pressure to remove excess solvent before being brought into the TEM as described in **Supporting Information Section 4**.

3.14 was crystallized from slow evaporation of diethyl ether. The crystals were placed onto a grid as a dry powder, and screened by a modified procedure of **Supporting Information Section 4**. Before typical screening, the prepared grid was plunge frozen in liquid nitrogen and transferred into the TEM while the holder was maintained at cryogenic temperatures.

3.8.7 Crystal Structures of Additional Samples

3.8.7.1 (S)-1-(5-fluoropyrimidin-2-yl)piperidine-3-sulfonamide (3.11).

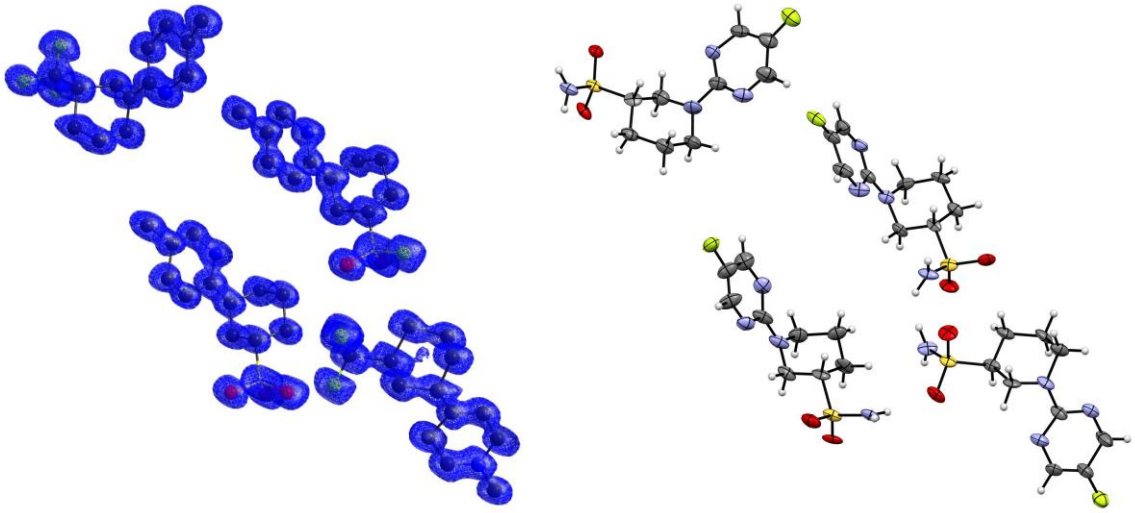


Figure 3.30 MicroED crystal structure of **3.11.** Initial direct methods solution of **3.11** (left) with electron density map (F_{obs}) contoured at 1.41 e $Å^{-3}$ and ORTEP diagram of refined **3.11** (right). Thermal ellipsoids shown as shaded octants at 30% probability.

Crystal data and structure refinement for 3.11.

Empirical formula	$C_9H_{13}N_4O_2SF$
Formula weight	260.29

Data Collection

Type of instrument	Talos F200C
Wavelength	0.0215 Å
Data collection temperature	96(4) K
Unit cell dimensions	a = 22.990(2)
	b = 37.240(4)
	c = 4.6400(10)
Volume	3972.5(10)

Z	16
Crystal system	Orthorhombic
Space group	$P2_12_12$
Density (calculated)	1.741 Mg/m^3
F(000)	43
Measured reflections	4964
Reflections with $I > 2\sigma(I)$	3031
Resolution	0.90 Å
Completeness	83.1%
Index ranges	$25 \leq h \leq -25$, $37 \leq k \leq -36$, $5 \leq l \leq -5$

Structure Solution and Refinement

Structure solution program	SHELXT (Uson & Sheldrick, 1999)
Primary solution method	Direct methods
Secondary solution method	Difference Fourier map
Hydrogen placement	Geometric positions
Structure refinement program	SHELXL-2018/3 (Sheldrick, 2018)
Refinement method	Full matrix least-squares on F^2
Data / restraints / parameters	4964 / 608 / 614
Treatment of hydrogen atoms	Riding
Goodness-of-fit on F^2	1.373
Final R indices [I>2s(I), 975 reflections]	R1 = 0.1254, *w*R2 = 0.2743
R indices (all data)	R1 = 0.1848, *w*R2 = 0.2955
Type of weighting scheme used	Sigma
Weighting scheme used	$w=1/s^2(Fo^2)$
Max shift/error	0.000
Average shift/error	0.000
Largest diff. peak and hole	0.20 and -0.16 $e.Å^{-3}$

Special Refinement Details

Refinement of F^2 against ALL reflections. The weighted R-factor (wR) and goodness of fit (S) are based on F^2, conventional R-factors (R) are based on F, with F set to zero for negative F^2. The threshold expression of $F^2 > 2s(F^2)$ is used only for calculating R-factors(gt) etc. and is not relevant to the choice of reflections for refinement. R-factors based on F^2 are statistically about twice as large as those based on F, and R-factors based on ALL data will be even larger.

All esds (except the esd in the dihedral angle between two l.s. planes) are estimated using the full covariance matrix. The cell esds are taken into account individually in the estimation of esds in distances, angles and torsion angles; correlations between esds in cell parameters are only used when they are defined by crystal symmetry. An approximate (isotropic) treatment of cell esds is used for estimating esds involving l.s. planes.

3.8.7.2 (2R,3S)-3-(5-methylpyrazin-2-yl)butane-2-sulfonamide (3.12).

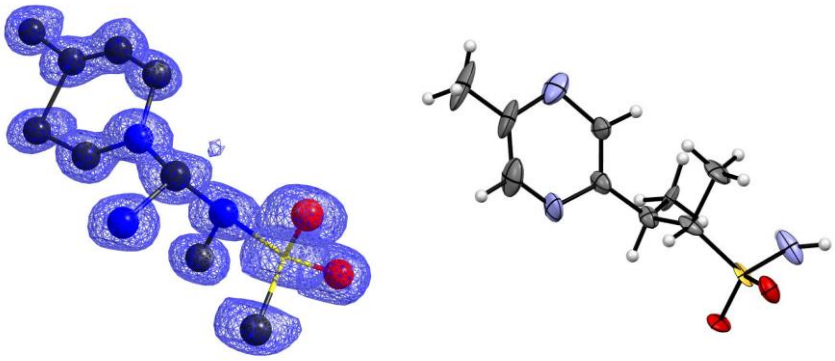


Figure 3.31 MicroED crystal structure of **3.12.** Initial direct methods solution of **3.12** (left) with electron density map (F_{obs}) contoured at 1.41 e $Å^{-3}$ and ORTEP diagram of refined **3.12** (right). Thermal ellipsoids shown as shaded octants at 30% probability.

Crystal data and structure refinement for 3.12.

Empirical formula	$C_9H_{15}N_3O_2S$
Formula weight	229.30

Data Collection

Type of instrument	Talos F200C
Wavelength	0.0215 Å
Data collection temperature	294(4) K
Unit cell dimensions	a = 22.0000(10)
	b = 6.410(2)

	c = 7.060(4)
	β = 91.18
Volume	995.4(4)
Z	4
Crystal system	Monoclinic
Space group	C2
Density (calculated)	1.530 Mg/m^3
F(000)	49
Measured reflections	1265
Reflections with I > 2σ(*I*)	789
Resolution	0.90 Å
Completeness	86.5%
Index ranges	$24 \leq h \leq -24$, $7 \leq k \leq -7$, $7 \leq l \leq -7$

Structure Solution and Refinement

Structure solution program	SHELXT (Uson & Sheldrick, 1999)
Primary solution method	Direct methods
Secondary solution method	Difference Fourier map
Hydrogen placement	Geometric positions
Structure refinement program	SHELXL-2018/3 (Sheldrick, 2018)
Refinement method	Full matrix least-squares on F^2
Data / restraints / parameters	1265 / 239 / 137
Treatment of hydrogen atoms	Riding
Goodness-of-fit on F^2	1.372
Final R indices [I>2s(I), 975 reflections]	R1 = 0.1348, *w*R2 = 0.3273
R indices (all data)	R1 = 0.1786, *w*R2 = 0.3500
Type of weighting scheme used	Sigma
Weighting scheme used	$w=1/s^2(Fo^2)$
Max shift/error	0.034
Average shift/error	0.000

Largest diff. peak and hole 0.14 and -0.18 e.Å^{-3}

Special Refinement Details

Refinement of F^2 against ALL reflections. The weighted R-factor (wR) and goodness of fit (S) are based on F^2, conventional R-factors (R) are based on F, with F set to zero for negative F^2. The threshold expression of $F^2 > 2s(F^2)$ is used only for calculating R-factors(gt) etc. and is not relevant to the choice of reflections for refinement. R-factors based on F^2 are statistically about twice as large as those based on F, and R-factors based on ALL data will be even larger.

All esds (except the esd in the dihedral angle between two l.s. planes) are estimated using the full covariance matrix. The cell esds are taken into account individually in the estimation of esds in distances, angles and torsion angles; correlations between esds in cell parameters are only used when they are defined by crystal symmetry. An approximate (isotropic) treatment of cell esds is used for estimating esds involving l.s. planes.

3.8.7.3 (R)-1-(5-fluoropyrimidin-2-yl)piperidine-3-sulfonamide (3.13).

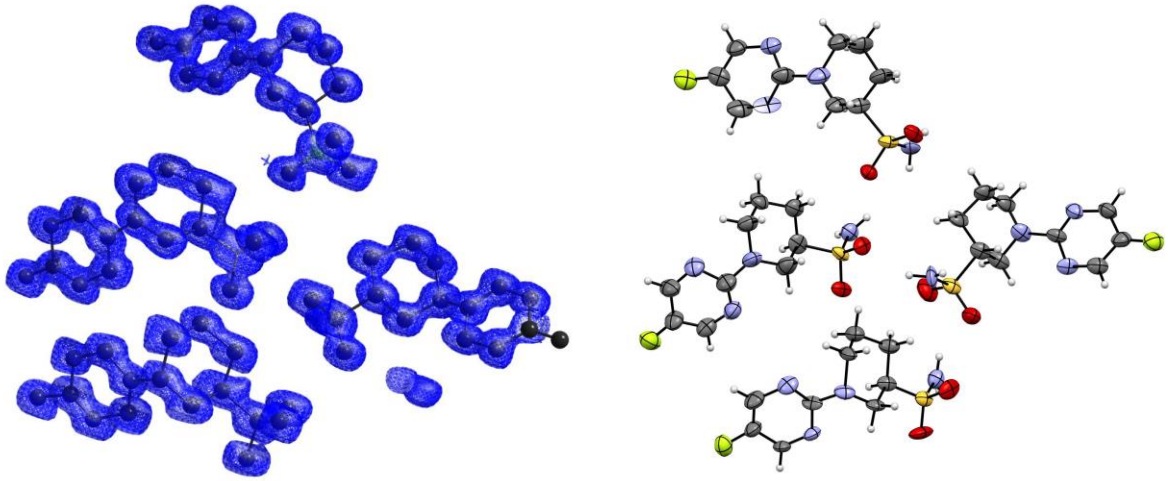


Figure 3.32 MicroED crystal structure of **3.13.** Initial direct methods solution of **3.13** (left) with electron density map (F_{obs}) contoured at 1.41 e Å^{-3} and ORTEP diagram of refined **3.13** (right). Thermal ellipsoids shown as shaded octants at 30% probability.

Crystal data and structure refinement for 3.13.

Empirical formula	$C_9H_{13}N_4O_2SF$
Formula weight	260.29

Data Collection

Type of instrument	Talos F200C
Wavelength	0.0215 Å
Data collection temperature	96(4) K
Unit cell dimensions	a = 23.000(2)
	b = 38.090(4)
	c = 4.6000(10)
Volume	4029.9(10)
Z	16
Crystal system	Orthorhombic
Space group	$P2_12_12$
Density (calculated)	1.716 Mg/m^3
F(000)	170
Measured reflections	4939
Reflections with $I > 2\sigma(I)$	3181
Resolution	0.90 Å
Completeness	82.6%
Index ranges	$25 \leq h \leq -25$, $38 \leq k \leq$
	-38, $5 \leq l \leq -5$

Structure Solution and Refinement

Structure solution program	SHELXD (Uson & Sheldrick, 1999)
Primary solution method	Direct methods
Secondary solution method	Difference Fourier map
Hydrogen placement	Geometric positions
Structure refinement program	SHELXL-2018/3 (Sheldrick, 2018)
Refinement method	Full matrix least-squares on F^2
Data / restraints / parameters	4939 / 608 / 614
Treatment of hydrogen atoms	Riding
Goodness-of-fit on F^2	1.323

Final R indices [I>2s(I), 975 reflections]	R1 = 0.1373, *w*R2 = 0.3069
R indices (all data)	R1 = 0.1752, *w*R2 = 0.3300
Type of weighting scheme used	Sigma
Weighting scheme used	$w=1/s^2(Fo^2)$
Max shift/error	0.095
Average shift/error	0.000
Largest diff. peak and hole	0.18 and -0.13 e.Å^{-3}

Special Refinement Details

Refinement of F^2 against ALL reflections. The weighted R-factor (wR) and goodness of fit (S) are based on F^2, conventional R-factors (R) are based on F, with F set to zero for negative F^2. The threshold expression of $F^2 > 2s(F^2)$ is used only for calculating R-factors(gt) etc. and is not relevant to the choice of reflections for refinement. R-factors based on F^2 are statistically about twice as large as those based on F, and R-factors based on ALL data will be even larger.

All esds (except the esd in the dihedral angle between two l.s. planes) are estimated using the full covariance matrix. The cell esds are taken into account individually in the estimation of esds in distances, angles and torsion angles; correlations between esds in cell parameters are only used when they are defined by crystal symmetry. An approximate (isotropic) treatment of cell esds is used for estimating esds involving l.s. planes.

3.8.7.4 (2R,3S)-3-methylhex-5-ene-2-sulfonamide (3.14).

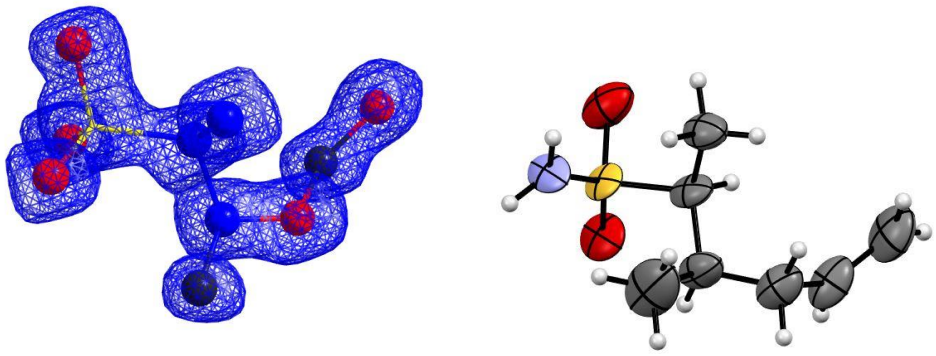


Figure 3.33 MicroED crystal structure of **3.14.** Initial direct methods solution of **3.14** (left) with electron density map (F_{obs}) contoured at 1.41 e Å^{-3} and ORTEP diagram of refined **3.14** (right). Thermal ellipsoids shown as shaded octants at 30% probability.

Crystal data and structure refinement for 3.14.

Empirical formula	$C_7H_{15}NO_2S$
Formula weight	177.26

Data Collection

Type of instrument	Talos F200C
Wavelength	0.0215 Å
Data collection temperature	96(4) K
Unit cell dimensions	a = 7.4100(10)
	b = 9.270(2)
	c = 12.490(4)
Volume	857.9(4)
Z	4
Crystal system	Orthorhombic
Space group	$P2_12_12_1$
Density (calculated)	1.372 Mg/m^3
F(000)	29
Measured reflections	1002
Reflections with I > 2σ(*I*)	461
Resolution	0.90 Å
Completeness	80.3%
Index ranges	$8 \leq h \leq -8$, $9 \leq k \leq$
	-9, $12 \leq l \leq -12$

Structure Solution and Refinement

Structure solution program	SHELXT (Uson & Sheldrick, 1999)
Primary solution method	Direct methods
Secondary solution method	Difference Fourier map
Hydrogen placement	Geometric positions
Structure refinement program	SHELXL-2018/3 (Sheldrick, 2018)
Refinement method	Full matrix least-squares on F^2
Data / restraints / parameters	1002 / 84 / 101

Treatment of hydrogen atoms	Riding
Goodness-of-fit on F^2	1.326
Final R indices [I>2s(I), 975 reflections]	R1 = 0.1396, *w*R2 = 0.3178
R indices (all data)	R1 = 0.2144, *w*R2 = 0.3575
Type of weighting scheme used	Sigma
Weighting scheme used	$w=1/s^2(Fo^2)$
Max shift/error	0.099
Average shift/error	0.002
Largest diff. peak and hole	0.12 and -0.12 e.Å^{-3}

Special Refinement Details

Refinement of F^2 against ALL reflections. The weighted R-factor (wR) and goodness of fit (S) are based on F^2, conventional R-factors (R) are based on F, with F set to zero for negative F^2. The threshold expression of $F^2 > 2s(F^2)$ is used only for calculating R-factors(gt) etc. and is not relevant to the choice of reflections for refinement. R-factors based on F^2 are statistically about twice as large as those based on F, and R-factors based on ALL data will be even larger.

All esds (except the esd in the dihedral angle between two l.s. planes) are estimated using the full covariance matrix. The cell esds are taken into account individually in the estimation of esds in distances, angles and torsion angles; correlations between esds in cell parameters are only used when they are defined by crystal symmetry. An approximate (isotropic) treatment of cell esds is used for estimating esds involving l.s. planes.

3.8.7.5 methyl piperazine-1-carboxylate phosphate hydrate (3.15)

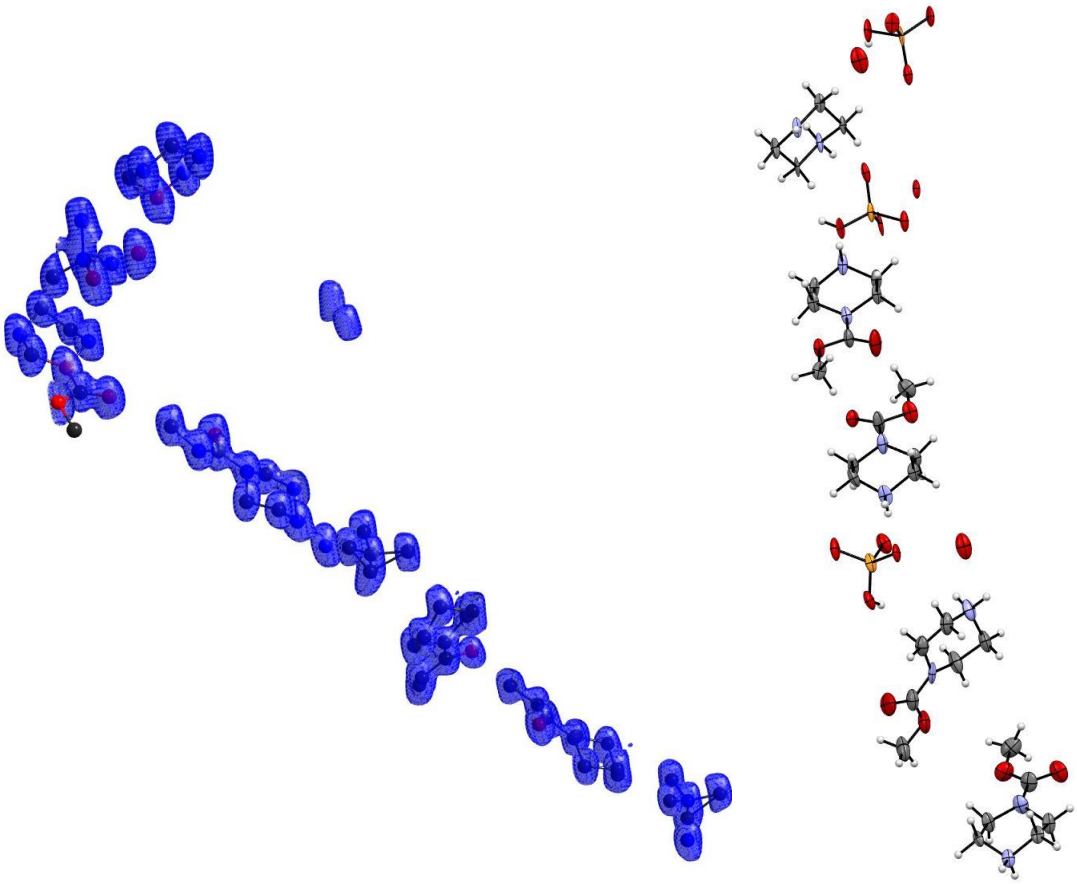


Figure 3.34 MicroED crystal structure of **3.15.** Initial direct methods solution of **3.15** (left) with electron density map (F_{obs}) contoured at 1.41 e $Å^{-3}$ and ORTEP diagram of refined **3.15** (right). Thermal ellipsoids shown as shaded octants at 30% probability.

Crystal data and structure refinement for 3.15.

Empirical formula	$C_{28}H_{66}N_{10}O_{23}P_3$
Formula weight	1003.82

Data Collection

Type of instrument	Talos F200C
Wavelength	0.0215 Å

Data collection temperature	96(4) K
Unit cell dimensions	a = 66.2000(10)
	b = 6.220(2)
	c = 9.940(4)
	β = 92.14
Volume	4090(2)
Z	4
Crystal system	Monoclinic
Space group	Cc
Density (calculated)	1.630 Mg/m^3
F(000)	169
Measured reflections	5976
Reflections with I > 2σ(*I*)	3714
Resolution	0.85 Å
Completeness	84.6%
Index ranges	$70 \leq h \leq -71$, $7 \leq k \leq -7$, $11 \leq l \leq -11$

Structure Solution and Refinement

Structure solution program	SHELXT (Uson & Sheldrick, 1999)
Primary solution method	Direct methods
Secondary solution method	Difference Fourier map
Hydrogen placement	Geometric positions
Structure refinement program	SHELXL-2018/3 (Sheldrick, 2018)
Refinement method	Full matrix least-squares on F^2
Data / restraints / parameters	5976 / 873 / 581
Treatment of hydrogen atoms	Riding
Goodness-of-fit on F^2	1.345
Final R indices [I>2s(I), 975 reflections]	R1 = 0.1383, *w*R2 = 0.2795
R indices (all data)	R1 = 0.1901, *w*R2 = 0.2995
Type of weighting scheme used	Sigma

Weighting scheme used	$w=1/s^2(Fo^2)$
Max shift/error	0.001
Average shift/error	0.000
Largest diff. peak and hole	0.29 and -0.28 e.Å^{-3}

Special Refinement Details

Refinement of F^2 against ALL reflections. The weighted R-factor (wR) and goodness of fit (S) are based on F^2, conventional R-factors (R) are based on F, with F set to zero for negative F^2. The threshold expression of $F^2 > 2s(F^2)$ is used only for calculating R-factors(gt) etc. and is not relevant to the choice of reflections for refinement. R-factors based on F^2 are statistically about twice as large as those based on F, and R-factors based on ALL data will be even larger.

All esds (except the esd in the dihedral angle between two l.s. planes) are estimated using the full covariance matrix. The cell esds are taken into account individually in the estimation of esds in distances, angles and torsion angles; correlations between esds in cell parameters are only used when they are defined by crystal symmetry. An approximate (isotropic) treatment of cell esds is used for estimating esds involving l.s. planes.

3.8.8 Transmission Electron Microscope Images of Pharmaceutical Crystals

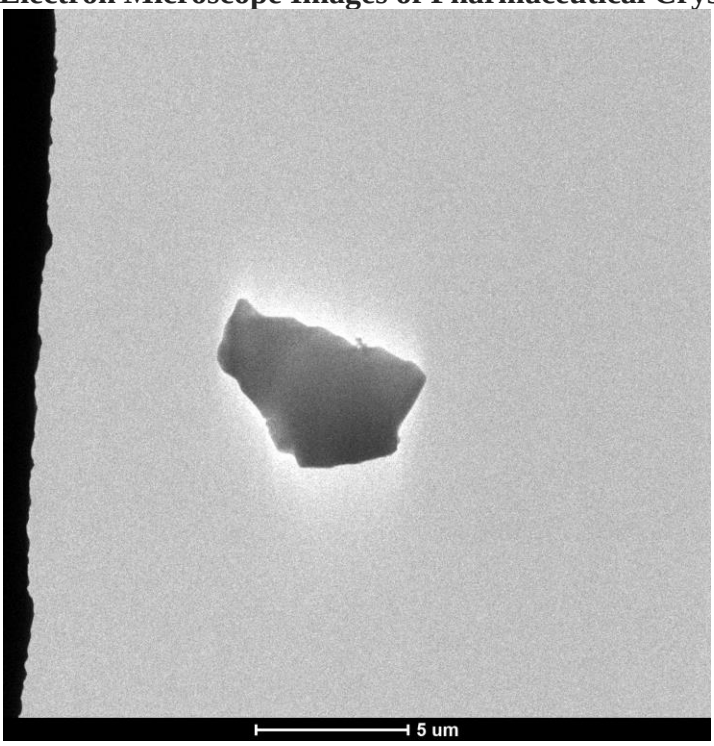


Figure 3.35 TEM image of **3.1** crystal at 2600x magnification

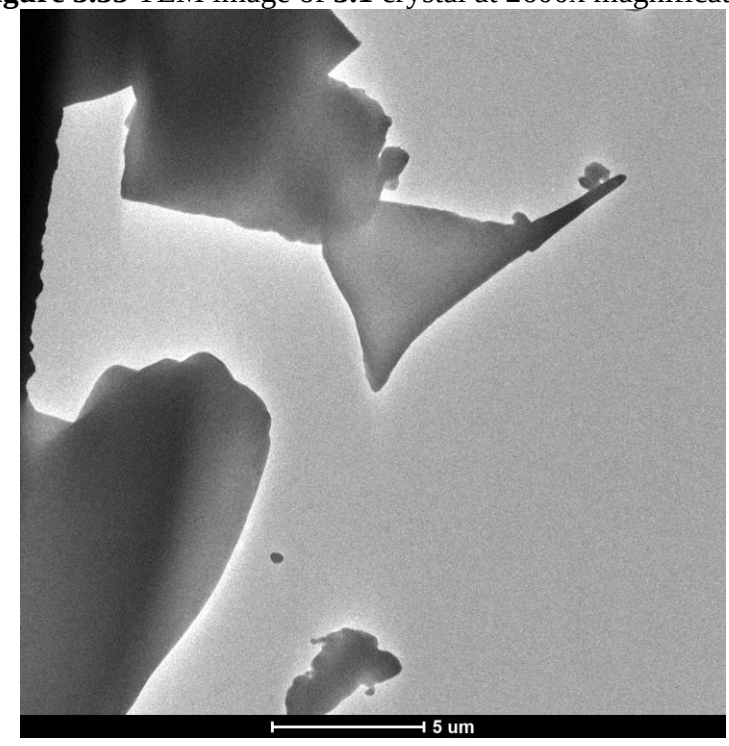


Figure 3.36 TEM image of **3.2** crystal at 2600x magnification

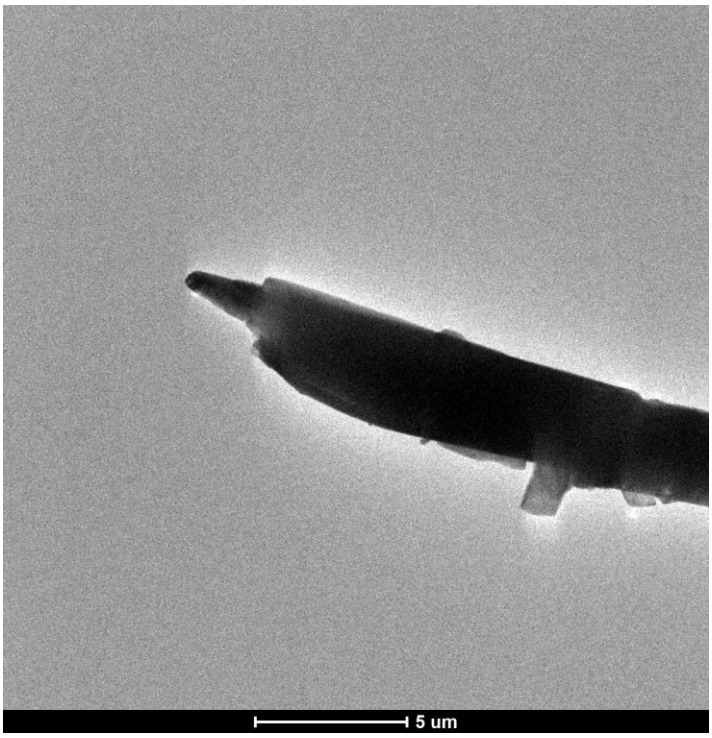


Figure 3.37 TEM image of **3.3** crystal at 2600x magnification

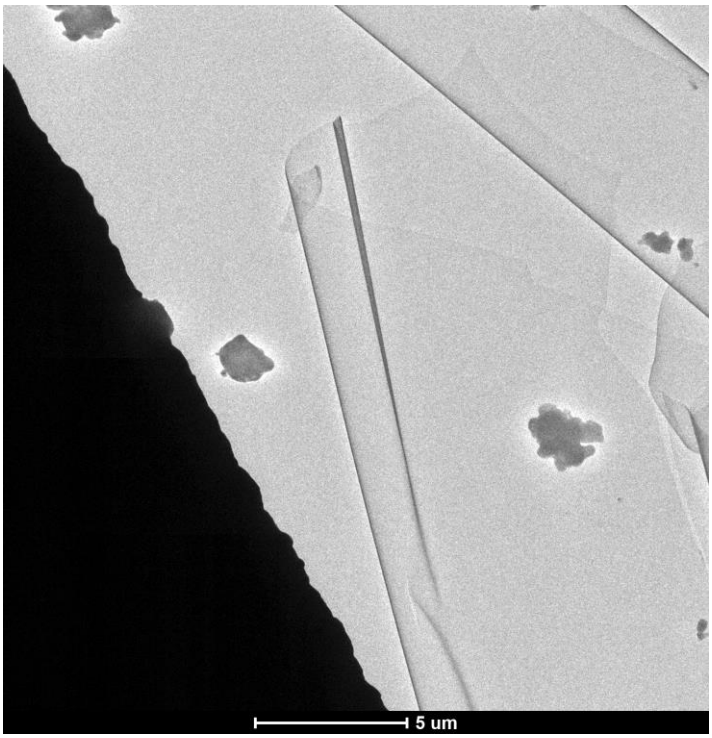


Figure 3.38 TEM image of **3.4** crystal at 2600x magnification

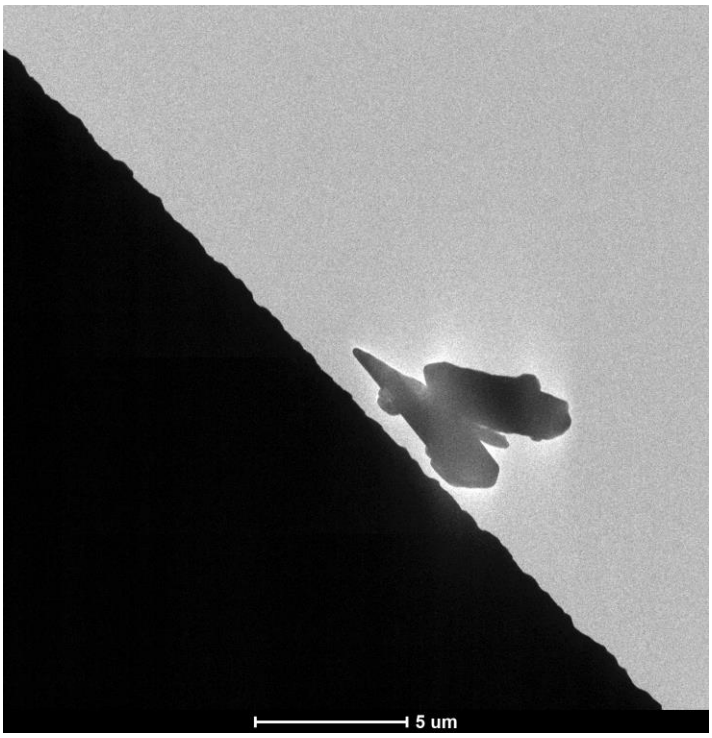


Figure 3.39 TEM image of **3.5** crystal at 2600x magnification

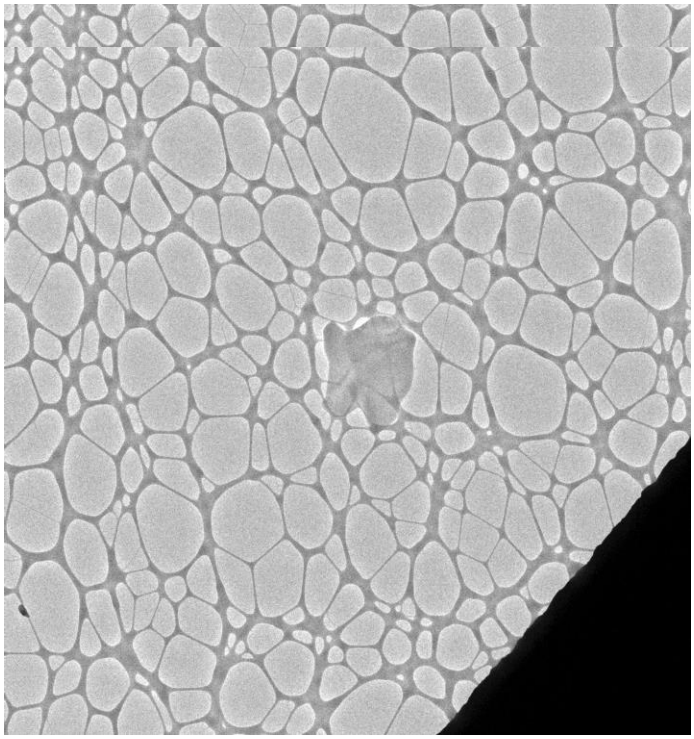


Figure 3.40 TEM image of **3.6** crystal at 2600x magnification.

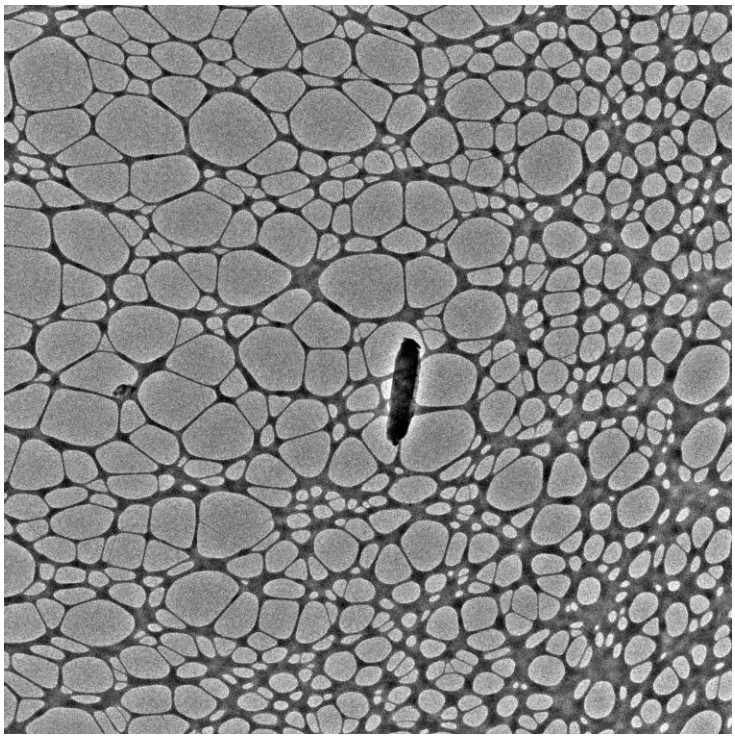


Figure 3.41TEM image of **3.7** crystal at 2600x magnification.

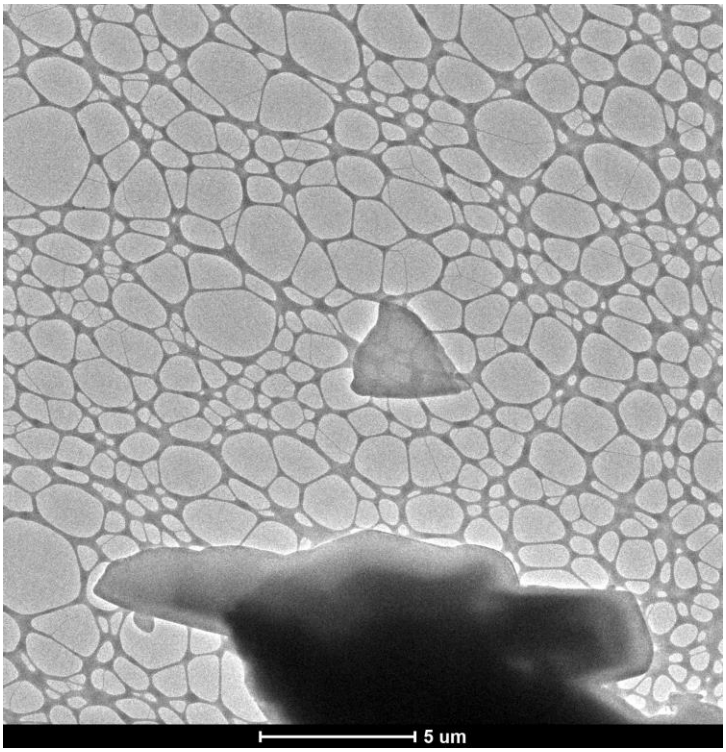


Figure 3.42 TEM image of **3.8** crystal at 2600x magnification.

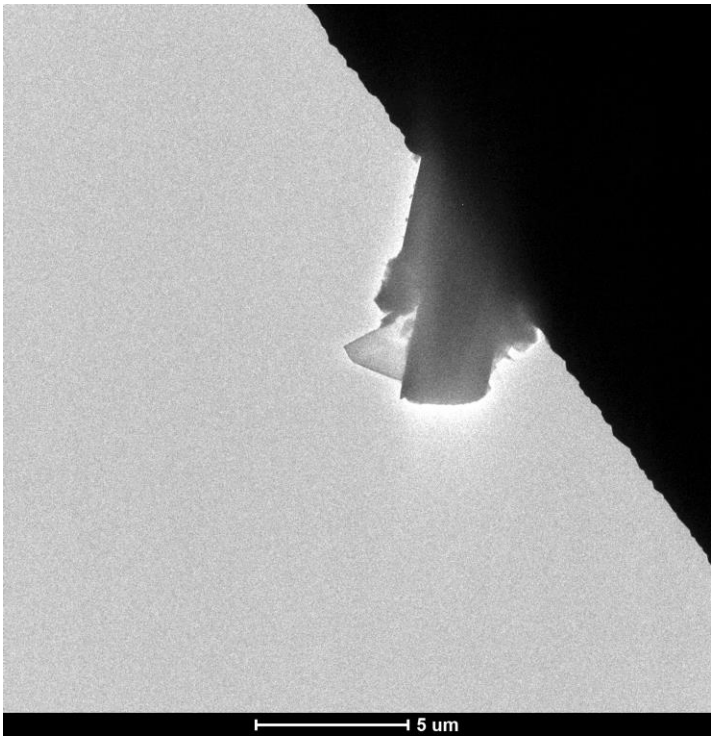


Figure 3.43 TEM image of **3.9** crystal at 2600x magnification.

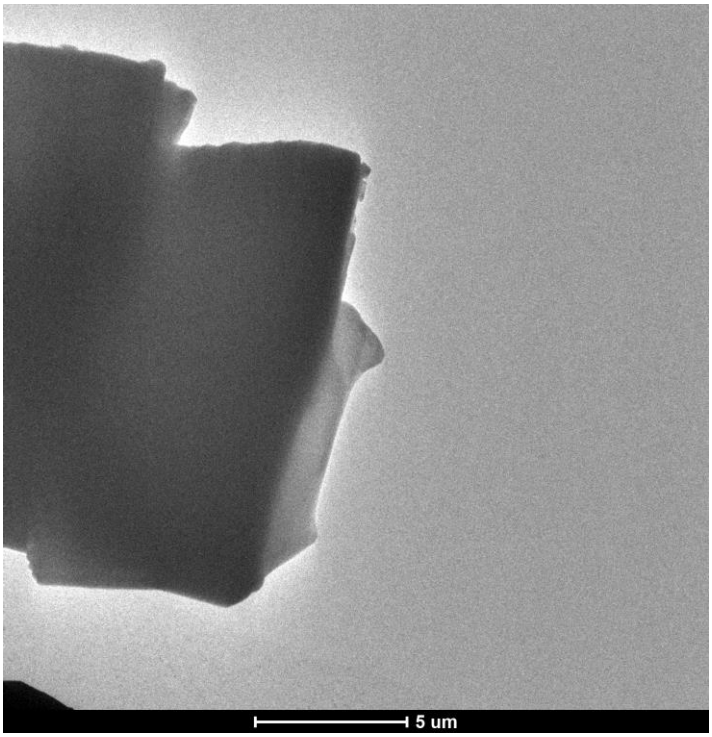


Figure 3.44 TEM image of **3.10** crystal at 2600x magnification.

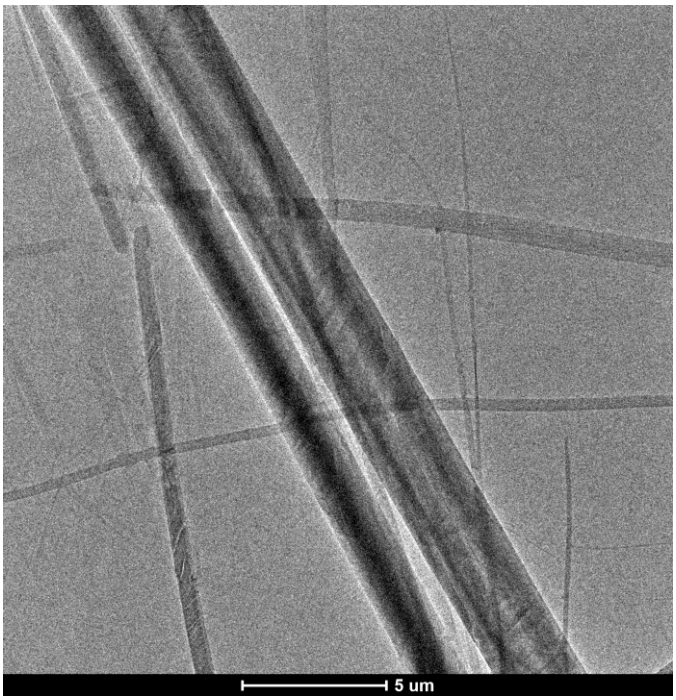


Figure 3.45 TEM image of **3.11** crystal at 2600x magnification.

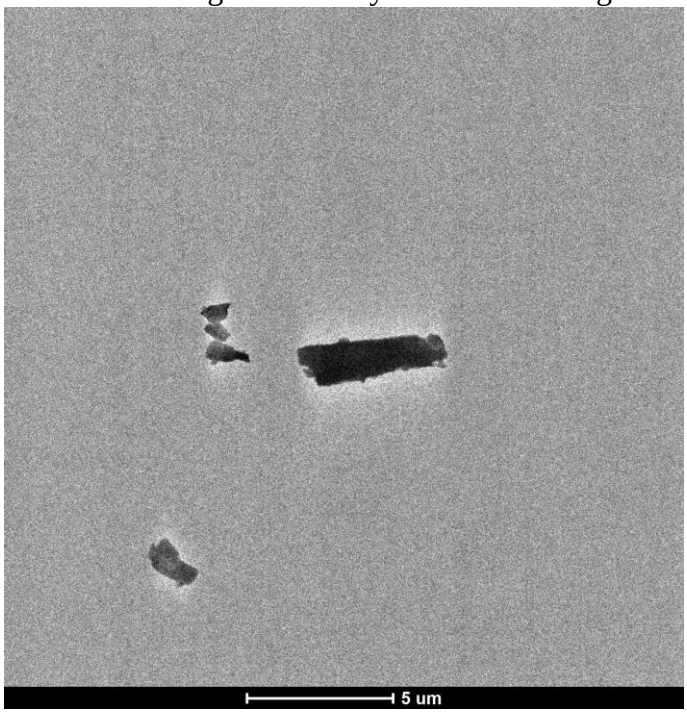


Figure 3.46 TEM image of **3.12** crystal at 2600x magnification.

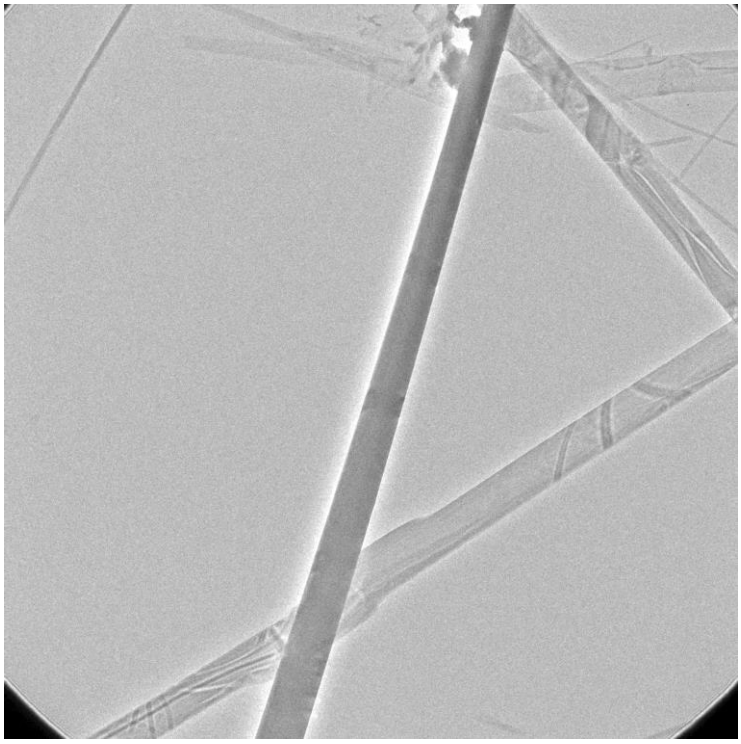


Figure 3.47 TEM image of **3.13** crystal at 2600x magnification.

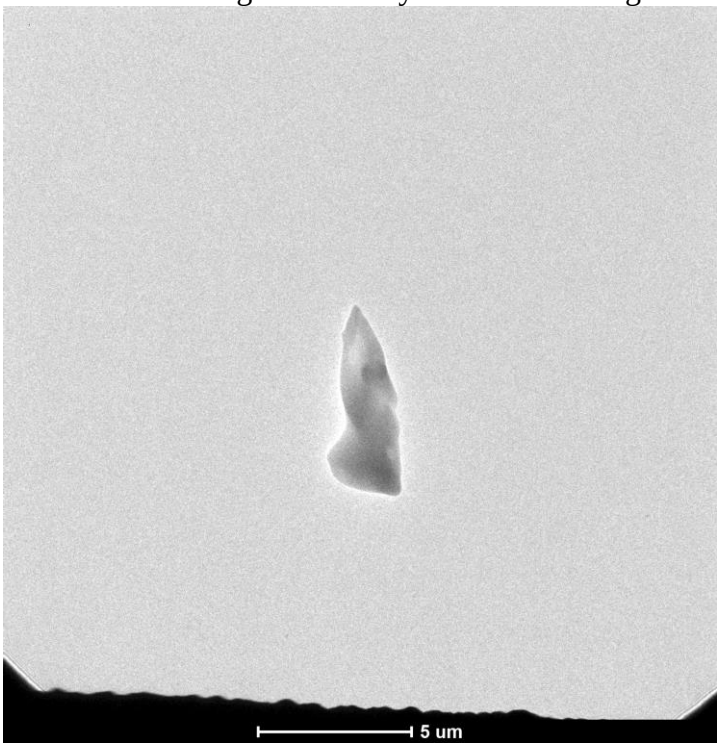


Figure 3.48 TEM image of **3.14** crystal at 2600x magnification.

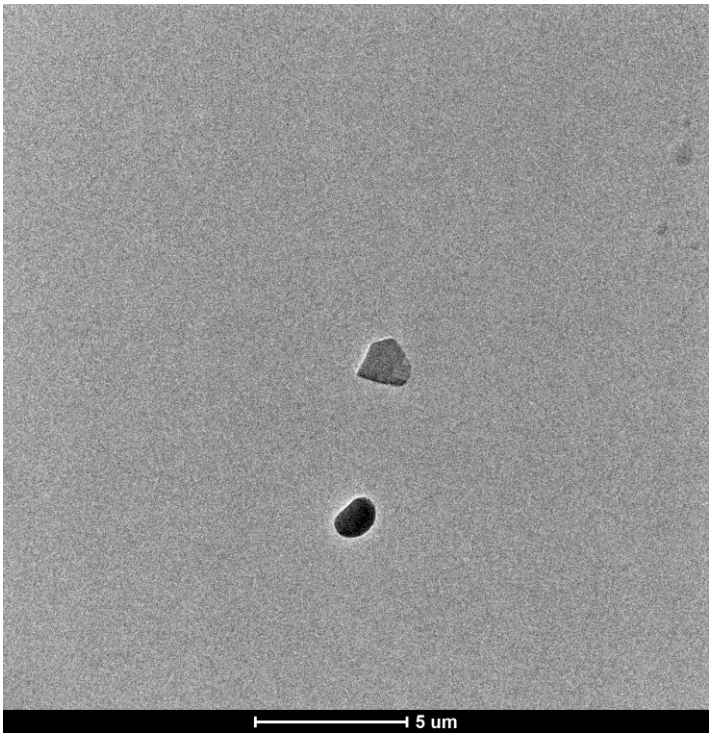


Figure 3.49 TEM image of **3.15** crystal at 2600x magnification.

3.8.9 Comparison of MicroED Data to X-ray Structures

Each **3.7** molecule from the asymmetric unit of two previously solved single crystal X-ray structures **AGX918A** and **AGX918B**, corresponding to solvated structures of **3.7** containing two molecules in the asymmetric unit with differing solvation states, were overlayed with the microED structure of **3.7**.

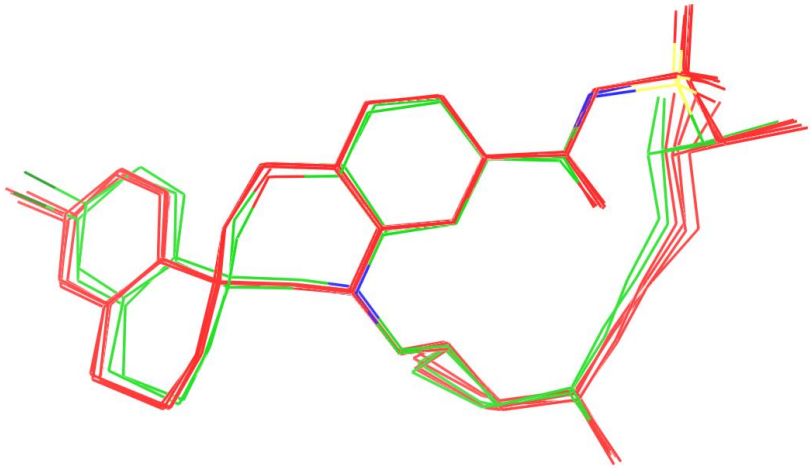


Figure 3.50 MicroED (green) and X-ray (red) crystallography data overlay of **3.7**. Any solvent molecules observed in the crystal structure were removed for this analysis.

Structure Name	Source	RMS	Maximum Difference
AGX918A_1	X-ray	0.0	0.0
AGX918A_2	X-ray	0.5645	2.5284
AGX918C_1	X-ray	0.1869	0.5034
AGX918C_2	X-ray	0.2704	0.8914
3.7_1	Electron	0.6067	2.6340
3.7_2	Electron	0.2539	0.4747

Table 3.2 RMS of structure overlay comparing one molecule of **AGX918A** to remaining five molecules in **AGX918A**, **AGX918B**, and **3.7**.

3.8.10 Automated Data Processing Procedure

Movie files were saved in a standardized format separated by underscores to allow for automated data processing. An example format is provided below:

samplename-mov1_960_0.3_3_cryo.ser

samplename-mov1 can be any name not including an underscore or special character. This will become the name of the folder containing processed data.

960 is the detector distance used in mm. This can be set to any value.

0.3 is the rotation speed of the stage, in °/s.

3 is the image integration time.

cryo can be any additional notes about the sample and can include underscores.

On a computer running Ubuntu Windows Subsystem for Linux with properly installed XDS suite and free ser2smv[21] data conversion file, "python3 auto_indexing.py" is called to run Python3.8 in a folder containing an executable copy of ser2smv, the python scripts, and the .ser movie files to be processed. Merging and solutions obtained subsequent to autoprocessing were done by the user.

3.8.11 Automated Data Processing Python Code

auto_indexing.py

```
"""
Written by Jessica Burch, jessburch@g.ucla.edu
This is a script to batch process individual MicroED datasets using XDS.
version: 03/01/2021
"""
import os
import shutil

def main():
    stats = open("stats.LP","w")
    stats.write("Data summary: ")
    files = os.listdir(".")
    if os.path.isfile("ser2smv") == True:
        for name in files:
            if name.endswith(".ser"):
                newname = name.split("_")
                path = os.getcwd()
                os.mkdir(path + "/" + str(newname[0]))
                os.mkdir(path + "/" + str(newname[0]) + "/images")
                os.mkdir(path + "/" + str(newname[0]) + "/auto_process")
                shutil.move(name,str(path + "/" + newname[0] + "/" + name))
                shutil.copyfile('xds_for_me.py', str(path + "/" + newname[0]
                + "/xds_for_me.py"))
                print("Setting up files for " + newname[0] + ".")
                os.chdir(path + "/" + str(newname[0]))
                os.system("python3 xds_for_me.py")
                os.system("rm xds_for_me.py")
                with open('auto_process/XSCALE.LP', 'r') as f:
                    lines = f.readlines()
                    for index, line in enumerate(lines):
                        if " ========== STATISTICS OF INPUT DATA SET ==========" in line:
                            t = lines[index-3]
                            t1 = t.split()
                            completeness = t1[4]
                            Roverall = t1[5]

                            l = lines[index-13]
                            l1 = l.split()
                            t = l1[5]
                            t2 = t[:-1]
                            if float(t2) < 100 and float(t2) > 0:
                                with open('xscale_report.LP','w') as f1:
```

```
            f1.write("\n" + str(l1[0]) + " " + str(l1[4])
                  + " " + str(l1[5]))
      l = lines[index-12]
      l1 = l.split()
      t = l1[5]
      t2 = t[:-1]
   if float(t2) < 100 and float(t2) > 0:
      with open('xscale_report.LP','a') as f1:
         f1.write("\n" + str(l1[0]) + " " + str(l1[4])
               + " " + str(l1[5]))
      l = lines[index-11]
      l1 = l.split()
      t = l1[5]
      t2 = t[:-1]
   if 1 < float(t2) < 100 and float(t2) > 0:
      with open('xscale_report.LP','a') as f1:
         f1.write("\n" + str(l1[0]) + " " + str(l1[4])
               + " " + str(l1[5]))
      l = lines[index-10]
      l1 = l.split()
      t = l1[5]
      t2 = t[:-1]
   if float(t2) < 100 and float(t2) > 0:
      with open('xscale_report.LP','a') as f1:
         f1.write("\n" + str(l1[0]) + " " + str(l1[4])
               + " " + str(l1[5]))
      l = lines[index-9]
      l1 = l.split()
      t = l1[5]
      t2 = t[:-1]
   if float(t2) < 100 and float(t2) > 0:
      with open('xscale_report.LP','a') as f1:
         f1.write("\n" + str(l1[0]) + " " + str(l1[4])
               + " " + str(l1[5]))
      l = lines[index-8]
      l1 = l.split()
      t = l1[5]
      t2 = t[:-1]
   if float(t2) < 100 and float(t2) > 0:
      with open('xscale_report.LP','a') as f1:
         f1.write("\n" + str(l1[0]) + " " + str(l1[4])
               + " " + str(l1[5]))
      l = lines[index-7]
      l1 = l.split()
      t = l1[5]
```

```
                t2 = t[:-1]
            if float(t2) < 100 and float(t2) > 0:
                with open('xscale_report.LP','a') as f1:
                    f1.write("\n" + str(l1[0]) + " " + str(l1[4])
                            + " " + str(l1[5]))
                l = lines[index-6]
                l1 = l.split()
                t = l1[5]
                t2 = t[:-1]
            if float(t2) < 100 and float(t2) > 0:
                with open('xscale_report.LP','a') as f1:
                    f1.write("\n" + str(l1[0]) + " " + str(l1[4])
                        + " " + str(l1[5]))
                l = lines[index-5]
                l1 = l.split()
                t = l1[5]
                t2 = t[:-1]
            if float(t2) < 100 and float(t2) > 0:
                with open('xscale_report.LP','a') as f1:
                    f1.write("\n" + str(l1[0]) + " " + str(l1[4])
                        + " " + str(l1[5]))
                l = lines[index-4]
                l1 = l.split()
                t = l1[5]
                t2 = t[:-1]
            if float(t2) < 100 and float(t2) > 0:
                with open('xscale_report.LP','a') as f1:
                    f1.write("\n" + str(l1[0]) + " " + str(l1[4])
                        + " " + str(l1[5]))
            else:
                with open('xscale_report.LP','a') as f1:
                    f1.write("\n ! low resolution data")

        with open('xscale_report.LP','r') as f2:
            xsc = f2.read()
        with open('auto_process/stats.LP','r') as f3:
            ind = f3.read()
        os.chdir("..")
        stats = open("stats.LP","a")
        stats.write("\n==========\n" + newname[0] + "\n" + ind + "\nXSCALE stats")
        stats.write("\n" + str(t1[4]) + " " + str(t1[5]) + "\n" + xsc)

if os.path.isfile("ser2smv") == False:
    print("Please add an executable copy of ser2smv to this folder!\n" +
        "Download from https://cryoem.ucla.edu/downloads/snapshots")
```

```
if __name__== "__main__":
    main()
```

xds_for_me.py

```
"""
Written by Jessica Burch, jessburch@g.ucla.edu
This is a script to automate indexing of MicroED data using XDS.
version: 03/01/2021
"""
import os
from subprocess import run
import random
"""
This portion reads your file name and converts .ser files to images.
Must have ser2smv program in the folder containing these scripts
This can be downloaded from https://cryoem.ucla.edu/downloads/snapshots
"""
def main():
    files = os.listdir(".")
    for name in files:
        if name.endswith(".ser"):
            newname = name.split("_")
            path = os.getcwd()
            newpath = str(path + "/images")
            os.chdir(newpath)
            #This is where data collection information such as
            conversion = str(path + "/../ser2smv -P 0.014 -B 2 -r " + newname[2]
                        + " -w 0.0251 -d " + newname[1] + " -E " + newname[3] +
                        " -M 200 -v -o " + newname[0] + "_###.img " + path
                        + "/" + name)
            print("Converting your .ser file to .img frames.")
            os.system(conversion + ' > summary.LP')
            global movname
            movname = newname[0]
            with open('summary.LP') as f1:
                lines = f1.readlines()
            with open('summary.LP', 'w') as f2:
                f2.writelines(lines[-15:])
            with open('summary.LP', 'r') as f:
                line = f.readline()
                for line in f:
                    if "+++" in line:
                        element = str.split(line)
```

```
            frame = str(element[2])
            print("You have " + frame + " images.")

path = os.getcwd()
newpath = str(path + "/../auto_process")
os.chdir(newpath)
f = open("XDS.INP","w+")
if os.path.isfile("*.LP") == True:
   os.remove("*.LP")
if os.path.isfile("*.XDS") == True:
   os.remove("*.XDS")
if os.path.isfile("*.HKL") == True:
   os.remove("*.HKL")
f.write("JOB= XYCORR INIT COLSPOT IDXREF DEFPIX INTEGRATE CORRECT" +
      "\n!JOB=DEFPIX INTEGRATE CORRECT")
#These are estimates for our beam center. The beam may be slightly off in
#actuality, but XDS does a good job of refining the beam center if the
#values are close.
x = str("1018")
y = str("1000")
osc = str(float(newname[3]) * float(newname[2]))
#This corrected distance value arises from indexing diffraction data of
#standard samples on our TEM and adjusting the detector distance value
#until these standards agree with the X-ray unit cell.
corrected_distance = float(newname[1]) * 0.943
data_path = str(path + "/" + newname[0])
f.write("\nORGX= " + x + " ORGY= " + y + " ! check these using adxv" +
      "\nDETECTOR_DISTANCE= " + str(corrected_distance) +
      "\nOSCILLATION_RANGE= " + osc + "\nX-RAY_WAVELENGTH= 0.0251000002")
f.write("\nNAME_TEMPLATE_OF_DATA_FRAMES=" + data_path + "_???.img" +
     "\nBACKGROUND_RANGE=1 10\n!DELPHI=15\n!SPACE_GROUP_NUMBER=0"
     + "\n!UNIT_CELL_CONSTANTS= 1 1 1 90 90 90")
#"res" is the high resolution cutoff based on our detector distance.
if newname[1] == str("1050"):
   res = str("0.8")
elif newname[1] == str("1100"):
   res = str("0.9")
elif newname[1] == str("960"):
   res = str("0.8")
elif newname[1] == str("850"):
   res = str("0.65")
elif newname[1] == str("1350"):
   res = str("1.1")
elif newname[1] == str("670"):
   res = str("0.45")
```

```
    elif newname[1] == str("420"):
        res = str("0.25")
    elif newname[1] == str("2200"):
        res = str("1.7")
    elif newname[1] == str("330"):
        res = str("0.15")

    #An important value to change based on your microscope is "ROTATION_AXIS"
    f.write("\nINCLUDE_RESOLUTION_RANGE= 40 " + res +
            "\nTEST_RESOLUTION_RANGE= 40 " + res + "\nTRUSTED_REGION=0.0 1.2"+
            "\nVALUE_RANGE_FOR_TRUSTED_DETECTOR_PIXELS=6000. 30000." +
            "! parameters for detector and beamline:" +
            "\nDETECTOR= ADSC MINIMUM_VALID_PIXEL_VALUE= 1 OVERLOAD=
65000" +
            "\nSENSOR_THICKNESS= 0.01" + "\nNX= 2048 NY= 2048 QX= 0.0280000009"
            + " QY= 0.0280000009" + "\nROTATION_AXIS=0 -1 0" +
            "\nDIRECTION_OF_DETECTOR_X-AXIS=1 0 0" +
            "\nDIRECTION_OF_DETECTOR_Y-AXIS=0 1 0" +
            "\nINCIDENT_BEAM_DIRECTION=0 0 1\nFRACTION_OF_POLARIZATION=0.98"
            + "\nPOLARIZATION_PLANE_NORMAL=0 1 0" +
            "\nREFINE(IDXREF)=CELL BEAM ORIENTATION AXIS ! DISTANCE" +
            "\nREFINE(INTEGRATE)= DISTANCE BEAM ORIENTATION ! AXIS CELL" +
            "\nREFINE(CORRECT)=CELL BEAM ORIENTATION AXIS ! DISTANCE !" +
            "\n\nDATA_RANGE= 1 " + str(element[2]) + "\nSPOT_RANGE= 1 "
            + str(element[2]))
    sp = "4"
    minpix = "7"
    f.write("\nSTRONG_PIXEL= " + sp +
"\nMINIMUM_NUMBER_OF_PIXELS_IN_A_SPOT= "
            + minpix + "\n!\n!")
    f.close()
    xds_out = open("XDS.LP", "w+")
    print("XDS is running...")
    run("xds", stdout= xds_out)

def autoprocessing():
    if os.path.isfile('X-CORRECTIONS.cbf') == False:
        xds_out = open("XDS.LP", "w+")
        print("XDS is running...")
        run("xds", stdout= xds_out)

    if os.path.isfile('XPARM.XDS') == False:
        for i in range(10):
            with open('XDS.INP') as f1:
                lines = f1.readlines()
```

```
        with open('XDS.INP', 'w') as f2:
           strong = random.randrange(3,9,1)
           mpix = random.randrange(4,9,1)
           f2.writelines(lines[:-4])
           f2.write("STRONG_PIXEL= " + str(strong) +
                 "\nMINIMUM_NUMBER_OF_PIXELS_IN_A_SPOT= " + str(mpix) +
                 "\n!\n!")
           f2.close()
           print("Screening new indexing values.")
           xds_out = open("XDS.LP", "w+")
           run("xds",stdout= xds_out)
        if os.path.isfile('XPARM.XDS') == True:
           if os.path.isfile('DEFPIX.LP') == False:
              with open('XDS.INP') as f1:
                 lines = f1.readlines()
              with open('XDS.INP', 'w') as f2:
                 f2.write("!JOB=XYCORR INIT COLSPOT IDXREF DEFPIX INTEGRATE
CORRECT"
                       + "\nJOB=DEFPIX INTEGRATE CORRECT\n")
                 f2.writelines(lines[2:])
                 f2.close()
                 print("Less than 70% of spots went through. Running with JOB= DEFPIX "
                     + "INTEGRATE CORRECT.")
                 xds_out = open("XDS.LP", "w+")
                 run("xds",stdout= xds_out)
                 if os.path.isfile('XPARM.XDS') == False:
                    print("Unable to autoprocess " + movname + "!")
                    exit()
                 else:
                    return autoprocessing()
           else:
              return autoprocessing()
     else:
        print("Unable to autoprocess " + movname + "!")
        f2.close()
        exit()

  if os.path.isfile('DEFPIX.LP') == False:
     with open('XDS.INP') as f1:
        lines = f1.readlines()
     with open('XDS.INP', 'w') as f2:
        f2.write("!JOB=XYCORR INIT COLSPOT IDXREF DEFPIX INTEGRATE
CORRECT"
              + "\nJOB=DEFPIX INTEGRATE CORRECT\n")
        f2.writelines(lines[2:])
```

```
            f2.close()
        print("Less than 70% of spots went through. Running with JOB= DEFPIX "
             + "INTEGRATE CORRECT.")
        xds_out = open("XDS.LP", "w+")
        run("xds",stdout= xds_out)
        return autoprocessing()

    if os.path.isfile("INTEGRATE.HKL") == False:
        with open('XDS.INP') as f1:
            lines = f1.readlines()
        with open('XDS.INP', 'w') as f2:
            f2.writelines(lines)
            f2.write("\nBEAM_DIVERGENCE= 0.03 BEAM_DIVERGENCE_E.S.D.= 0.003" +
            "\nREFLECTING_RANGE=1.0 REFLECTING_RANGE_E.S.D.= 0.2")
            f2.close()
            print("Adding beam divergence values to correct a common error.")
            xds_out = open("XDS.LP", "w+")
            run("xds",stdout= xds_out)
            return autoprocessing()
    if os.path.isfile("CORRECT.LP") == True:
        print ("Successful indexing!")
        return mosaicity()
def mosaicity():
    with open('XDS.INP') as f1:
        lines = f1.readlines()
    with open('XDS.INP', 'w') as f2:
        f2.write("!JOB=XYCORR INIT COLSPOT IDXREF DEFPIX INTEGRATE CORRECT"
                + "\nJOB=DEFPIX INTEGRATE CORRECT\n")
        f2.writelines(lines[2:-2])
    with open('INTEGRATE.LP', 'r') as l1:
        f2 = open('XDS.INP', 'a')
        line = l1.readline()
        for line in l1:
            if "BEAM_DIVERGENCE= " in line:
                f2.write(line)
            if "REFLECTING_RANGE=" in line:
                f2.write(line)
        f2.close()
        return iterate_opt()

def iterate_opt():
    with open('XDS.LP') as f1:
        lines = f1.readlines()
    with open('XDS.LP', 'w') as f2:
        f2.writelines(lines[-26:])
```

```
with open('XDS.LP', 'r') as f:
   line = f.readline()
   for line in f:
      if "     a        b          ISa" in line:
         next_line = f.readline()
         stats = str.split(next_line)
         Isa1 = float(stats[2])
         print("Isa: " + str(Isa1) + ". Testing new values now.")
xds_out = open("XDS.LP", "w+")
run("xds",stdout= xds_out)
with open('XDS.LP') as f1:
   lines = f1.readlines()
with open('XDS.LP', 'w') as f2:
   f2.writelines(lines[-26:])
with open('XDS.LP', 'r') as f:
   line = f.readline()
   for line in f:
      if "     a        b          ISa" in line:
         new_next_line = f.readline()
         new_stats = str.split(new_next_line)
         Isa2 = float(new_stats[2])
         print("Isa: " + str(Isa2))
      if "SPACE_GROUP_NUMBER=" in line:
         number = str.split(line)
         space_group = number[1]
      if "UNIT_CELL_CONSTANTS=" in line:
         cell = str.split(line)
         temp = cell[-6:]
         temp_str = str(temp).strip("]['")
         temp_str2 = temp_str.replace(",","")
         unit_cell = temp_str2.replace("'","")
Isa_change = abs(Isa2 - Isa1)
if Isa_change > 0.5:
   print("Optimizing beam divergence values.")
   return iterate_opt()
else:
   print("Optimized beam divergence values.")
   f = open('stats.LP','w')
   f.write(str(space_group) + "\n" + unit_cell)

   f.close
   print("Autoprocessing found space group " + str(space_group) + " and a unit cell of "
        + "\n" + unit_cell)
```

```
def scale_conv():
    xscale = open('XSCALE.INP','w')
    xscale_out = open("xscale.LP","w+")
    m = movname
    xscale.write("OUTPUT_FILE= " + m +".ahkl"+"\nINPUT_FILE= XDS_ASCII.HKL"
                 + "\nRESOLUTION_SHELLS= 10 8 5 3 2.3 2.0 1.7 1.5 1.3 " +
                 "1.2 1.1 1.0 0.90 0.80")
    xscale.close()
    run("xscale", stdout= xscale_out)
    print("Data scaled with XSCALE.")
    xdsconv_out = open("xdsconv.LP", "w+")
    xdsconv = open('XDSCONV.INP','w')
    xdsconv.write("INPUT_FILE= " + m + ".ahkl" + "\nOUTPUT_FILE= " +
                  m + ".hkl" + " SHELX" +
                  "\nGENERATE_FRACTION_OF_TEST_REFLECTIONS=0.10"
                  + "\nFRIEDEL'S_LAW=FALSE")
    xdsconv.close()
    run("xdsconv",stdout= xdsconv_out)
    print("Data converted for use in shelx!")




if __name__== "__main__":
    main()
    autoprocessing()
    scale_conv()
"""

"""
```

3.8.12 Data Collection of Atropisomeric 3.17

Dry powder sample analyzed as received at room temperature (see **3.8.1** for general room temperature microED procedure) and diffracted to sufficient resolution (< 1.2 Å) to provide a direct methods solution. Preliminary solution obtained from merging three datasets from three separate crystals with no user input other than molecular formula.

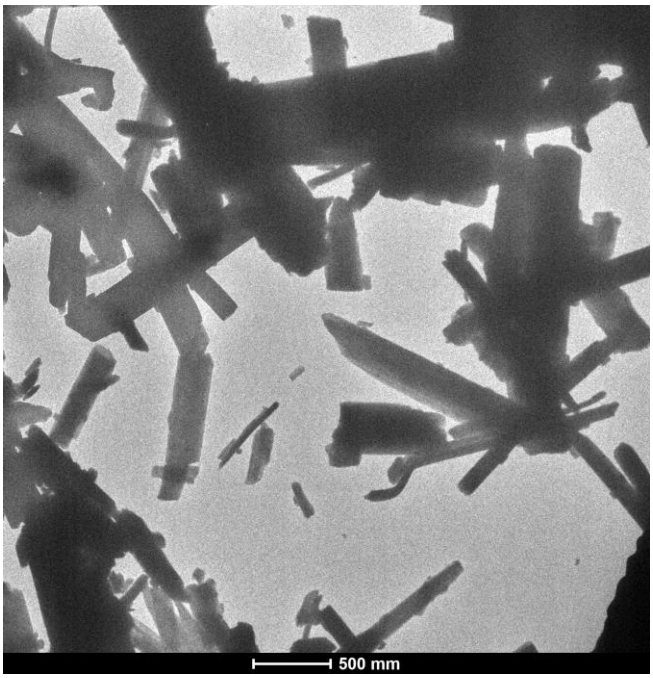


Figure 3.51 TEM image of **3.17** crystal at 2600x magnification.

3.8.13 Crystal Structure of Atropisomer 3.17

3.8.13.1 N-(1-(3-(4-chloro-3-(methylsulfonamido)-1-(2,2,2-trifluoroethyl)-1H-indazol-7-yl)-6-(3-methyl-3-(methylsulfonyl)but-1-yn-1-yl)pyridin-2-yl)-2-(3,5-difluorophenyl)ethyl)-2-((3bS,4aR)-5,5-difluoro-3-(trifluoromethyl)-3b,4,4a,5-tetrahydro-1H-cyclopropa[3,4]cyclopenta[1,2-c]pyrazol-1-yl)acetamide (3.17)

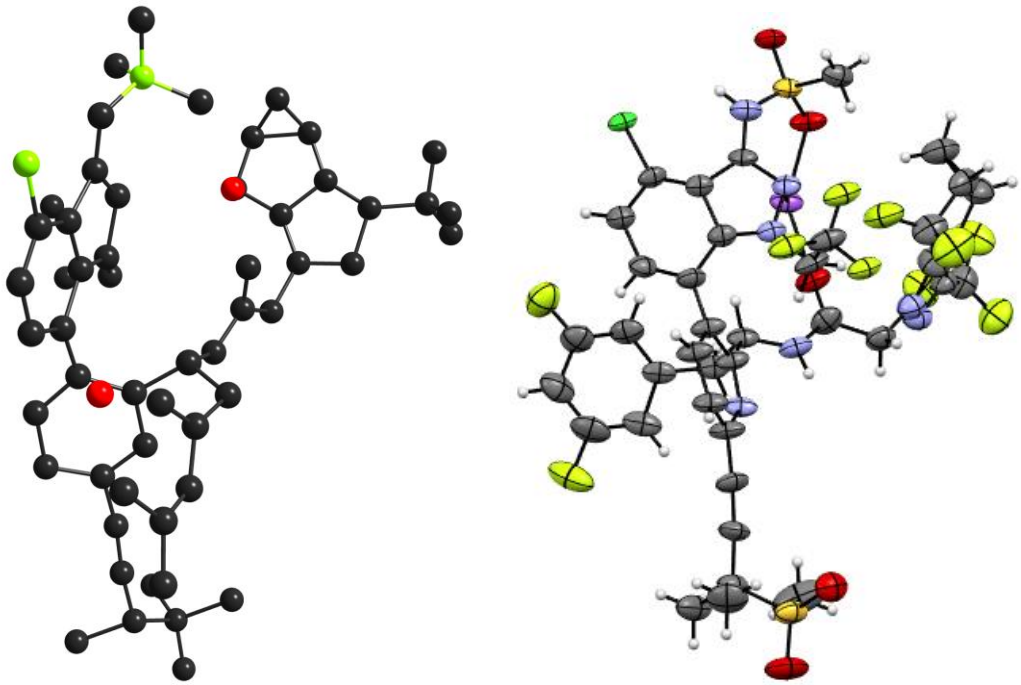


Figure 3.52 MicroED crystal structure of **3.17**. ORTEP diagram for the asymmetric unit of **3.17**. Thermal ellipsoids shown as shaded octants at 30% probability.

Crystal data and structure refinement for 3.17.

Empirical formula	$C_{39}H_{32}ClF_{10}N_7O_5S_2Na$
Formula weight	991.27
Crystal Habit	acicular
Crystal size	50 – 90 μm

Sample Lot Number	6207-02-AC-1P

Data Collection

Type of instrument	Talos F200C
Wavelength	0.0215 Å
Data collection temperature	–178(4) °C
Unit cell dimensions	a = 8.6500(10) b = 19.580(2) c = 26.050(4)
Volume	4,412.0(10)
Z	4
Crystal system	Orthorhombic
Space group	$P2_12_12_1$
Density (calculated)	1.492 Mg/m^3
F(000)	178
Measured reflections	26556
Reflections with I > 2σ(*I*)	5213
Resolution	0.95 Å
Completeness	95.7%
Index ranges	$9 \leq h \leq -9$, $19 \leq k \leq -19$, $26 \leq l \leq -26$

Structure solution and Refinement

Structure solution program	SHELXD (Uson & Sheldrick, 1999)
Primary solution method	Direct methods
Secondary solution method	Difference Fourier map
Hydrogen placement	Geometric positions
Structure refinement program	SHELXL-2018/3 (Sheldrick, 2018)
Refinement method	Full matrix least-squares on F^2
Data / restraints / parameters	5213 / 989 / 586

Treatment of hydrogen atoms	Riding
Goodness-of-fit on F^2	1.662
Final R indices [I>2s(I), 3563 reflections]	R1 = 0.1502, *w*R2 = 0.3357
R indices (all data)	R1 = 0.1928, *w*R2 = 0.3510
Type of weighting scheme used	Sigma
Weighting scheme used	$w=1/s^2(Fo^2)$
Max shift/error	0.034
Average shift/error	0.004
Largest diff. peak and hole	0.15 and -0.19 e.$Å^{-3}$

Special Refinement Details

Refinement of F2 against ALL reflections. The weighted R-factor (wR) and goodness of fit (S) are based on F2 , conventional R-factors (R) are based on F, with F set to zero for negative F2 . The threshold expression of F2 > 2s(F2) is used only for calculating R-factors(gt) etc. and is not relevant to the choice of reflections for refinement. R-factors based on F2 are statistically about twice as large as those based on F, and R-factors based on ALL data will be even larger.

All esds (except the esd in the dihedral angle between two l.s. planes) are estimated using the full covariance matrix. The cell esds are taken into account individually in the estimation of esds in distances, angles and torsion angles; correlations between esds in cell parameters are only used when they are defined by crystal symmetry. An approximate (isotropic) treatment of cell esds is used for estimating esds involving l.s. planes.

3.8.14 Data Collection of Natural Product Compound

Quantifoil holey-carbon EM grids were placed in a dram vial with purified **3.23** and shaken lightly. Residual compound was removed by tapping lightly against the surface of a filter paper. All diffraction data was collected on FEI Tecnai F200C electron microscope with an operating voltage of 200 keV, corresponding to a wavelength of 0.025 Å, using Gatan 626 cryo-holder under cryogenic temperature (100 K). During data acquisition, the crystal of interest was isolated using a selected area aperture and continuously rotated at a rate of -0.3º/s over a tilt range of 50–100º. Continuous rotation diffraction data was recorded using rolling shutter mode with a Ceta-D CMOS 4k x 4k camera, integrating at a rate of 3 s per frame and binning by 2 to

produce final images of 2k x 2k. Diffraction movies saved as SER files were converted to SMV format using ser2smv software as described previously.[3,21] Frames were indexed and integrated in XDS.[1] Data from four crystals were scaled and merged together using XSCALE to produce the final data set.[2] Finally, intensities were converted to SHELX format using XDSCONV. The structure of **3.23** was solved *ab initio* using direct methods in SHELXD and refined with SHELXL in ShelXle.[4–7] All non-hydrogen atoms were refined anisotropically, and hydrogen atoms were placed using the riding model. Crystallographic information files (CIF) for compound **3.23** have been deposited at the Cambridge Crystallographic Data Center (Deposition Number: 2020283).

3.8.15 Crystal Structure of Natural Product Compound

3.8.15.1 (2R,3R,4S,5R,6R)-2-((benzoyloxy)methyl)-6-(2,6,8-trioxo-1,2,6,7,8,9-hexahydro-3H-purin-3-yl)tetrahydro-2H-pyran-3,4,5-triyl tribenzoate (3.23)

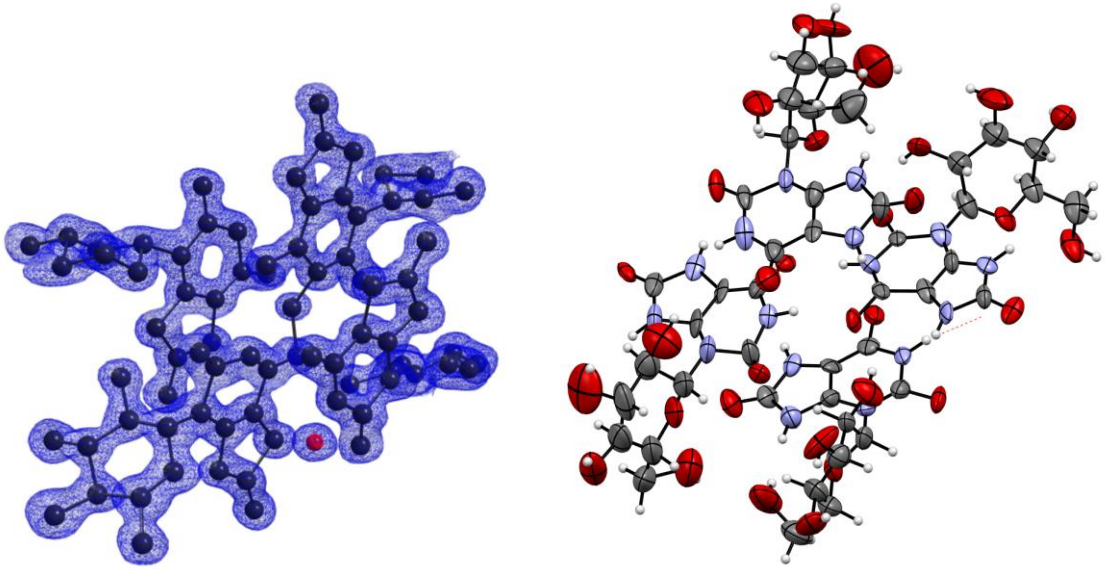


Figure 3.53 MicroED crystal structure of **3.23.** Initial direct methods solution of **3.23** (left) with electron density map (F_{obs}) contoured at 1.41 e Å^{-3} and ORTEP diagram of refined **3.23** (right). Thermal ellipsoids shown as shaded octants at 30% probability.

Crystal data and structure refinement for 3.23.

Empirical formula	$C_{11}H_{14}N_4O_8$
Formula weight	330.25

Data Collection

Type of instrument	Talos F200C
Wavelength	0.0215 Å
Data collection temperature	96(4) K
Unit cell dimensions	a = 8.990(9)
	b = 9.730(2)
	c = 14.16(3)
	α = 93.05(9)
	β = 94.62(5)
	γ = 108.54(5)
Volume	1166.5(4)
Z	4
Crystal system	Triclinic
Space group	P1
Density (calculated)	1.881 Mg/m^3
F(000)	239
Measured reflections	8144
Reflections with I > 2σ(*I*)	2122
Resolution	1.0 Å
Completeness	87.1%
Index ranges	8 ≤ h ≤ -8, 9 ≤ k ≤
	-9, 11 ≤ l ≤ -11

Structure Solution and Refinement

Structure solution program	SHELXD (Uson & Sheldrick, 1999)
Primary solution method	Direct methods
Secondary solution method	Difference Fourier map
Hydrogen placement	Geometric positions
Structure refinement program	SHELXL-2018/3 (Sheldrick, 2018)
Refinement method	Full matrix least-squares on F^2
Data / restraints / parameters	4210 / 914 / 798
Treatment of hydrogen atoms	Riding
Goodness-of-fit on F^2	1.300
Final R indices [I>2s(I), 975 reflections]	R1 = 0.1299, *w*R2 = 0.2990
R indices (all data)	R1 = 0.1586, *w*R2 = 0.3223
Type of weighting scheme used	Sigma
Weighting scheme used	$w=1/s^2(Fo^2)$
Max shift/error	0.163
Average shift/error	0.001
Largest diff. peak and hole	0.15 and -0.15 e.Å^{-3}

Special Refinement Details

Refinement of F^2 against ALL reflections. The weighted R-factor (wR) and goodness of fit (S) are based on F^2, conventional R-factors (R) are based on F, with F set to zero for negative F^2. The threshold expression of $F^2 > 2s(F^2)$ is used only for calculating R-factors(gt) etc. and is not relevant to the choice of reflections for refinement. R-factors based on F^2 are statistically about twice as large as those based on F, and R-factors based on ALL data will be even larger.

All esds (except the esd in the dihedral angle between two l.s. planes) are estimated using the full covariance matrix. The cell esds are taken into account individually in the estimation of esds in distances, angles and torsion angles; correlations between esds in cell parameters are only used when they are defined by crystal symmetry. An approximate (isotropic) treatment of cell esds is used for estimating esds involving l.s. planes.

3.8.14 Supplementary Notes & References

(1) Kabsch, W. *Acta Cryst.* **2010**, *D66*, 125–132.

(2) Kabsch, W. *Acta Cryst.* **2010,** *D66*, 133–144.

(3) Hattne, J., *et al*. *Acta Cryst.* ***2015***, *71*, 353–360.

(4) Sheldrick, G. M. A short history of SHELX. *Acta Cryst.* **2008**, *A64*, 112–122.

(5) Sheldrick, G. M. *Acta Cryst.* **2015** *A71*, 3–8.

(6) Sheldrick, G. M. *Acta Cryst.* **2015**, *C71*, 3–8.

(7) Hübschle, C. B., Sheldrick, G. M. & Dittrich, B. *J. Appl. Cryst.* **2011,** *44*, 1281–1284.

(8) *PCT Int. Appl.* **2016**, WO 2016187308 A1 20161124.

(9) *PCT Int. Appl.* **2018**, WO 2018097945 A1 20180531.

(10) PCT/US2015/047472, WO 2016033486 A1.

(11) *PCT Int. Appl.* **2018**, WO 2018093576 A1 20180524.

(12) *PCT Int. Appl.* **2018**, WO 2018093577 A1 20180524.

(13) *PCT Int. Appl.* **2018**, WO 2018093579 A1 20180524.

(14) *PCT Int. Appl.* **2018**, WO 2018097944 A1 20180531.

(15) *PCT Int. Appl.* **2018**, WO 2018097945 A1 20180531.

(16) *PCT Int. Appl.* **2019**, WO 2019006231 A1 20190103.

(17) *PCT Int. Appl.* **2016**, WO 2016141035 A1.

(18) *J. Org. Chem.* **2014**, *79*, 3684–3687.

(19) *J. Med. Chem.* **2014**, *57*, 9796–9810.

(20) *J. Med. Chem.* **2018**, *61*, 453–461.

(21) Program ser2smv obtained from https://cryoem.ucla.edu/downloads/snapshots.

CHAPTER FOUR

Optimization of Microcrystal Electron Diffraction Data Collection and Processing Routines through Scripted Automation

4.1 Abstract

The work detailed in this chapter focuses on the automation of data processing and collection for microcrystal electron diffraction (microED), in an effort to make the technique more accessible and practical for routine structure elucidation of small molecules. The development of Python scripts to adapt XDS, a diffraction indexing program designed for X-ray crystallography, for automated microED data is described. Additionally, automated data collection routines for incident diffraction screening of microcrystalline particles through an efficient, automated process utilizing image recognition is described.

4.2 Introduction

Using the popularization of NMR spectroscopy and cryoEM as examples, it is clear that advancement of reliable, automated tools is critical to the wide-spread use of emerging analytical techniques.[1–3] Despite the increasing number of unique reported small molecule compounds, only ~50,000 crystal structures per year have been deposited into the Cambridge Structural Database over the past decade.[4,5] The incorporation of more straightforward operation and automated data processing into the microED workflow would lower the barrier for non-specialists to utilize the technique. These developments would not only allow chemists to obtain crystal structures of a subset of compounds unable to be solved by X-ray crystallography, but also render the study of microcrystalline compounds a routine part of organic synthesis.

At present, microED faces a number of unique challenges for routine implementation. Manual data collection and processing can be repetitive, time-consuming, and hit-or-miss; that is to say while some compounds can provide data capable of structural solution within minutes, others require collection of hundreds of datasets and exhaustive crystallization attempts (See Chapter 3). There exists a handful of programs that automate portions of the collection and processing of microED data, as is briefly described in Chapter 1; however, these programs treat data processing and collection separately and these two separate stages have not yet been fully integrated into a single, complete workflow.[6–9] Given the variability between samples, we propose that a dynamic process, in which processed data enters a feedback loop to inform active data collection, is critical for efficient sample analysis. This is particularly important when automating data collection of crystals that display preferred orientation or have low symmetry.

4.3 Automation of Data Processing

Small molecule samples most commonly crystallize in low symmetry space groups belonging to triclinic, monoclinic, and orthorhombic Laue groups.[10] Samples that produce crystals in these lower symmetry crystal systems require wide sampling of reciprocal space for a complete dataset. This presents a challenge for studies by microED: while an X-ray diffractometer can collect 360° of data in ideal cases, physical limitations of transmission electron microscopes (TEMs) typically allow for a maximum tilt range of ~130°. This means that multiple datasets often must be combined to reach high enough dataset completeness to obtain an *ab initio* solution, which is not typically encountered in X-ray crystallographic studies.[11] Additionally, microcrystals may lie on a TEM grid with a preferred orientation due to its physical dimensions, or rapidly degrade under the electron beam, further hindering collection of a full dataset.[12,13] Much of this non-ideal

diffraction behavior cannot be reliably determined by automated procedures until after the data has gone through indexing and merging.

In our ideal scenario, a user will insert their prepared TEM grid into the microscope and incident diffraction screening will begin (Figure 4.1). If the incident diffraction data is determined to be promising, a continuous rotation dataset would be collected and processed. A feedback loop

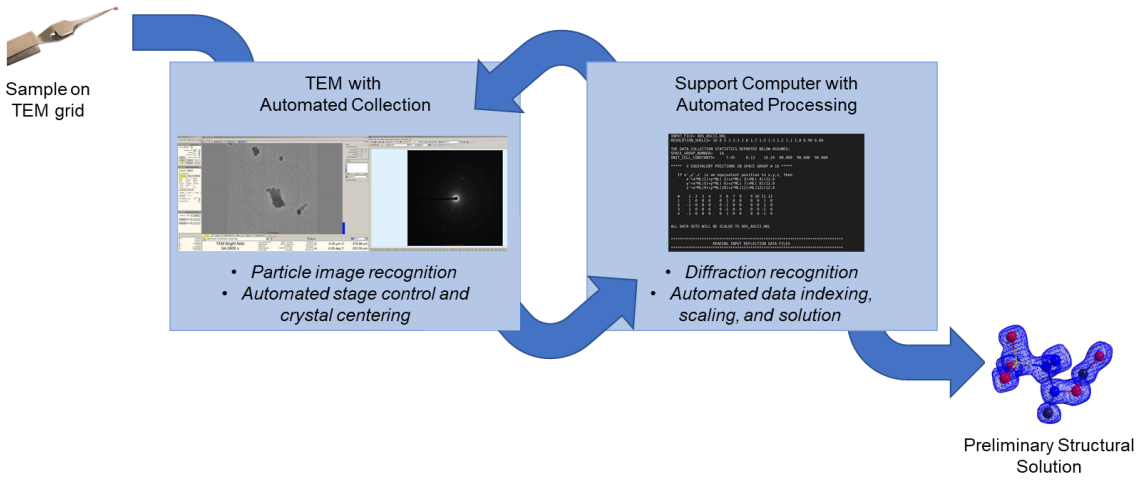


Figure 4.1 Overview of feedback loop proposed for fully automated microED structural solutions.

would then be generated, in which data will be collected and combined until a solution is obtained. Inserting and removing a sample into the TEM, particularly at cryogenic temperatures, can be a time-consuming operation, taking over an hour per sample. It is therefore critical to have reliable automation of both data collection and data processing to ensure sufficient data has been collected before a sample is removed. The data collection would then be halted after a structure has been obtained.

There are multiple reports outlining settings that can be modified from the X-ray crystallographic program, XDS, to manually process electron diffraction data. After conversion of continuous rotation diffraction movies into individual images, the XDS program performs indexing and data reduction through a series of simple commands.[9,14–16] The status of the

processing can easily be monitored through generation, or lack thereof, of a series of data files and error messages. Applying our experience from manually processing hundreds of datasets, we developed an automation regime capable of catching and correcting common errors in data processing to ultimately return crystallographic statistics from raw diffraction datasets.

To remove the need for human input in the data processing pipeline, we needed to make a series of assumptions about values a user would insert after careful evaluation of the diffraction images; one of these values is the location of the beam center. Typically, users visualize diffraction images and utilize diagonally-related diffraction spots to measure the beam center using a visualization program such as adxv (Figure 4.2).[17]

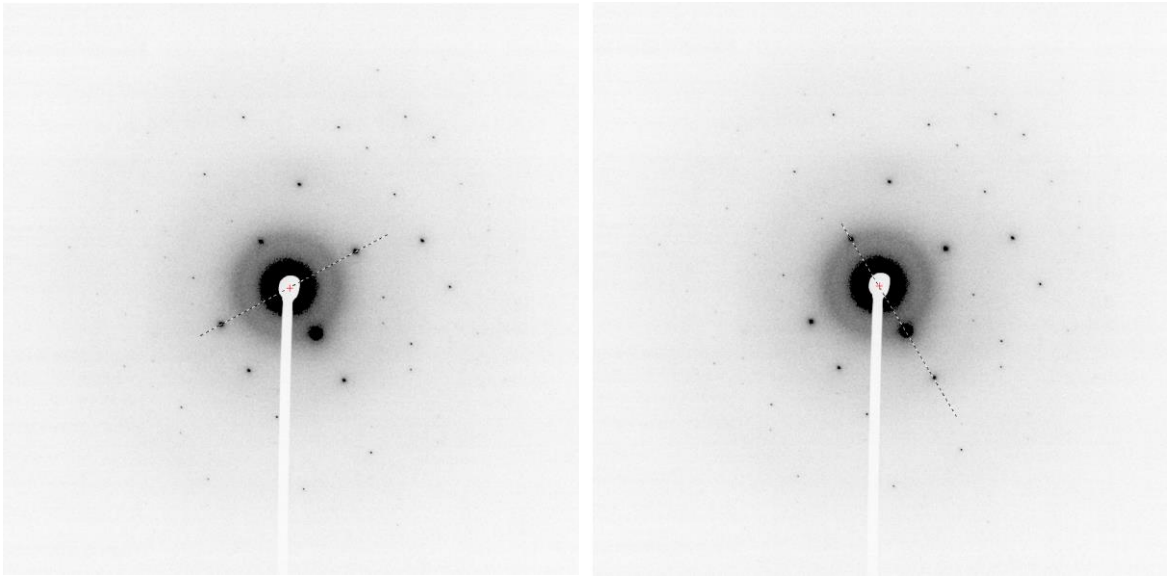


Figure 4.2 Manual measurement of beam center (red crosshairs) using the line tool (dashed line) for a microED diffraction image in adxv.

As long as this value is relatively close to the true beam center, XDS can automatically refine this value in the "IDXREF" step while determining symmetry of indexed diffraction spots.[14] Utilizing a Thermo Fisher Talos F200C with a beam stop designed for use in electron diffraction, paired

with a Ceta-D detector, the beam center can only deviate approximately ± 25 pixels (for a 2k x 2k image) in each direction while remaining behind the beam stop. We have found that inputting the center of the beam stop as the beam center is sufficient for XDS to then perform fine refinement of the beam position, providing a simple workaround for the previous need to visualize and measure the diffraction pattern prior to processing.

Another approximation made is for "STRONG_PIXEL" and "NUMBER_OF_PIXELS_IN_A_SPOT". The ideal values are sample- and dose-dependent: they relate to how "bright" the diffraction is relative to background noise ("STRONG_PIXEL") and the minimum number of bright pixels needed to be considered a diffraction spot ("NUMBER_OF_PIXELS_IN_A_SPOT"). An underestimation of these values can lead to indexing of background noise, and an overestimation may lead to few or no diffraction spots being detected. Manual determination of these values is done by directly reading the pixel values from diffraction spots and may be necessary if standard values do not work, typically due to poor data quality. This can be done visually, using a program such as adxv to magnify a diffraction spot of

interest, as shown in Figure 4.3.[17] To avoid this visual estimation, the automated program inputs an initial guess for the "STRONG_PIXEL" and

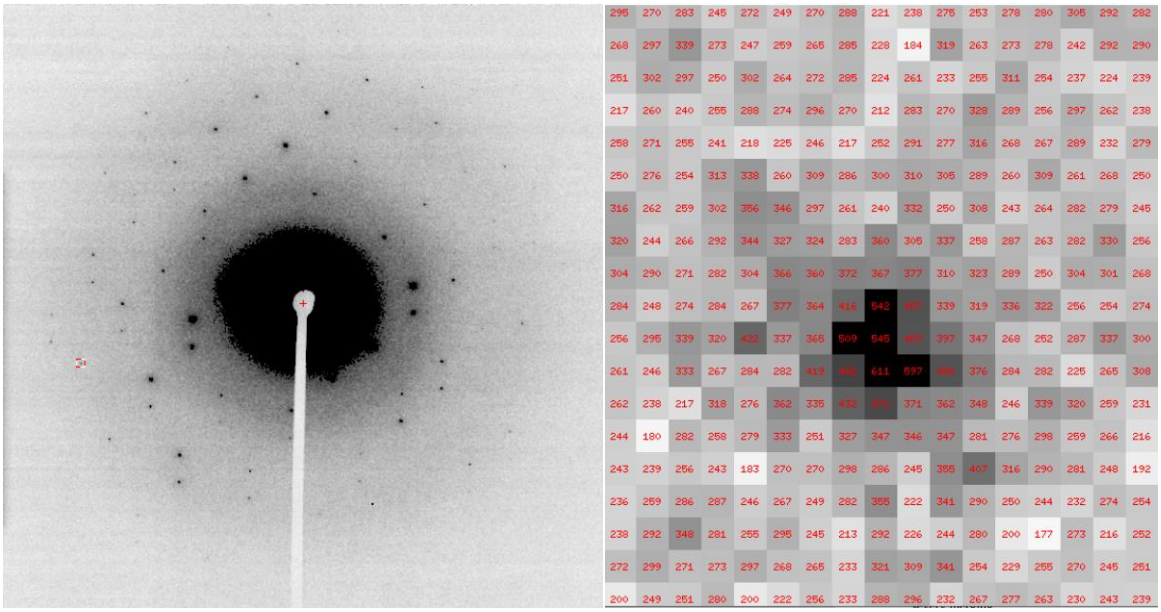


Figure 4.3 TEM image of microED diffraction pattern with a region selected (left) and magnified diffraction spot displaying pixel values in adxv to determine indexing values (right).

"MINIMUM_NUMBER_OF_PIXELS_IN_A_SPOT", with values of 4 and 7 respectively, based on the typical values that lead to successful indexing of our datasets. If XDS fails to index the dataset using these values, the automation script will loop through random attempts of changing these parameters until successful indexing occurs. In our experience, if a diffraction dataset is too poor in quality to be indexed by XDS, it will fail in this step; therefore, the program will terminate processing if ten failed indexing attempts have been completed at this step.

Resolving these roadblocks to automation and translating our knowledge solving microED datasets into Python code led to development of two scripts able to be launched using a single, simple command in conjunction with a freely available data conversion program.[18] This can be utilized from a Mac terminal, Linux system, or Windows Subsystem for Linux to convert, index, and scale batches of raw diffraction datasets. The program returns statistics of the dataset and

generates files needed for submission to SHELX for structural solution in organized subfolders (Figure 4.4).[19–21] Every microED dataset generated from compounds resolved in Chapter 3, 161 datasets in total, were successfully indexed using this automated program. The average time for indexing on a 64-core server is ~1 minute per dataset. Examples of diffraction datasets from our data library that failed analysis are outlined in Figure 4.5. It is worth nothing that manual re-processing of these datasets, typically by removing portions of the full dataset, could lead to successful indexing; however, the crystallographic statistics remained very poor, as indicated by low resolution and high error values.

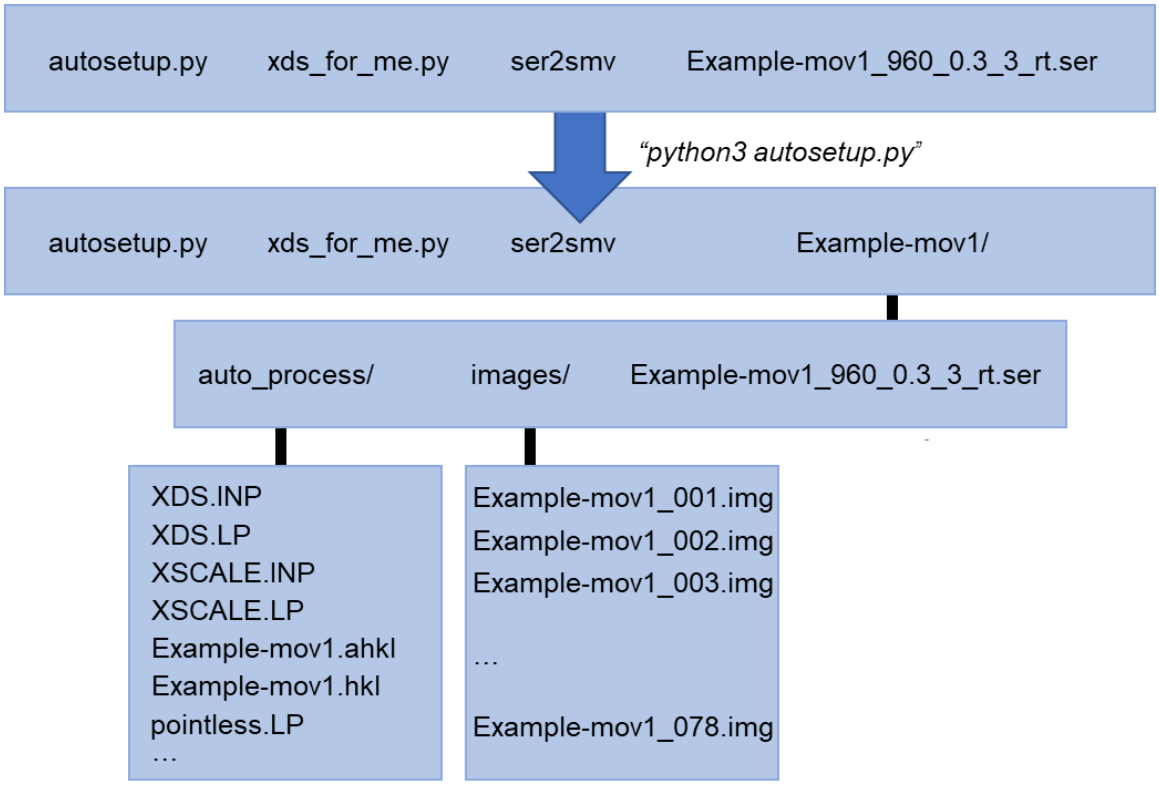


Figure 4.4 Example of input **a**, execution **b**, and output **c** from Python program for automatically processing single datasets.

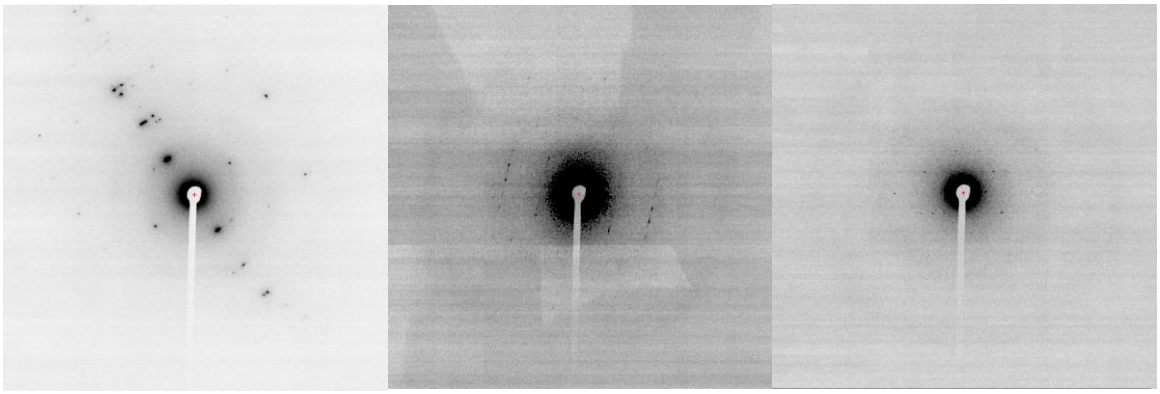


Figure 4.5 Images taken from diffraction movies that failed indexing through development of our automated diffraction program.

After successfully developing a program to process individual datasets, we turned our attention to automating the combination of multiple processed datasets utilizing XSCALE to generate structural solutions.[15] This process, which is often referred to as dataset merging, involves scaling of multiple datasets together for solution as if it were one complete dataset. In manual microED structural solutions, a user will typically test many dataset permutations by trial-and-error in XSCALE, and submit promising merged sets to SHELX for an attempt at a solution: typically those with < 1.2 Å resolution, above 80% completeness, and the lowest merged error statistics possible.

At first, we sought to replicate this process by developing a Python code to generate each possible permutation. This is effective for a small number of datasets: for example, five movies would have 325 unique combinations, and all can be screened in under three minutes. This quickly becomes impractical for evaluating large numbers of individual movies, where 10 movies would have almost 10 million possible permutations and would require months to screen exhaustively. A recent microED report from our laboratory describes a scenario in which over 200 individual diffraction movies were collected for the eventual solution of a complex natural product.[22] While this presents an astonishingly large number of permutations to explore, the final resolved structure

came from merging only four datasets. This is broadly seen to be true from reported microED structures: most structural solutions originate from combination of four or fewer individual diffraction datasets. Knowing this information, we wrote a program to evaluate all permutations of four movies or less, improving the speed at which merges can be tested. We then developed additional programs to subject each of these merges to SHELXT and to sort the resultant data. Finally, we wrapped these individual components into a user-friendly graphical user interface (GUI, Figure 4.6). In straightforward cases, a user inputs diffraction files and molecular formula, and a structural solution will be produced.

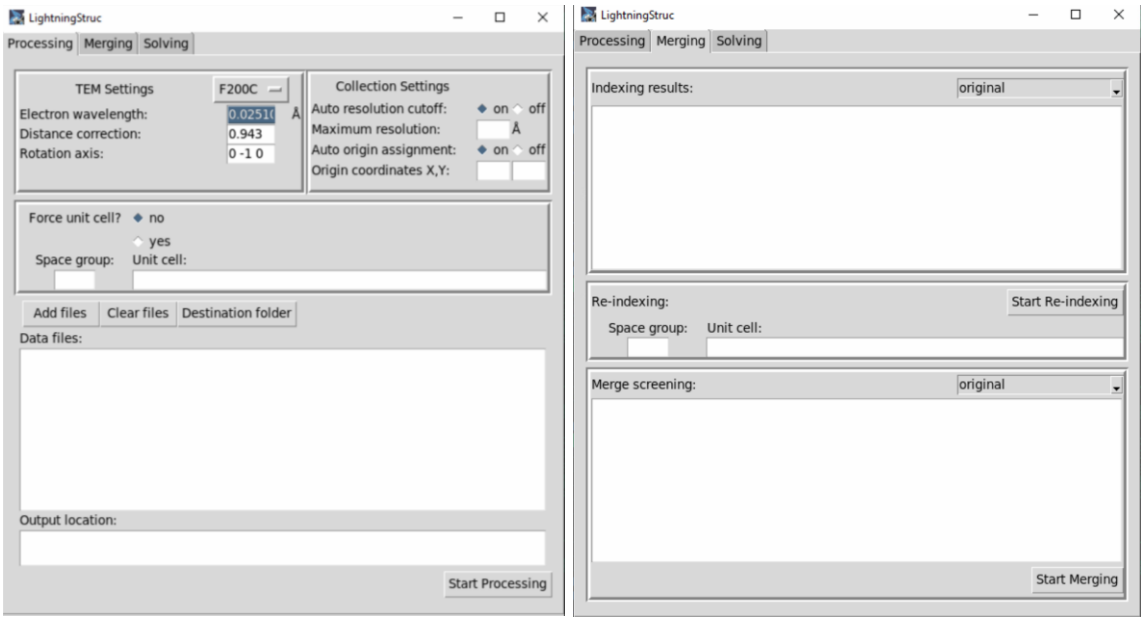


Figure 4.6 GUI developed to automate batch indexing and processing of microED datasets.

4.4 Automation of Data Collection

After developing this simple, standalone programs to process microED data, we turned to development of automated data collection tools with the eventual goal of combining these efforts into a single, comprehensive microED solution platform. The open-source program currently closest to achieving this for continuous rotation electron diffraction is Instamatic/InsteaDMatic

developed by the Zou group; however, these programs are developed to operate through the Gatan DigitalMicrograph software suite and cannot directly communicate with ThermoFisher Scientific camera software, TIA.[7–9,23] The ThermoFisher Scientific Ceta-D camera is sufficient for small molecule diffraction studies and its affordable price point makes it an attractive option for widespread adoption of microED. With this in mind, we begam development of automation compatible with this platform.

Our software relies upon communication with the ThermoFisher Scientific (TFS) Talos F200C microscope server through TEMScripting and Advanced Scripting features. We first sought to develop a simple tool to automate Low Dose, the TFS tool commonly utilized in microED to rapidly switch between imaging and diffraction modes. Quickly, we ran into difficulties: the beam stop, critical to protecting the Ceta-D detector from the condensed incident electron beam, was not accessible through TEMScripting or AdvancedScripting. To circumvent this, we employed a Python library called PyAutoGUI to map the microscope user interface (MUI) and identify coordinate values that correspond to desired commands (ex: inserting the beam stop, removing the beam stop, switching between Low Dose modes).[24] Simply creating a map of the MUI and incorporating PyAutoGUI commands into our workflow allowed us to develop a tool to reliably collect selected area incident diffraction patterns through our standard Low Dose methods using three clicks of a GUI (Figure 4.7). After development of this screening tool, we incorporated it into a script that can perform particle recognition, similar to Instamatic, for a comprehensive incident diffraction screening program.

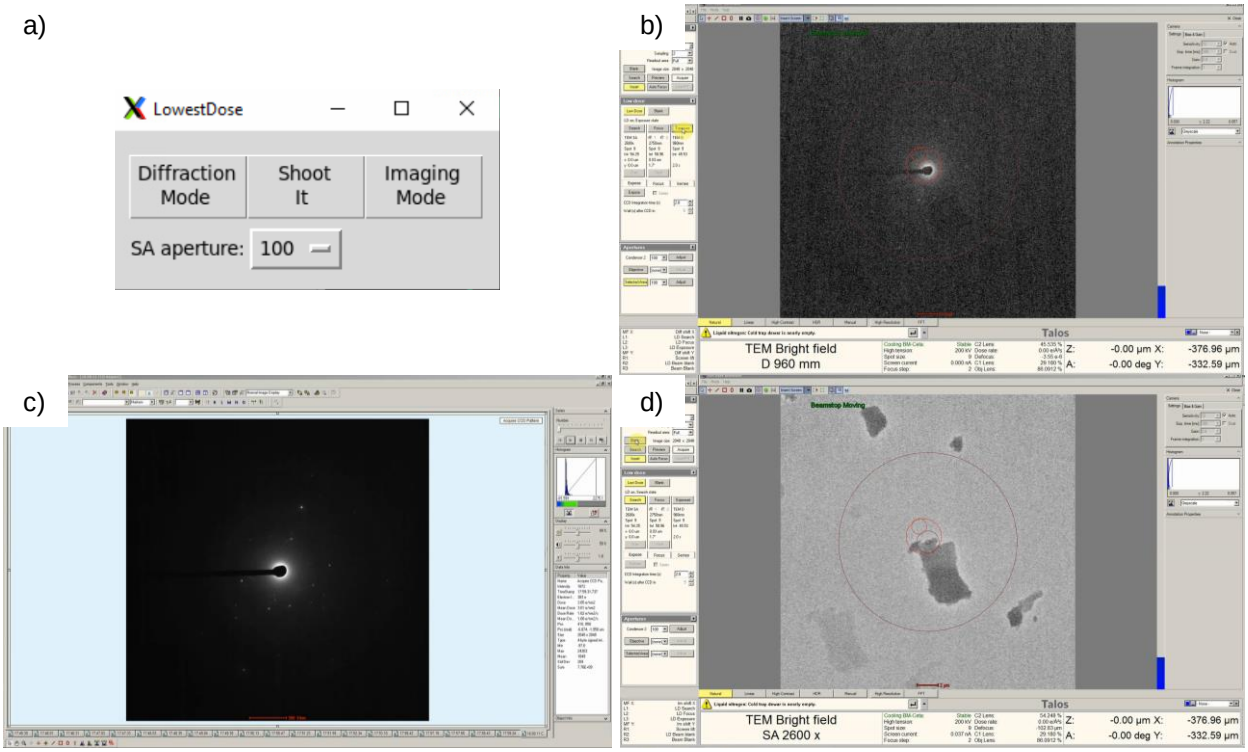


Figure 4.7 GUI **a** developed to automate collection of incident diffraction set-up **b**, recording of diffraction image **c**, and returning to imaging mode utilizing low dose **d**.

During incident diffraction screening, a user identifies particles of interest in imaging mode and quickly collects a single diffraction image to evaluate its crystallinity and likelihood of providing a useful microED dataset. Particularly when loading solvated suspensions of crystals onto a grid, the distribution of particles is often not uniform, and it is therefore important to image the entire grid. Currently available automated diffraction programs perform this imaging at a magnification of around 2600-3400x, but fully imaging a grid at this magnification is impractical. This is because the field of view at this magnification is 0.04% of the viewable grid area. Obtaining an image overview of the entire grid at low magnification (380x) is more routinely employed as "maps" or "montages" in TEM techniques such as cryoEM and cryoET, through use of programs like TFS Maps or SerialEM.[25,26] While it is relatively straightforward for a microscope operator to

identify promising particles at low magnification and collect data from them at higher magnification, we needed to develop an efficient strategy to perform this using automation.

We developed a technique to perform image recognition using the Python library OpenCV at low magnification (380x), giving us coordinates to regions of the grid that contain particles (Figure 4.8).[27] We found that, due to TEM hardware limitations, these coordinates do not

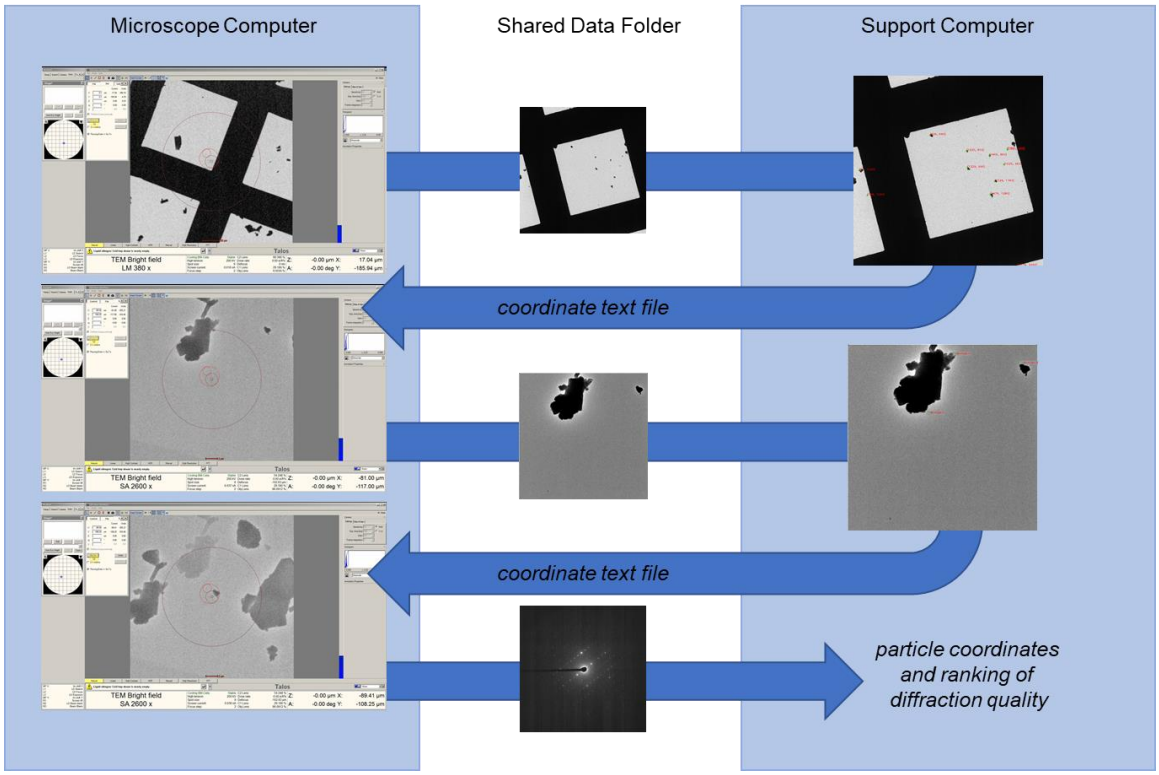


Figure 4.8 Automation workflow for collection and analysis of particles at low magnification, realignment at increased magnification, and recording and ranking of diffraction patterns from particles.

consistently correspond to the same values at higher magnification (2600x). This held true even when they were carefully recorded manually to exclude errors from the image recognition process. The values were deviated from the true center an average of 7.3 μm as determined manually from 20 datapoints from differing crystals, with poor consistency in which direction they were offset. This variability was overcome in our automation program by quickly taking a second image at

2600x magnification and readjusting to accurately center the crystal for selected area diffraction (Figure 4.8).

Multiple particle recognition strategies developed from the OpenCV library were evaluated by analyzing a standard grid containing 50% sodium chloride and 50% carbamazepine. The rate of successful diffraction was compared to manual incident diffraction screening of the same standard sample and found success rates approaching that of an experienced TEM user (Figure 4.9).

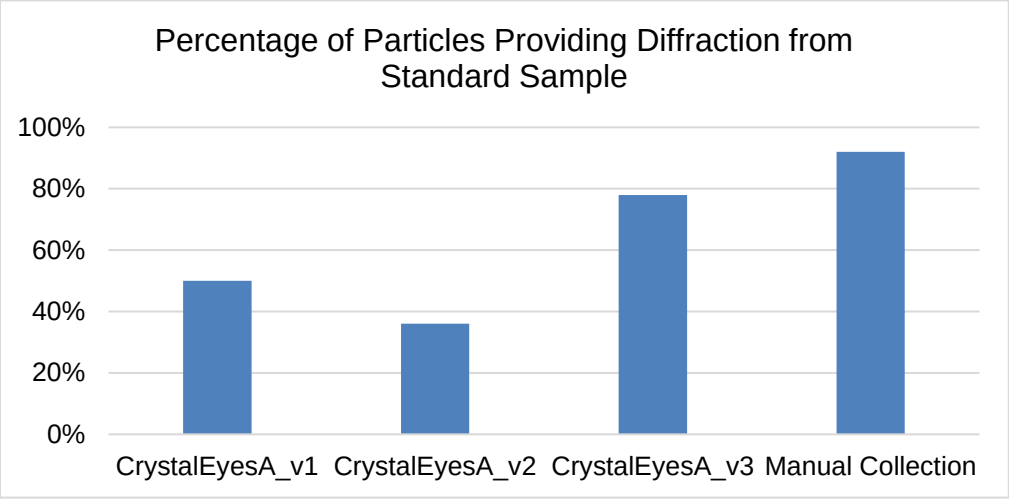


Figure 4.9 Evaluation of image recognition scripts to manual collection based on percentage of particles that provide diffraction.

4.5 Conclusion

In summary, this chapter explored the development of microED data processing and data collection automation regimes to ultimately increase accessibility and allow for routine structural elucidation of small molecules. We have developed a series of processing scripts can transform most microED datasets into indexed files ready for merging and structural solution. Through use of a GUI, in straightforward cases, a user can obtain a structural solution in a matter of minutes by inputting the appropriate files and estimated molecular formula. We also developed tools to

improve the tedious process of screening an entire grid for incident diffraction through use of image recognition and TEMScripting. We expect this work will lay the groundwork for future efforts combining these tools into a fully automated microED pipeline, lowering the barrier to entry for chemists to perform such analyses. Much in the same way that NMR came to the forefront of structural characterization through a combination of fundamental and applied methods development, we expect that microED will similarly become a commonplace method for unambiguous structural assignment of small molecules.

4.6 Notes and References

(1) Gunther, H. NMR Spectroscopy: Basic Principles, Concepts, and Applications in Chemistry. *John Wiley & Sons: New York*, **2013**.

(2) Shen, P. S. The 2017 nobel prize in chemistry: cryo-EM comes of age. *Anal. Bioanal. Chem.* **2018**, *410*, 2053–2057.

(3) Schorb, M.; Haberbosch, I.; Hagen, W. J. H.; Schwab, Y.; Mastronarde, D. N. Software tools for automated transmission electron microscopy *Nat. Methods* **2019**, *16*, 471–477.

(4) Cambridge Structural Database. *CSD Entries: Summary Statistics*, **2021**. https://www.ccdc.cam.ac.uk/CCDCStats/Stats.

(5) Lipkus, A. H.; Watkins, S. P.; Gengras, K.; McBride, M. J.; Wills, T. J. Recent changes in the scaffold diversity of organic chemistry as seen in the CAS registry. *J. Org. Chem.* **2019**, *84*, 13948–13956.

(6) de la Cruz, J. M.; Martynowycz, M.; Hattne, J.; Gonen, T. MicroED data collection with SerialEM *Ultramicroscopy* **2019**, *201*, 77–80.

(7) Smeets, S.; Zou, X.; Wan, W. Serial electron crystallography for structure determination and phase analysis of nanocrystalline materials. *J. Appl. Crystallogr.* **2018**, *51*, 1262–1273.

(8) Roslova, M.; Smeets, S.; Wing, B.; Thersleff, T.; Xu, H.; Zou, X. InsteaDMatic: towards cross-platform automated continuous rotation electron diffraction *J. Appl. Crystallogr.* **2020**, *53*, 1217–1224.

(9) Wang, B.; Zou, X.; Smeets, S. Automated serial rotation electron diffraction combined with cluster analysis: an efficient multi-crystal workflow for structure determination *IUCrJ.* **2019**, *6*, 854–867.

(10) Cambridge Structural Database. *CSD Crystal System Statistics*, **2021**. https://www.ccdc.cam.ac.uk/CCDCStats/Stats.

(11) Dauter, Z. Collection of X-ray diffraction data from macromolecular crystals. *Methods Mol Biol.* **2017**, *1607*, 165–184.

(12) (a) Barth, M. *et al.* Estimation of missing cone data in three-dimensional electron microscopy. *Scanning Microsc. Suppl.* **1988**, *2*, 277–284. (b) Glaeser, R. M. *et al.* Three-dimensional reconstructions from incomplete data: interpretability of density maps at 'atomic' resolution. *Ultramicroscopy* **1989**, *27*, 307–318. (c) Ford, R. C.; Holzenburg, A. Electron crystallography of biomolecules: mysterious membranes and missing cones. *Trends Biochem. Sci.* **2008**, *1*, 38–43.

(13) Hattne, J.; Shi, D.; Glynn, C.; Zee, C.-T.; Gallagher-Jones, M.; Martynowycz, M. W.; Rodriguez, J. A.; Gonen, T. Analysis of Global and Site-Specific Radiation Damage in Cryo-EM *Structure* **2018**, *5*, 759–766.

(14) Kabsch, W. Xds. *Acta Crystallogr.* **2010**, *D66*, 125–132.

(15) Kabsch, W. Integration, scaling, space-group assignment and post-refinement. *Acta Crystallogr.* **2010,** *D66*, 133–144.

(16) Hattne, J.; Reyes, F. E.; Nannenga, B. L.; Shi, D.; de la Cruz, M. J.; Leslie, A. G. W.; Gonen, T. MicroED data collection and processing. *Acta Crystallogr.*, Sect. A: Found. Adv. **2015**, *71*, 353−360.

(17) Arvai, A. J. Adxv – a program to display x-ray diffraction images. **1994**, https://www.scripps.edu/tainer/arvai/adxv.html.

(18) Program ser2smv obtained from https://cryoem.ucla.edu/downloads/snapshots.

(19) (a) Sheldrick, G. M. A short history of SHELX. *Acta Cryst.* **2008**, *A64*, 112–122.(b) Sheldrick, G. M. SHELXT – Integrated space-group and crystal-structure determination. *Acta Cryst.* **2015** *A71*, 3–8.

(20) Sheldrick, G. M. Crystal structure refinement with SHELXL. *Acta Cryst.* **2015**, *C71*, 3–8.

(21) Hübschle, C. B., Sheldrick, G. M. & Dittrich, B. ShelXle: A Qt graphical user interface for SHELXL. *J. Appl. Cryst.* **2011,** *44*, 1281–1284.

(22) Kim, L. J.; Ohashi, M.; Zhang, Z.; Tan, D.; Asay, M.; Cascio, D.; Rodriguez, J. A.; Tang, Y.; Nelson, H. M. Prospecting for natural products by genome mining and microcrystal electron diffraction. *Nat. Chem. Biol.* **2021**, *17*, 872−877.

(23) Gatan DigitalMicrograph can be accessed from: https://www.gatan.com/products/tem-analysis/gatan-microscopy-suite-software

(24) Bradski G. The OpenCV Library. *Dr Dobb's Journal of Software Tools*, **2000**.

(25) ThermoFisher Scientific Maps can be accessed from: https://www.thermofisher.com/us/en/home/electron-microscopy/products/software-em-3d-vis/maps-software.html.

(26) Mastronarde, D. N. Automated electron microscope tomography using robust prediction of specimen movements. *J. Struct. Biol.* **2005**, *152*, 36–51.

(27) Sweigart, A. PyAutoGUI, **2019**. https://github.com/asweigart/pyautogui.

(28) Evans, P. Scaling and assessment of data quality. *Acta Cryst.* **2006**, *D62*, 72–82.

(29) Winn, M. D. *et al*. Overview of the CCP4 suite and current developments. *Acta. Cryst.* **2011**, *D67*, 235–242.

4.7 Experimental Section

4.7.1 Automated MicroED Diffraction Processing Programs

4.7.1.1 Automated MicroED Diffraction Processing Program Procedure

Movie files were saved in a standardized format separated by underscores to allow for automated data processing. An example format is provided below:

samplename-mov1_960_0.3_3_cryo.ser

samplename-mov1 can be any name not including an underscore or special character. This will become the name of the folder containing processed data.

960 is the detector distance used in mm. This can be set to any value.

0.3 is the rotation speed of the stage, in °/s.

3 is the image integration time.

cryo can be any additional notes about the sample and can include underscores.

On a computer running Ubuntu Windows Subsystem for Linux with properly installed XDS suite, free ser2smv[18] data conversion file, and pointless program as part of the CCP4 data suite[28,29], "python3 auto_indexing.py" is called to run Python3.8 in a folder containing an executable copy of ser2smv, the python scripts, and the .ser movie files to be processed.

4.7.1.2 xds_for_me_0302.py

```
"""
Jessica Burch
script for microED data processing using XDS
version: 03/02/2020
"""
import os
from subprocess import run
import random
import string

# This portion reads your file name and converts .ser files to images.
def main():
    files = os.listdir(".")
    for name in files:
        if name.endswith(".ser"):
            newname = name.split("_")
            path = os.getcwd()
            newpath = str(path + "/images")
            os.chdir(newpath)
            conversion = str(path + "/../ser2smv -P 0.014 -B 2 -r " + newname[2]
                        + " -w 0.0251 -d " + newname[1] + " -E " + newname[3] +
                        " -M 200 -v -o " + newname[0] + "_###.img " + path
                        + "/" + name)
            print("Converting your .ser file to .img frames. Hold on!")
            os.system(conversion + ' > summary.LP')
            global movname
            movname = newname[0]
            with open('summary.LP') as f1:
                lines = f1.readlines()
            with open('summary.LP', 'w') as f2:
                f2.writelines(lines[-15:])
            with open('summary.LP', 'r') as f:
                line = f.readline()
                for line in f:
                    if "+++" in line:
                        element = str.split(line)
                        frame = str(element[2])
                        print("You have " + frame + " images.")

    path = os.getcwd()
    newpath = str(path + "/../auto_process")
    os.chdir(newpath)
    f = open("XDS.INP","w+")
```

```
    if os.path.isfile("*.LP") == True:
        os.remove("*.LP")
    if os.path.isfile("*.XDS") == True:
        os.remove("*.XDS")
    if os.path.isfile("*.HKL") == True:
        os.remove("*.HKL")
    f.write("JOB= XYCORR INIT COLSPOT IDXREF DEFPIX INTEGRATE CORRECT" +
            "\n!JOB=DEFPIX INTEGRATE CORRECT")
    x = str("1018")
    y = str("1000")
    osc = str(float(newname[3]) * float(newname[2]))
    corrected_distance = float(newname[1]) * 0.943
    data_path = str(path + "/" + newname[0])
    f.write("\nORGX= " + x + " ORGY= " + y + " ! check these using adxv" +
            "\nDETECTOR_DISTANCE= " + str(corrected_distance) +
            "\nOSCILLATION_RANGE= " + osc + "\nX-RAY_WAVELENGTH= 0.0251000002")
    f.write("\nNAME_TEMPLATE_OF_DATA_FRAMES=" + data_path + "_???.img" +
           "\nBACKGROUND_RANGE=1 10\n!DELPHI=15\n!SPACE_GROUP_NUMBER=0"
           + "\n!UNIT_CELL_CONSTANTS= 1 1 1 90 90 90")

    if newname[1] == str("1050"):
        res = str("0.8")
    elif newname[1] == str("1100"):
        res = str("0.9")
    elif newname[1] == str("960"):
        res = str("0.85")
    elif newname[1] == str("850"):
        res = str("0.65")
    elif newname[1] == str("1350"):
        res = str("1.1")
    elif newname[1] == str("670"):
        res = str("0.45")
    elif newname[1] == str("420"):
        res = str("0.25")
    elif newname[1] == str("2200"):
        res = str("1.7")
    elif newname[1] == str("330"):
        res = str("0.15")

    f.write("\nINCLUDE_RESOLUTION_RANGE= 40 " + res +
            "\nTEST_RESOLUTION_RANGE= 40 " + res + "\nTRUSTED_REGION=0.0 1.2"+
            "\nVALUE_RANGE_FOR_TRUSTED_DETECTOR_PIXELS=6000. 30000." +
            "! parameters for detector and beamline:" +
            "\nDETECTOR= ADSC MINIMUM_VALID_PIXEL_VALUE= 1 OVERLOAD=
65000" +
            "\nSENSOR_THICKNESS= 0.01" + "\nNX= 2048 NY= 2048 QX= 0.0280000009"
```

```
        + " QY= 0.0280000009" + "\nROTATION_AXIS=0 -1 0" +
        "\nDIRECTION_OF_DETECTOR_X-AXIS=1 0 0" +
        "\nDIRECTION_OF_DETECTOR_Y-AXIS=0 1 0" +
        "\nINCIDENT_BEAM_DIRECTION=0 0 1\nFRACTION_OF_POLARIZATION=0.98"
        + "\nPOLARIZATION_PLANE_NORMAL=0 1 0" +
        "\nREFINE(IDXREF)=CELL BEAM ORIENTATION AXIS ! DISTANCE" +
        "\nREFINE(INTEGRATE)= DISTANCE BEAM ORIENTATION ! AXIS CELL" +
        "\nREFINE(CORRECT)=CELL BEAM ORIENTATION AXIS ! DISTANCE !" +
        "\n\nDATA_RANGE= 1 " + str(element[2]) + "\nSPOT_RANGE= 1 "
        + str(element[2]))
   sp = "4"
   minpix = "7"
   f.write("\nSTRONG_PIXEL=                 "                 +                 sp                 +
"\nMINIMUM_NUMBER_OF_PIXELS_IN_A_SPOT= "
        + minpix + "\n!\n!")
   f.close()
   xds_out = open("XDS.LP", "w+")
   print("XDS is running...")
   run("xds", stdout= xds_out)

def autoprocessing():
   if os.path.isfile('X-CORRECTIONS.cbf') == False:
      xds_out = open("XDS.LP", "w+")
      print("XDS is running...")
      run("xds", stdout= xds_out)

   if os.path.isfile('XPARM.XDS') == False:
      for i in range(10):
         with open('XDS.INP') as f1:
            lines = f1.readlines()
         with open('XDS.INP', 'w') as f2:
            strong = random.randrange(3,9,1)
            mpix = random.randrange(4,9,1)
            f2.writelines(lines[:-4])
            f2.write("STRONG_PIXEL= " + str(strong) +
                  "\nMINIMUM_NUMBER_OF_PIXELS_IN_A_SPOT= " + str(mpix) +
                  "\n!\n!")
            f2.close()
            print("Screening new indexing values for you.")
            xds_out = open("XDS.LP", "w+")
            run("xds",stdout= xds_out)
         if os.path.isfile('XPARM.XDS') == True:
            if os.path.isfile('DEFPIX.LP') == False:
               with open('XDS.INP') as f1:
                  lines = f1.readlines()
               with open('XDS.INP', 'w') as f2:
```

```
                f2.write("!JOB=XYCORR INIT COLSPOT IDXREF DEFPIX INTEGRATE
CORRECT"
                        + "\nJOB=DEFPIX INTEGRATE CORRECT\n")
                f2.writelines(lines[2:])
                f2.close()
                print("Less than 70% of spots went through. Running with JOB= DEFPIX "
                     + "INTEGRATE CORRECT.")
                xds_out = open("XDS.LP", "w+")
                run("xds",stdout= xds_out)
                if os.path.isfile('XPARM.XDS') == False:
                   print("Unable to autoprocess " + movname + "!")
                   exit()
                else:
                   return autoprocessing()
             else:
                return autoprocessing()
       else:
          print("Unable to autoprocess " + movname + "!")
          f2.close()
          exit()

    if os.path.isfile('DEFPIX.LP') == False:
       with open('XDS.INP') as f1:
          lines = f1.readlines()
       with open('XDS.INP', 'w') as f2:
          f2.write("!JOB=XYCORR INIT COLSPOT IDXREF DEFPIX INTEGRATE
CORRECT"
                 + "\nJOB=DEFPIX INTEGRATE CORRECT\n")
          f2.writelines(lines[2:])
          f2.close()
       print("Less than 70% of spots went through. Running with JOB= DEFPIX "
           + "INTEGRATE CORRECT.")
       xds_out = open("XDS.LP", "w+")
       run("xds",stdout= xds_out)
       return autoprocessing()

    if os.path.isfile("INTEGRATE.HKL") == False:
       with open('XDS.INP') as f1:
          lines = f1.readlines()
       with open('XDS.INP', 'w') as f2:
          f2.writelines(lines)
          f2.write("\nBEAM_DIVERGENCE= 0.03 BEAM_DIVERGENCE_E.S.D.= 0.003" +
          "\nREFLECTING_RANGE=1.0 REFLECTING_RANGE_E.S.D.= 0.2")
          f2.close()
          print("Adding beam divergence values to correct a common error.")
          xds_out = open("XDS.LP", "w+")
```

```
            run("xds",stdout= xds_out)
            return autoprocessing()
    if os.path.isfile("CORRECT.LP") == True:
        print ("Successful indexing!")
        return mosaicity()
def mosaicity():
    with open('XDS.INP') as f1:
        lines = f1.readlines()
    with open('XDS.INP', 'w') as f2:
        f2.write("!JOB=XYCORR INIT COLSPOT IDXREF DEFPIX INTEGRATE CORRECT"
                 + "\nJOB=DEFPIX INTEGRATE CORRECT\n")
        f2.writelines(lines[2:-2])
    with open('INTEGRATE.LP', 'r') as l1:
        f2 = open('XDS.INP', 'a')
        line = l1.readline()
        for line in l1:
            if "BEAM_DIVERGENCE= " in line:
                f2.write(line)
            if "REFLECTING_RANGE=" in line:
                f2.write(line)
        f2.close()
        return iterate_opt()

def iterate_opt():
    with open('XDS.LP') as f1:
        lines = f1.readlines()
    with open('XDS.LP', 'w') as f2:
        f2.writelines(lines[-26:])
    with open('XDS.LP', 'r') as f:
        line = f.readline()
        for line in f:
            if "     a        b          ISa" in line:
                next_line = f.readline()
                stats = str.split(next_line)
                Isa1 = float(stats[2])
                print("Isa: " + str(Isa1) + ". Testing new values now.")
    xds_out = open("XDS.LP", "w+")
    run("xds",stdout= xds_out)
    with open('XDS.LP') as f1:
        lines = f1.readlines()
    with open('XDS.LP', 'w') as f2:
        f2.writelines(lines[-26:])
    with open('XDS.LP', 'r') as f:
        line = f.readline()
        for line in f:
            if "     a        b          ISa" in line:
```

```
            new_next_line = f.readline()
            new_stats = str.split(new_next_line)
            Isa2 = float(new_stats[2])
            print("Isa: " + str(Isa2))
         if "SPACE_GROUP_NUMBER=" in line:
            number = str.split(line)
            space_group = number[1]
         if "UNIT_CELL_CONSTANTS=" in line:
            cell = str.split(line)
            temp = cell[-6:]
            temp_str = str(temp).strip("]['")
            temp_str2 = temp_str.replace(",","")
            unit_cell = temp_str2.replace("","")
   Isa_change = abs(Isa2 - Isa1)
   if Isa_change > 0.5:
      print("I'm trying to optimize beam divergence values.")
      return iterate_opt()
   else:
      print("Optimized beam divergence values.")
      f = open('stats.LP','w')
      f.write(str(space_group) + "\n" + unit_cell)

      f.close
      print("I found space group " + str(space_group) + " and a unit cell of "
          + "\n" + unit_cell)


def scale_conv():
   xscale = open('XSCALE.INP','w')
   xscale_out = open("xscale.LP","w+")
   m = movname
   xscale.write("OUTPUT_FILE= " + m +".ahkl"+"\nINPUT_FILE= XDS_ASCII.HKL"
            + "\nRESOLUTION_SHELLS= 10 8 5 3 2.3 2.0 1.7 1.5 1.3 " +
            "1.2 1.1 1.0 0.90 0.80")
   xscale.close()
   run("xscale", stdout= xscale_out)
   print("I scaled the data in XSCALE.")
   xdsconv_out = open("xdsconv.LP", "w+")
   xdsconv = open('XDSCONV.INP','w')
   xdsconv.write("INPUT_FILE= " + m + ".ahkl" + "\nOUTPUT_FILE= " +
            m + ".hkl" + " SHELX" +
            "\nGENERATE_FRACTION_OF_TEST_REFLECTIONS=0.10"
            + "\nFRIEDEL'S_LAW=FALSE")
   xdsconv.close()
   run("xdsconv",stdout= xdsconv_out)
   print("I converted it for use in shelx!")
```

```
   os.system("echo CHIRALITY NONCHIRAL | pointless xdsin XDS_ASCII.HKL > pointless.LP")
   with open('pointless.LP','r') as p1:
      lines = p1.readlines()
      for index, line in enumerate(lines):
         if "   Spacegroup         TotProb SysAbsProb     Reindex         Conditions" in line:
            os.mknod("pointless_group.LP")
            sp1 = lines[index+2]
            sp1_1 = sp1.split()
            for item in sp1_1:
               if item.endswith(")") == True:
                  sp1_2 = str(item).strip(')(')
                  with open('pointless_group.LP','a') as pg:
                     pg.write(str(sp1_2) + "\n")
            sp2 = lines[index+3]
            sp2_1 = sp2.split()
            for item in sp2_1:
               if item.endswith(")") == True:
                  sp2_2 = str(item).strip(')(')
                  with open('pointless_group.LP','a') as pg:
                     pg.write(str(sp2_2) + "\n")
            sp3 = lines[index+4]
            sp3_1 = sp3.split()
            for item in sp3_1:
               if item.endswith(")"):
                  sp3_2 = str(item).strip(')(')
                  with open('pointless_group.LP','a') as pg:
                     pg.write(str(sp3_2) + "\n")
            sp4 = lines[index+5]
            sp4_1 = sp4.split()
            for item in sp4_1:
               if item.endswith(")"):
                  sp4_2 = str(item).strip(')(')
                  with open('pointless_group.LP','a') as pg:
                     pg.write(str(sp4_2) + "\n")
            sp5 = lines[index+6]
            sp5_1 = sp5.split()
            for item in sp5_1:
               if item.endswith(")"):
                  sp5_2 = str(item).strip(')(')
                  with open('pointless_group.LP','a') as pg:
                     pg.write(str(sp5_2) + "\n")
```

```
if __name__== "__main__":
   main()
   autoprocessing()
   scale_conv()
```

4.7.1.3 autosetup_0302_new.py

```
"""
Jessica Burch
"""
import os
import shutil

def main():
   stats = open("stats.LP","w")
   stats.write("Data summary: ")
   files = os.listdir(".")
   for name in files:
      if name.endswith(".ser"):
         newname = name.split("_")
         path = os.getcwd()
         os.mkdir(path + "/" + str(newname[0]))
         os.mkdir(path + "/" + str(newname[0]) + "/images")
         os.mkdir(path + "/" + str(newname[0]) + "/auto_process")
         shutil.move(name,str(path + "/" + newname[0] + "/" + name))
         shutil.copyfile('xds_for_me_0302_new.py', str(path + "/" + newname[0]
         + "/xds_for_me_0302_new.py"))
         print("Setting up files for " + newname[0] + ".")
         os.chdir(path + "/" + str(newname[0]))
         os.system("python3 xds_for_me_0302_new.py")
         os.system("rm xds_for_me_0302_new.py")
         with open('auto_process/XSCALE.LP', 'r') as f:
            lines = f.readlines()
            for index, line in enumerate(lines):
               if " ========== STATISTICS OF INPUT DATA SET ==========" in line:
                  t = lines[index-3]
                  t1 = t.split()
                  completeness = t1[4]
                  Roverall = t1[5]

                  l = lines[index-13]
                  l1 = l.split()
                  t = l1[5]
                  t2 = t[:-1]
                  if float(t2) < 100 and float(t2) > 0:
```

```
    with open('xscale_report.LP','w') as f1:
        f1.write("\n" + str(l1[0]) + " " + str(l1[4])
                + " " + str(l1[5]))
    l = lines[index-12]
    l1 = l.split()
    t = l1[5]
    t2 = t[:-1]
if float(t2) < 100 and float(t2) > 0:
    with open('xscale_report.LP','a') as f1:
        f1.write("\n" + str(l1[0]) + " " + str(l1[4])
                + " " + str(l1[5]))
    l = lines[index-11]
    l1 = l.split()
    t = l1[5]
    t2 = t[:-1]
if 1 < float(t2) < 100 and float(t2) > 0:
    with open('xscale_report.LP','a') as f1:
        f1.write("\n" + str(l1[0]) + " " + str(l1[4])
                + " " + str(l1[5]))
    l = lines[index-10]
    l1 = l.split()
    t = l1[5]
    t2 = t[:-1]
if float(t2) < 100 and float(t2) > 0:
    with open('xscale_report.LP','a') as f1:
        f1.write("\n" + str(l1[0]) + " " + str(l1[4])
                + " " + str(l1[5]))
    l = lines[index-9]
    l1 = l.split()
    t = l1[5]
    t2 = t[:-1]
if float(t2) < 100 and float(t2) > 0:
    with open('xscale_report.LP','a') as f1:
        f1.write("\n" + str(l1[0]) + " " + str(l1[4])
                + " " + str(l1[5]))
    l = lines[index-8]
    l1 = l.split()
    t = l1[5]
    t2 = t[:-1]
if float(t2) < 100 and float(t2) > 0:
    with open('xscale_report.LP','a') as f1:
        f1.write("\n" + str(l1[0]) + " " + str(l1[4])
                + " " + str(l1[5]))
    l = lines[index-7]
    l1 = l.split()
    t = l1[5]
```

```
                t2 = t[:-1]
            if float(t2) < 100 and float(t2) > 0:
                with open('xscale_report.LP','a') as f1:
                    f1.write("\n" + str(l1[0]) + " " + str(l1[4])
                            + " " + str(l1[5]))
                l = lines[index-6]
                l1 = l.split()
                t = l1[5]
                t2 = t[:-1]
            if float(t2) < 100 and float(t2) > 0:
                with open('xscale_report.LP','a') as f1:
                    f1.write("\n" + str(l1[0]) + " " + str(l1[4])
                            + " " + str(l1[5]))
                l = lines[index-5]
                l1 = l.split()
                t = l1[5]
                t2 = t[:-1]
            if float(t2) < 100 and float(t2) > 0:
                with open('xscale_report.LP','a') as f1:
                    f1.write("\n" + str(l1[0]) + " " + str(l1[4])
                            + " " + str(l1[5]))
                l = lines[index-4]
                l1 = l.split()
                t = l1[5]
                t2 = t[:-1]
            if float(t2) < 100 and float(t2) > 0:
                with open('xscale_report.LP','a') as f1:
                    f1.write("\n" + str(l1[0]) + " " + str(l1[4])
                            + " " + str(l1[5]))
            else:
                with open('xscale_report.LP','a') as f1:
                    f1.write("\n ! low resolution data")

        with open('xscale_report.LP','r') as f2:
            xsc = f2.read()
        with open('auto_process/stats.LP','r') as f3:
            ind = f3.read()
        with open('auto_process/pointless_group.LP','r') as f4:
            pg = f4.read()
        os.chdir("..")
        stats = open("stats.LP","a")
        stats.write("\n==========\n" + newname[0] + "\n" + ind + "\nXSCALE stats")
        stats.write("\n" + str(t1[4]) + " " + str(t1[5]) + "\n" + xsc)
        stats.write("\npointless\n" + str(pg))

if __name__== "__main__":
```

```
    main()
```

4.7.1.4 merging_all4.py

```
"""
This script will generate every combination possible for XSCALE merges based
on 4 movies. Written by Jessica Burch
"""

import os
import sys
from itertools import *
from subprocess import run
import itertools
import re

def main():
    successful_files = []
    path = os.getcwd()
    files = os.listdir(".")
    if os.path.isdir(path + "/every_merge") == False:
        os.mkdir(path + "/every_merge")

    #checks for files that successfully processed and adds to a list
    for file in files:
        if os.path.isfile(file + "/auto_process/XSCALE.LP") == True:
            successful_files.append(str(file))

    #makes all possible combinations of files
    for group in power_set(successful_files):
        if len(group) <= 1:
            pass
        if len(group) >= 5:
            pass
        else:
            for item in group:
                r = len(group)
                successful_files.extend(list(permutations(group,r)))
        new_list = list(set(successful_files))

    for entry in new_list:
        if isinstance(entry,str) == True:
            if os.path.isdir(path + "/every_merge/" + entry) == False:
                os.mkdir(path + "/every_merge/" + entry)
                os.chdir(path + "/every_merge/" + entry)
                f = open("XSCALE.INP","w+")
```

```
            f.write("!automatically generated XSCALE file\nOUTPUT_FILE=" +
                  entry + ".ahkl\nINPUT_FILE=../../" + entry
                  + "/auto_process/XDS_ASCII.HKL")
            f.close()
            xscale_out = open("xscale.LP","w")
            run("xscale", stdout=xscale_out)
            os.chdir("../../")
      else:
         title = str(entry)
         name = re.sub("\ |\'|\(|\)|",",title)
         file_name = re.sub(",","_",name)

         if os.path.isdir(path + "/every_merge/" + file_name) == False:
            os.mkdir(path + "/every_merge/" + file_name)
            os.chdir(path + "/every_merge/" + file_name)
            f = open ("XSCALE.INP","w")
            f.write("!automatically generated XSCALE file\nOUTPUT_FILE=" +
                  file_name + ".ahkl\n")
            for item in entry:
               f.write("INPUT_FILE=../../" + item + "/auto_process/"
                     + "XDS_ASCII.HKL\n")
            f.close()
            xscale_out = open("xscale.LP","w")
            run("xscale", stdout=xscale_out)
            os.chdir("../../")

def power_set(iterable):
   s = list(iterable)
   return chain.from_iterable(combinations(s, r) for r in range(len(s)+1))

def combinations(iterable, r):
   pool = tuple(iterable)
   n = len(pool)
   if r > n:
      return
   indices = list(range(r))
   yield tuple(pool[i] for i in indices)
   while True:
      for i in reversed(range(r)):
         if indices[i] != i + n - r:
            break
      else:
         return
      indices[i] += 1
      for j in range(i+1, r):
         indices[j] = indices[j-1] + 1
```

```
        yield tuple(pool[i] for i in indices)


if __name__== "__main__":
    main()
```

4.7.1.5 Lightning_Struc.py

```
"""
Movie to structure with GUI
"""
import os
import sys
from tkinter import *
import tkinter as tk
from tkinter import ttk
from subprocess import run
import random
from tkinter import filedialog
from itertools import *
import re
import shutil
from time import time
from shelxt_library import library


global mol
global rein
global reind
global reindex
global sfac
global filePaths
```

```
global destPath
global indexedFiles
global masterindexedFiles
global mergedFiles

rein = []
reind = []
destPath = []
filePaths = []
indexedFiles = []
masterindexedFiles = []
mergedFiles = []

def processing():
    if int(rein[-1])==0:
        destination = str(destPath[-1]).strip("'")
        reindex.set("original")
    if int(rein[-1])==1:
        indexedFiles.clear()
        d = str(destPath[-1]).strip("'")
        num = str(re_spa_gr.get()).strip("'")
        if os.path.isdir(d + "/" + num) == True:
            for i in range(1,99):
                if os.path.isdir(d + "/" + str(num) + "_" + str(i)) == False:
                    destination = str(d + "/" + str(num) + "_" + str(i))
                    n = str(num + "_" + i)
                    reind.append(n)
                    reindex.set(n)
                    break
        if os.path.isdir(d + "/" + str(num)) == False:
            destination = str(d + "/" + str(num))
            reind.append(num)
            reindex.set(num)
        os.mkdir(destination)
        os.chdir(d)
        filez = os.listdir(".")
        for file in filez:
            f = str(file).strip("'")
            if os.path.isdir(d + "/" + f + "/auto_process") == True:
                os.mkdir(destination + "/" + f)
                os.mkdir(destination +"/" + f + "/auto_process")
                thething = str("cp -r " + d + "/" + f + "/auto_process/*" + " "
                               + destination + "/" + f + "/auto_process/.")
                os.system(thething)
    stats = open("stats.LP","w")
    stats.write("Data summary: ")
```

```
os.chdir(destination)
os.listdir(".")
for file in filePaths:
   newfile = str(file).strip("'")
   if newfile.endswith(".ser"):
      filename = newfile.split("/")
      movname = str(filename[-1]).strip("'")
      newname = movname.split("_")
      n = str(newname[0])
      if os.path.isdir(destination + "/" + n) == True:
         if os.path.isdir(str(destPath[-1]).strip("'") + "/" + n + "/images") == True:
            os.chdir(str(destPath[-1]).strip("'") + "/" + n + "/images")
            images = os.listdir(".")
            frames = str(len(images))
         if os.path.isdir(destination + "/" + n + "/auto_process") == True:
            os.chdir(destination + "/" + n + "/auto_process")
            os.system("rm *")
         if os.path.isdir(str(destPath[-1]).strip("'") + "/" + n + "/images") == False:
            os.mkdir(destination + "/" + n + "/images")
            conversion = str("/mnt/c/Users/jessi/Desktop/Test_Dataset/" +
                     "LightningStruc_Testing/ser2smv -P 0.014 -B 2 -r "
                     + newname[2] + " -w 0.0251 -d " + newname[1] +
                     " -E " + newname[3] + " -M 200 -v -o " +
                     n + "_###.img " + newfile)
            os.system(conversion + ' > summary.LP')
            with open('summary.LP') as f1:
               lines = f1.readlines()
            with open('summary.LP', 'w') as f2:
               f2.writelines(lines[-15:])
            with open('summary.LP', 'r') as f:
               line = f.readline()
               for line in f:
                  if "+++" in line:
                     element = str.split(line)
                     frames = str(element[2])
            os.system("rm summary.LP")
         if os.path.isdir(destination +"/"+ n +"/auto_process") == False:
            os.mkdir(destination + "/" + n + "/auto_process")
      if os.path.isdir(destination + "/" + n) == False:
         os.mkdir(destination + "/" + n)
         os.mkdir(destination + "/" + n + "/images")
         os.mkdir(destination + "/" + n + "/auto_process")
         os.chdir(destination + "/" + n + "/images")
         conversion = str("/mnt/c/Users/jessi/Desktop/Test_Dataset/" +
             "LightningStruc_Testing/ser2smv -P 0.014 -B 2 -r "
             + newname[2] + " -w 0.0251 -d " + newname[1] +
```

```
            " -E " + newname[3] + " -M 200 -v -o " +
            n + "_###.img " + newfile)
        os.system(conversion + ' > summary.LP')
        with open('summary.LP') as f1:
            lines = f1.readlines()
        with open('summary.LP', 'w') as f2:
            f2.writelines(lines[-15:])
        with open('summary.LP', 'r') as f:
            line = f.readline()
            for line in f:
                if "+++" in line:
                    element = str.split(line)
                    frames = str(element[2])
        os.system("rm summary.LP")
    os.chdir(destination + "/" + n + "/auto_process")
    f = open("XDS.INP","w+")
    f.write("JOB= XYCORR INIT COLSPOT IDXREF DEFPIX INTEGRATE CORRECT"
+
            "\n!JOB=DEFPIX INTEGRATE CORRECT")
    if int(auto_o.get())==0:
        x = str("1018")
        y = str("1000")
    if int(auto_o.get())==1:
        x1 = str(ORGX.get(1.0,END)).strip("'")
        x = x1.strip("\n")
        y1 = str(ORGY.get(1.0,END)).strip("'")
        y = y1.strip("\n")
    osc = str(float(newname[3]) * float(newname[2]))
    adj = float(corrf.get())
    corrected_distance = str(float(newname[1]) * adj)
    if int(rein[-1])==0:
        data_path = str(destination + "/" + n + "/images/" + n)
    if int(rein[-1])==1:
        data_path = str(str(destPath[-1]).strip("'") + "/" + n +
                    "/images/" + n)
    e = str(elec.get())
    f.write("\nORGX= " + x + " ORGY= " + y +
            "\nDETECTOR_DISTANCE= " + corrected_distance +
            "\nOSCILLATION_RANGE= " + osc + "\nX-RAY_WAVELENGTH= " + e)
    f.write("\nNAME_TEMPLATE_OF_DATA_FRAMES=" + data_path + "_???.img" +
            "\nBACKGROUND_RANGE=1 10\n!DELPHI=15\n")
    if int(rein[-1])==0:
        if int(auto_uc.get())==0:
            f.write("!SPACE_GROUP_NUMBER=0" +
                    "\n!UNIT_CELL_CONSTANTS= 1 1 1 90 90 90")
        if int(auto_uc.get())==1:
```

```
            man_sp1 = str(spa_gr.get(1.0,END)).strip("'")
            man_sp = man_sp1.strip("\n")
            man_uc1 = str(uni_c.get(1.0,END)).strip("'")
            man_uc = man_uc1.strip("\n")
            f.write("SPACE_GROUP_NUMBER=" + man_sp +
                  "\nUNIT_CELL_CONSTANTS= " + man_uc)
        if int(rein[-1])==1:
           man_sp = str(re_spa_gr.get()).strip("'")
           man_uc = str(re_uni_c.get()).strip("'")
           f.write("SPACE_GROUP_NUMBER=" + man_sp +
                 "\nUNIT_CELL_CONSTANTS= " + man_uc)
        if int(auto_r.get())==0:
           res1 = float(newname[1]) * 0.0007
           res = str(round(res1, 2))
        if int(auto_r.get())==1:
           r = str(reso.get(1.0,END)).strip("'")
           res = r.strip("\n")
        rota = str(rotaxis.get())
        f.write("\nINCLUDE_RESOLUTION_RANGE= 40 " + res +
            "\nTEST_RESOLUTION_RANGE= 40 " + res + "\nTRUSTED_REGION=0.0 1.2"
+
            "\nVALUE_RANGE_FOR_TRUSTED_DETECTOR_PIXELS=6000. 30000." +
            "! parameters for detector and beamline:" +
            "\nDETECTOR= ADSC MINIMUM_VALID_PIXEL_VALUE= 1 OVERLOAD=
65000" +
            "\nSENSOR_THICKNESS= 0.01" + "\nNX= 2048 NY= 2048 QX= 0.0280000009"
            + " QY= 0.0280000009" + "\nROTATION_AXIS=" + rota +
            "\nDIRECTION_OF_DETECTOR_X-AXIS=1 0 0" +
            "\nDIRECTION_OF_DETECTOR_Y-AXIS=0 1 0" +
            "\nINCIDENT_BEAM_DIRECTION=0                                    0
1\nFRACTION_OF_POLARIZATION=0.98"
            + "\nPOLARIZATION_PLANE_NORMAL=0 1 0" +
            "\nREFINE(IDXREF)=CELL BEAM ORIENTATION AXIS ! DISTANCE" +
            "\nREFINE(INTEGRATE)= DISTANCE BEAM ORIENTATION ! AXIS CELL" +
            "\nREFINE(CORRECT)=CELL BEAM ORIENTATION AXIS ! DISTANCE !" +
            "\n\nDATA_RANGE= 1 " + frames + "\nSPOT_RANGE= 1 "
            + frames)
        sp = "4"
        minpix = "7"
        f.write("\nSTRONG_PIXEL=          "          +          sp          +
"\nMINIMUM_NUMBER_OF_PIXELS_IN_A_SPOT= "
            + minpix + "\n!\n!")
        f.close()
        xds_out = open("XDS.LP", "w+")
        run("xds", stdout= xds_out)
        autoprocessing()
```

```
            os.listdir(".")
            if os.path.isfile("CORRECT.LP") == True:
                scale_conv()
        r = ("\n".join(indexedFiles))
        masterindexedFiles.append(r)
        option_changed2()


def autoprocessing():
    if os.path.isfile('X-CORRECTIONS.cbf') == False:
        xds_out = open("XDS.LP", "w+")
        run("xds", stdout= xds_out)

    if os.path.isfile('XPARM.XDS') == False:
        for i in range(10):
            with open('XDS.INP') as f1:
                lines = f1.readlines()
            with open('XDS.INP', 'w') as f2:
                strong = random.randrange(3,9,1)
                mpix = random.randrange(4,9,1)
                f2.writelines(lines[:-4])
                f2.write("STRONG_PIXEL= " + str(strong) +
                         "\nMINIMUM_NUMBER_OF_PIXELS_IN_A_SPOT= " + str(mpix) +
                         "\n!\n!")
                f2.close()
                xds_out = open("XDS.LP", "w+")
                run("xds",stdout= xds_out)
            if os.path.isfile('XPARM.XDS') == True:
                if os.path.isfile('DEFPIX.LP') == False:
                    with open('XDS.INP') as f1:
                        lines = f1.readlines()
                    with open('XDS.INP', 'w') as f2:
                        f2.write("!JOB=XYCORR INIT COLSPOT IDXREF DEFPIX INTEGRATE
CORRECT"
                                 + "\nJOB=DEFPIX INTEGRATE CORRECT\n")
                        f2.writelines(lines[2:])
                        f2.close()
                        print("Less than 70% of spots went through. Running with JOB= DEFPIX "
                              + "INTEGRATE CORRECT.")
                        xds_out = open("XDS.LP", "w+")
                        run("xds",stdout= xds_out)
                        if os.path.isfile('XPARM.XDS') == False:
                            newname = str(os.getcwd()).split("/")
                            m = str(newname[-2]).strip("'")
                            result = str(m + ", Failed to index")
                            indexedFiles.append(result)
```

```
                else:
                    return autoprocessing()
            else:
                return autoprocessing()
        else:
            newname = str(os.getcwd()).split("/")
            m = str(newname[-2]).strip("'")
            result = str(m + ", Failed to index")
            indexedFiles.append(result)
            f2.close()

    if os.path.isfile('DEFPIX.LP') == False:
        with open('XDS.INP') as f1:
            lines = f1.readlines()
        with open('XDS.INP', 'w') as f2:
            f2.write("!JOB=XYCORR INIT COLSPOT IDXREF DEFPIX INTEGRATE
CORRECT"
                     + "\nJOB=DEFPIX INTEGRATE CORRECT\n")
            f2.writelines(lines[2:])
            f2.close()
        print("Less than 70% of spots went through. Running with JOB= DEFPIX "
              + "INTEGRATE CORRECT.")
        xds_out = open("XDS.LP", "w+")
        run("xds",stdout= xds_out)
        return autoprocessing()

    if os.path.isfile("INTEGRATE.HKL") == False:
        with open('XDS.INP') as f1:
            lines = f1.readlines()
        with open('XDS.INP', 'w') as f2:
            f2.writelines(lines)
            f2.write("\nBEAM_DIVERGENCE= 0.03 BEAM_DIVERGENCE_E.S.D.= 0.003" +
            "\nREFLECTING_RANGE=1.0 REFLECTING_RANGE_E.S.D.= 0.2")
            f2.close()
            print("Adding beam divergence values to correct a common error.")
            xds_out = open("XDS.LP", "w+")
            run("xds",stdout= xds_out)
            return autoprocessing()
    if os.path.isfile("CORRECT.LP") == True:
        print ("Successful indexing!")
        return mosaicity()

def mosaicity():
    with open('XDS.INP') as f1:
        lines = f1.readlines()
    with open('XDS.INP', 'w') as f2:
```

```
        f2.write("!JOB=XYCORR INIT COLSPOT IDXREF DEFPIX INTEGRATE CORRECT"
                 + "\nJOB=DEFPIX INTEGRATE CORRECT\n")
        f2.writelines(lines[2:-2])
    with open('INTEGRATE.LP', 'r') as l1:
        f2 = open('XDS.INP', 'a')
        line = l1.readline()
        for line in l1:
            if "BEAM_DIVERGENCE= " in line:
                f2.write(line)
            if "REFLECTING_RANGE=" in line:
                f2.write(line)
        f2.close()
        return iterate_opt()

def iterate_opt():
    with open('XDS.LP') as f1:
        lines = f1.readlines()
    with open('XDS.LP', 'w') as f2:
        f2.writelines(lines[-26:])
    with open('XDS.LP', 'r') as f:
        line = f.readline()
        for line in f:
            if "     a        b          ISa" in line:
                next_line = f.readline()
                stats = str.split(next_line)
                Isa1 = float(stats[2])
                print("Isa: " + str(Isa1) + ". Testing new values now.")
    xds_out = open("XDS.LP", "w+")
    run("xds",stdout= xds_out)
    with open('XDS.LP') as f1:
        lines = f1.readlines()
    with open('XDS.LP', 'w') as f2:
        f2.writelines(lines[-26:])
    with open('XDS.LP', 'r') as f:
        line = f.readline()
        for line in f:
            if "     a        b          ISa" in line:
                new_next_line = f.readline()
                new_stats = str.split(new_next_line)
                Isa2 = float(new_stats[2])
                print("Isa: " + str(Isa2))
            if "SPACE_GROUP_NUMBER=" in line:
                number = str.split(line)
                space_group = number[1]
            if "UNIT_CELL_CONSTANTS=" in line:
                cell = str.split(line)
```

```
            temp = cell[-6:]
            temp_str = str(temp).strip("]['")
            temp_str2 = temp_str.replace(",","")
            unit_cell = temp_str2.replace("'","")
    Isa_change = abs(Isa2 - Isa1)
    if Isa_change > 0.5:
        print("I'm trying to optimize beam divergence values.")
        return iterate_opt()
    else:
        print("Optimized beam divergence values.")
        f = open('stats.LP','w')
        f.write(str(space_group) + ", " + unit_cell)

        f.close
        print("I found space group " + str(space_group) + " and a unit cell of "
              + "\n" + unit_cell)

def scale_conv():
    newname = str(os.getcwd()).split("/")
    xscale = open('XSCALE.INP','w')
    xscale_out = open("xscale.LP","w+")
    m = newname[-2]
    xscale.write("OUTPUT_FILE= " + m +".ahkl"+"\nINPUT_FILE= XDS_ASCII.HKL"
                 + "\nRESOLUTION_SHELLS= 10 8 5 3 2.3 2.0 1.7 1.5 1.3 " +
                 "1.2 1.1 1.0 0.90 0.80")
    xscale.close()
    run("xscale", stdout= xscale_out)
    with open('XSCALE.LP', 'r') as f:
        line = f.readline()
        for line in f:
            if "    total" in line:
                if len(str.split(line)) >= 10:
                    stats = str.split(line)
                    I_sig = str(stats[8])
                    complet = str(stats[4]).strip("%")
                    R_fac = str(stats[5]).strip("%")
                    CC_half = str(stats[10]).strip("*")
    with open('stats.LP','r') as f3:
        ind = f3.read()
    if os.path.isfile("pointless_group.LP") == True:
        with open('pointless_group.LP','r') as f4:
            pg = f4.read()
    os.chdir("../../")
    stats = open("stats.LP","a")
    stats.write("\n==========\n" + newname[-2] + "\n" + ind + "\nXSCALE stats")
    stats.write("\n" + complet + " " + R_fac + "\n" + CC_half)
```

```
    if os.path.isfile(newname[-2] + "/auto_process/pointless_group.LP") == True:
        stats.write("\npointless\n" + str(pg))
    result = str(m + ", " + ind + "\n" + complet + "% completeness, " +
                 R_fac + "% R1, " + CC_half + " CC_half")
    indexedFiles.append(result)

def redo():
    rein.clear()
    rein.append("1")
    processing()

def processing_call():
    rein.clear()
    rein.append("0")
    processing()


def merge():
    m = str(merge1.get()).strip('"')
    path = str(destPath[-1]).strip('"')
    os.chdir(path)
    if m != "original":
        os.chdir(path + "/" + m)
    files = os.listdir(".")
    if os.path.isdir(path + "/every_merge_" + m) == False:
        os.mkdir(path + "/every_merge_" + m)

    #checks for files that successfully processed and adds to a list
    successful_files = []
    for file in files:
        if os.path.isfile(file + "/auto_process/XSCALE.LP") == True:
            successful_files.append(str(file))

    #makes all possible combinations of files
    for group in power_set(successful_files):
        if len(group) <= 1:
            pass
        if len(group) >= 4:
            pass
        else:
            for item in group:
                r = len(group)
                successful_files.extend(list(permutations(group,r)))
        new_list = list(set(successful_files))

    for entry in new_list:
```

```
    if isinstance(entry,str) == True:
        if os.path.isdir(path + "/every_merge_" + m + "/" + entry) == False:
            os.mkdir(path + "/every_merge_" + m + "/" + entry)
            os.chdir(path + "/every_merge_" + m + "/" + entry)
            f = open("XSCALE.INP","w+")
            f.write("!automatically generated XSCALE file\nOUTPUT_FILE=" +
                    entry + ".ahkl\nINPUT_FILE=../../" + entry
                    + "/auto_process/XDS_ASCII.HKL")
            f.close()
            xscale_out = open("xscale.LP","w")
            run("xscale", stdout=xscale_out)
            os.chdir("../../")
    else:
        title = str(entry)
        name = re.sub("\ |\'|\(|\)|","",title)
        file_name = re.sub(",","_",name)

        if os.path.isdir(path + "/every_merge_" + m + "/" + file_name) == False:
            os.mkdir(path + "/every_merge_" + m + "/" + file_name)
            os.chdir(path + "/every_merge_" + m + "/" + file_name)
            f = open ("XSCALE.INP","w")
            f.write("!automatically generated XSCALE file\nOUTPUT_FILE=" +
                    file_name + ".ahkl\n")
            for item in entry:
                f.write("INPUT_FILE=../../" + item + "/auto_process/"
                        + "XDS_ASCII.HKL\n")
            f.close()
            xscale_out = open("xscale.LP","w")
            run("xscale", stdout=xscale_out)
            os.chdir("../../")
os.chdir(path + "/every_merge_" + m + "/" )
files = os.listdir(".")
for file in files:
    if str(file).endswith("_"):
        shutil.rmtree(file)
files = os.listdir(".")
for file in files:
    with open(file + '/XSCALE.LP', 'r') as f:
                line = f.readline()
                for line in f:
                    if "    total" in line:
                        if len(str.split(line)) >= 10:
                            stats = str.split(line)
                            I_sig = str(stats[8])
                            complet = str(stats[4]).strip("%")
                            R_fac = str(stats[5]).strip("%")
```

```
                        CC_half = str(stats[10]).strip("*")
                        formatted = str(CC_half + " CC half," +
                                complet + "%completeness,"
                                + R_fac + "% R," + file +
                                "\n")
                        mergedFiles.append(formatted)

    mergedFiles.sort(reverse= True)
    separator = ' '
    m = str(separator.join(mergedFiles)).strip("'")
    merge_update.configure(state='normal')
    merge_update.insert('end',m)
    merge_update.configure(state='disabled')

def power_set(iterable):
    s = list(iterable)
    return chain.from_iterable(combinations(s, r) for r in range(len(s)+1))

def combinations(iterable, r):
    pool = tuple(iterable)
    n = len(pool)
    if r > n:
        return
    indices = list(range(r))
    yield tuple(pool[i] for i in indices)
    while True:
        for i in reversed(range(r)):
            if indices[i] != i + n - r:
                break
        else:
            return
        indices[i] += 1
        for j in range(i+1, r):
            indices[j] = indices[j-1] + 1
        yield tuple(pool[i] for i in indices)

def solve():
    s = solv.get()
    path = str(destPath[-1]).strip("'")
    if os.path.isdir(path + "/" + "every_merge_" + s) == True:
        os.chdir(path + "/" + "every_merge_" + s)
    temp_ins = open('temp.ins','w')
    temp_ins.write("entry from widget\n" + str(atom_entry.get()) + "\n" +
                str(SFAC_entry.get()))
    temp_ins.close()
    if os.path.isdir(path + "/solutions") == True:
```

```
    os.mkdir(path + "/solutions/" + s)
    os.mkdir(path + "/solutions/" + s + "/no_res")
    os.mkdir(path + "/solutions/" + s + "/yes_res")
    os.mkdir(path + "/solutions/" + s + "/best_solutions")
files = os.listdir(".")
filecount = len(files)
path2 = os.getcwd()
completed_files = []
times = []
merged_data = open("merged_stats.csv","w+")
merged_data.write("CC_half,I_sig,completeness,R_factor,file_name")
with open('temp.ins', 'r') as f1:
    lines = f1.readline()
    l1 = []
    for lines in f1:
        l1.append(lines)
    print(str(l1))
for file in files:
    completed_files.append(file)
    t1 = time()
    name = str(file)
    if os.path.isfile(path2 + "/" + file + "/" + name + ".ahkl") == True:
        os.chdir(path2 + "/" + file)
        if os.path.isfile(path2 + "/" + file + "/" + name + ".hkl") == False:
            xdsconv_out = open("xdsconv.LP", "w+")
            xdsconv = open('XDSCONV.INP','w')
            xdsconv.write("INPUT_FILE= " + name + ".ahkl" + "\nOUTPUT_FILE= "
                    + name + ".hkl" + " SHELX" +
                    "\nGENERATE_FRACTION_OF_TEST_REFLECTIONS=0.10"
                    + "\nFRIEDEL'S_LAW=FALSE")
            xdsconv.close()
            run("xdsconv",stdout= xdsconv_out)

        if os.path.isfile(path2 + "/" + file + "/" + name + ".hkl") == True:
            with open('xdsconv.LP', 'r') as f:
                line = f.readline()
                for line in f:
                    if "SPACE_GROUP_NUMBER=" in line:
                        element = str.split(line)
                        sp_grp = str(element[1])
                    if "UNIT_CELL_CONSTANTS=" in line:
                        element = str.split(line)
                        uc = str(element[1:])
                        for ch in ["]","[",",","'"]:
                            if ch in uc:
                                uc2 = uc.replace(ch," ")
```

```
            uc3= uc2.strip("][,")
            unit_cell = uc3.replace(",", " ")

        library(sp_grp)
        atoms = str(l1[0]).strip("\n")
        unit = l1[1]
        insfile = open(name + '.ins', "w+")
        insfile.write("TITL automerge " + name + " in spgrp " + sp_grp
                  + "\nCELL  0.0251000002   " + unit_cell +
                  "ZERR   4.00   0.0010   0.0020   0.0041   0   0   0\n\n"
                  + ins_sym + "\nSFAC " + atoms + "\nUNIT "
                  + unit +
                  "\nREM NTRY 50000\nFIND 6\nPLOP 9 6 6 7 7 8 8 9"
                  + "\nHKLF 4\nEND")
        insfile.close()
        shelxt_out = open("shelxtout.LP","w+")
        com = str("shelxt " + g + " " + name)
        os.system(com)
        os.listdir(".")
        if os.path.isfile(path + "/" + file + "/" + name + ".hkl") == True:
           if os.path.isfile(path + "/" + file + "/" + name + "_a.res") == True:
              with open(name + "_a.res") as f2:
                 line2 = f2.readline()
                 for line2 in f2:
                    if "REM SHELXT solution" in line2:
                       s_stat = str.split(line2)
                       R1 = s_stat[6]
                    if "REM Formula found by SHELXT:" in line2:
                       sp_stat = str(line2).split(":")
                       formula = sp_stat[-1]
                       stats = str(R1 + " R1, " + formula +
                                 + " formula found for " +
                                 name)
                       solve_update.configure(state='normal')
                       solve_update.insert('end',stats)
                       solve_update.configure(state='disabled')
              os.chdir("..")
              os.rename(path + "/solutions/" + s + "/yes_res/" + file)
        elif os.path.isfile(path + "/" + file + "/" + name + "_a.res") == False:
           os.chdir("..")
           os.rename(path + "/solutions/" + s + "/no_res/" + file)
t2 = time()
elapsed = t2 - t1
times.append(elapsed)
files_compl = len(completed_files)
files_remaining = (filecount - files_compl)
```

```
        if files_remaining % 10 == 0:
            avg_time = (sum(times)/float(len(times)))
            time_remain = (avg_time * float(files_remaining)) / 60.0
            print(str(files_compl) + "/" + str(filecount) + " files completed. Estimated time
remaining: "
                  + str(time_remain))


def fileadd():
    files = filedialog.askopenfilename(title="Select data files",filetypes =
                                ((("ser files","*.ser"),
                                 ("all files","*.*")),multiple=True)
    var = root.tk.splitlist(files)
    for f in var:
        f_a = str(f).strip(")(")
        f_b = f_a.strip(",")
        if f_b in filePaths:
          process_update.configure(state='normal')
          process_update.configure(state='disabled')
        else:
            filePaths.append(f_b)
            f1 = f.split("/")
            f2 = f1[-1]
            process_update.configure(state='normal')
            process_update.insert('end',str(f2 + "\n"))
            process_update.configure(state='disabled')
def fileclear():
    process_update.configure(state='normal')
    process_update.delete("1.0", END)
    process_update.configure(state='disabled')
    filePaths.clear()
def workdir():
    destPath.clear()
    workdir = filedialog.askdirectory()
    v = root.tk.splitlist(workdir)
    v1 = str(v).strip(")(")
    v2 = v1.strip(",")
    v3 = v2.strip("'")
    destPath.append(v2)
    process3_update.configure(state='normal')
    process_update.delete("1.0", END)
    process3_update.insert('end',v3)
    process3_update.configure(state='disabled')

def option_changed():
    s = scope.get()
```

```
    if str(s) == "F200C":
        elec.set("0.0251000002")
        corrf.set("0.943")
        rotaxis.set("0 -1 0")
    if str(s) == "F200C old":
        elec.set("0.0251000002")
        corrf.set("0.943")
        rotaxis.set("-1 0 0")
    if str(s) == "F30":
        elec.set("0.019687")
        corrf.set("1.76")
        rotaxis.set("-0.829 -0.559 0")
    if str(s) == "other":
        elec.set("")
        corrf.set("1")
        rotaxis.set("")

def option_changed2():
    if len(masterindexedFiles) >= 1:
        for item in reind:
            obj = str(item).strip("'")
            if str(reindex.get()) == obj:
                numero = int(reind.index(item))
                if len(masterindexedFiles) >= (numero + 1):
                    summary_text = str(masterindexedFiles[numero])
                    process2_update.configure(state='normal')
                    process2_update.delete("1.0", END)
                    f2 = summary_text.strip("'")
                    process2_update.insert('end',str(f2 + "\n"))
                    process2_update.configure(state='disabled')
                    combobox1.configure(values=reind)

def selected():
    auto_r.get()
    if int(auto_r.get())==0:
        reso.configure(state='normal')
        reso.delete("1.0", END)
        reso.configure(state='disabled')
    if int(auto_r.get())==1:
        reso.configure(state='normal')
def selected2():
    auto_o.get()
    if int(auto_o.get())==0:
        ORGX.configure(state='normal')
        ORGX.delete("1.0", END)
        ORGX.configure(state='disabled')
```

```
        ORGY.configure(state='normal')
        ORGY.delete("1.0", END)
        ORGY.configure(state='disabled')
    if int(auto_o.get())==1:
        ORGX.configure(state='normal')
        ORGY.configure(state='normal')
def s3():
    auto_uc.get()
    if int(auto_uc.get())==0:
        spa_gr.configure(state='normal')
        spa_gr.delete("1.0", END)
        spa_gr.configure(state='disabled')
        uni_c.configure(state='normal')
        uni_c.delete("1.0", END)
        uni_c.configure(state='disabled')
    if int(auto_uc.get())==1:
        spa_gr.configure(state='normal')
        uni_c.configure(state='normal')
def s4():
    shelxd.get()
    if int(shelxd.get())==0:
        print("yay")
    if int(shelxd.get())==1:
        print("boo")
def main():
    print("howdy ho")
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Start                                    of
GUI~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
root = Tk()
photo = PhotoImage(file = "icon.gif")
root.title("LightningStruc")
root.iconphoto(False,photo)
root.columnconfigure(0,weight=1)
root.rowconfigure(0,weight=1)
n = ttk.Notebook(root)
n.grid(column=0,row=0)
ttk.Label(root, text="Status:",justify=LEFT).grid(column=0,row=1,sticky=W,pady=5)
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~First
page~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
f1 = ttk.Frame(n, padding="12 12 12 12")
f1.grid(column=2, row=1, sticky=(N, W, E, S))
f1.columnconfigure(0, weight=1)
f1.rowconfigure(0, weight=1)
sf1 = ttk.Frame(f1, borderwidth=5, relief=GROOVE)
sf1.grid(column=1,row=0,sticky=(N, W, E, S),pady=3)
sf1.columnconfigure(0, weight=1)
```

```
sf1.columnconfigure(0, weight=1)
s1f1 = ttk.Frame(f1, borderwidth=5, relief=GROOVE)
s1f1.grid(column=2,row=0,sticky=(N, W, E, S),pady=3)
s1f1.columnconfigure(0, weight=1)
s1f1.columnconfigure(0, weight=1)
s4f1 = ttk.Frame(f1, borderwidth=5, relief=GROOVE)
s4f1.grid(column=1,row=1,columnspan=2,sticky=W,pady=3)
s4f1.columnconfigure(0, weight=1)
s4f1.columnconfigure(0, weight=1)
s2f1= ttk.Frame(f1)
s2f1.grid(column=1,row=2,pady=3)
s3f1= ttk.Frame(f1)
s3f1.grid(column=1,row=3,columnspan=2)
elec = StringVar()
corrf = StringVar()
scope = StringVar()
rotaxis = StringVar()
elec.set("0.0251000002")
corrf.set("0.943")
rotaxis.set("0 -1 0")
scope.set("F200C")
scope.trace("w",lambda a,b,c: option_changed())
sc = tk.OptionMenu(sf1, scope, "F200C", "F200C old", "F30", "other").grid(
      column=1,row=0)
kEv = ttk.Entry(sf1, width=6, textvariable=elec).grid(column=1, row=1)
SFAC_entry = ttk.Entry(sf1, width=6, textvariable=corrf).grid(column=1, row=2)
rot = ttk.Entry(sf1, width =6, textvariable=rotaxis).grid(column=1, row=3)
ttk.Label(sf1, text="TEM Settings").grid(column=0,row=0)
ttk.Label(sf1, text="Electron wavelength:", justify=LEFT).grid(column=0, row=1,sticky=W)
ttk.Label(sf1, text="Å",justify=LEFT).grid(column=2,row=1,sticky=W)
ttk.Label(sf1, text="Distance correction:", justify=LEFT).grid(column=0, row=2,sticky=W)
ttk.Label(sf1, text="Rotation axis: ", justify=LEFT).grid(column=0,row=3,sticky=W)
ttk.Button(f1, text="Start Processing",command=processing_call).grid(
         column=2, row=5,sticky=E,pady=5)
ttk.Button(s2f1, text='Add files', command=fileadd).grid(column=1, row=1,
        sticky=E)
ttk.Button(s2f1, text='Clear files', command=fileclear).grid(column=2,row=1,
       sticky=E)
ttk.Button(s2f1, text='Destination folder', command=workdir).grid(
        column=3, row=1, sticky=E)
ttk.Label(s3f1, text="Data files:",justify=LEFT).grid(column=1,row=1,sticky=W)
process_txt = StringVar()
process_update = tk.Text(s3f1,height=10, width=70)
process_update.grid(column=1,row=2, columnspan=3)
process_update.configure(state='disabled')
```

```
ttk.Label(s3f1, text="Output location:",justify=LEFT).grid(column=1,row=3,sticky=W)
process3_txt = StringVar()
process3_update = tk.Text(s3f1,height=2, width=70)
process3_update.grid(column=1,row=4, columnspan=3)
process3_update.configure(state='disabled')

reso = StringVar()
reso = tk.Text(s1f1, height=1, width=4)
reso.grid(column=1, row=2)
ttk.Label(s1f1, text="Collection Settings").grid(column=0,row=0,pady=4)

ttk.Label(s1f1, text="Auto resolution cutoff:",justify=LEFT).grid(column=0,
        row=1,sticky=W)
auto_r = StringVar()
auto_r.set("0")
reso.configure(state='disabled')
rb1 = ttk.Radiobutton(s1f1, text='on', variable=auto_r, value=0,command=selected)
rb2 = ttk.Radiobutton(s1f1, text='off', variable=auto_r, value=1,command=selected)
rb1.grid(column=1,row=1)
rb2.grid(column=2,row=1)
ttk.Label(s1f1, text="Maximum resolution:",justify=LEFT).grid(column=0, row=2,
        sticky=W)
ttk.Label(s1f1, text="Å",justify=LEFT).grid(column=2,row=2,sticky=W)
ttk.Label(s1f1, text="Auto origin assignment:",justify=LEFT).grid(column=0,
        row=4,sticky=W)
auto_o = StringVar()
auto_o.set("0")
rb1 = ttk.Radiobutton(s1f1, text='on', variable=auto_o, value=0,command=selected2)
rb2 = ttk.Radiobutton(s1f1, text='off', variable=auto_o, value=1,command=selected2)
rb1.grid(column=1,row=4)
rb2.grid(column=2,row=4)
ttk.Label(s1f1, text="Origin coordinates X,Y:",justify=LEFT).grid(column=0,
        row=5,sticky=W)
ORGX = StringVar()
ORGY = StringVar()
ORGX = tk.Text(s1f1,height=1,width=4)
ORGX.grid(column=1,row=5)
ORGY = tk.Text(s1f1,height=1,width=4)
ORGY.grid(column=2,row=5)
ORGX.configure(state='disabled')
ORGY.configure(state='disabled')
ttk.Label(s4f1,                                    text="Force                                    unit
cell?",justify=LEFT).grid(column=0,row=0,sticky=W,padx=10,pady=5)
auto_uc = StringVar()
auto_uc.set("0")
radb1 = ttk.Radiobutton(s4f1, text="no",variable=auto_uc, value=0,command=s3)
```

```
radb2 = ttk.Radiobutton(s4f1, text="yes",variable=auto_uc, value=1, command=s3)
radb1.grid(column=1,row=0,sticky=W)
radb2.grid(column=1,row=1,sticky=W)
ttk.Label(s4f1, text="Space group:").grid(column=0,row=2)
ttk.Label(s4f1, text="Unit cell:",justify=LEFT).grid(column=1,row=2,sticky=W)
spa_gr = StringVar()
uni_c = StringVar()
spa_gr = tk.Text(s4f1,width=5,height=1)
spa_gr.grid(column=0,row=3)
uni_c = tk.Text(s4f1,width=55,height=1)
uni_c.grid(column=1,row=3,sticky=W,columnspan=3)
spa_gr.configure(state='disabled')
uni_c.configure(state='disabled')
#end of first page  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Second
page~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
f2 = ttk.Frame(n, padding="12 12 12 12") # second page
f2.grid(column=0, row=0, sticky=(N, W, E, S))
f2.columnconfigure(0, weight=1)
f2.rowconfigure(0, weight=1)

sf2 = ttk.Frame(f2, borderwidth=5, relief=GROOVE)
sf2.grid(column=0,row=0,sticky=(N, W, E, S),pady=3)
sf2.columnconfigure(0, weight=1)
sf2.rowconfigure(0, weight=1)

s1f2 = ttk.Frame(f2, borderwidth=5, relief=GROOVE)
s1f2.grid(column=0,row=1,sticky=(N, W, E, S),pady=3)
s1f2.columnconfigure(0, weight=1)
s1f2.rowconfigure(0, weight=1)

s2f2 = ttk.Frame(f2, borderwidth=5, relief=GROOVE)
s2f2.grid(column=0,row=2,sticky=(N,W,E,S),pady=3)
s2f2.columnconfigure(0, weight=1)
s2f2.rowconfigure(0, weight=1)

ttk.Label(sf2, text="Indexing results:",justify=LEFT).grid(column=0,
        row=0,sticky=W)

process2_update = tk.Text(sf2,height=10, width=70)
process2_update.grid(column=0,row=1, columnspan=3)
process2_update.configure(state='disabled')
reindex = StringVar()
reind.append("original")
reindex.set("original")
combobox1 = ttk.Combobox(sf2,textvariable=reindex,state='readonly',values=reind)
```

```
combobox1.grid(column=2,row=0,sticky=E)
reindex.trace("w",lambda a,b,c: option_changed2())

ttk.Label(s1f2, text="Re-indexing:", justify=LEFT).grid(column=0,row=0,
        sticky=W,pady=4)
ttk.Label(s1f2, text="Space group:").grid(column=0,row=1)
ttk.Label(s1f2, text="Unit cell:",justify=LEFT).grid(column=1,row=1,sticky=W)
re_spa_gr = StringVar()
re_uni_c = StringVar()
r_spa_gr = ttk.Entry(s1f2,width=5,textvariable=re_spa_gr).grid(column=0,row=2)
r_uni_c = ttk.Entry(s1f2,width=55,textvariable=re_uni_c).grid(column=1,row=2,
                 sticky=W,columnspan=3)

ttk.Button(s1f2, text='Start Re-indexing', command=redo).grid(
          column=3, row=0, sticky=E,pady=4)

ttk.Label(s2f2, text="Merge screening:",justify=LEFT).grid(column=0,row=0,
        sticky=W,pady=4)
merge_update = tk.Text(s2f2,height=10, width=70)
merge_update.grid(column=0,row=1, columnspan=3)
merge_update.configure(state='disabled')
ttk.Button(s2f2, text='Start Merging',command=merge).grid(column=2,row=2,
         sticky=E,pady=4)

merge1 = StringVar()
merge1.set("original")
combobox2 = ttk.Combobox(s2f2,textvariable=merge1,state='readonly',values=reind)
combobox2.grid(column=2,row=0,sticky=E)
#end of second page  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~Third
page~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
f3 = ttk.Frame(n, padding="12 12 12 12") # third page
f3.grid(column=0, row=0, sticky=(N, W, E, S))
f3.columnconfigure(0, weight=1)
f3.rowconfigure(0, weight=1)

sf3 = ttk.Frame(f3, borderwidth=5, relief=GROOVE)
sf3.grid(column=0,row=0,sticky=(N, W, E, S),pady=3)
sf3.columnconfigure(0, weight=1)
sf3.rowconfigure(0, weight=1)

s1f3 = ttk.Frame(f3, borderwidth=5, relief=GROOVE)
s1f3.grid(column=0,row=1,sticky=(N, W, E, S),pady=3)
s1f3.columnconfigure(0, weight=1)
s1f3.rowconfigure(0, weight=1)
```

```
mol = StringVar()
sfac = StringVar()
atom_entry = ttk.Entry(sf3, width=20, textvariable=mol)
atom_entry.grid(column=1, row=2,columnspan=2,sticky=W)
SFAC_entry = ttk.Entry(sf3, width=20, textvariable=sfac)
SFAC_entry.grid(column=1, row=3,columnspan=2,sticky=W)
ttk.Label(sf3, text="Structural information:",justify=LEFT).grid(column=0,
        row=0,columnspan=2,sticky=W,pady=4)
ttk.Label(sf3, text="Merge folder:", justify=LEFT).grid(column=0,row=1, sticky=W)
ttk.Label(sf3, text="Atoms present:", justify=LEFT).grid(column=0, row=2,sticky=W)
ttk.Label(sf3, text="Number of atoms:", justify=LEFT).grid(column=0, row=3,sticky=W)
ttk.Label(sf3, text="Run shelxd?", justify=LEFT).grid(column=0, row=4, sticky=W)
ttk.Label(sf3,text="                                ").grid(column=3,row=3,pady=3)
shelxd = StringVar()
shelxd.set("0")
radb1a = ttk.Radiobutton(sf3, text="no",variable=shelxd, value=0,command=s3)
radb2a = ttk.Radiobutton(sf3, text="yes",variable=shelxd, value=1, command=s3)
radb1a.grid(column=1,row=4,sticky=W)
radb2a.grid(column=2,row=4,sticky=W)
solv = StringVar()
combobox3 = ttk.Combobox(sf3,textvariable=solv,state='readonly',values=reind)
combobox3.grid(column=1,row=1,sticky=W,columnspan=2)
ttk.Label(s1f3, text="Solutions obtained:",justify=LEFT).grid(column=0, row=0,
        sticky=W)

solve_update = tk.Text(s1f3,height=22, width=70)
solve_update.grid(column=0,row=1, columnspan=3)
solve_update.configure(state='disabled')


ttk.Button(f3, text='Start', command=solve).grid(
          column=0, row=3, sticky=E)



#end of third page  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

n.add(f1, text='Processing')
n.add(f2, text='Merging')
n.add(f3, text='Solving')
n.select(f1)
n.enable_traversal()
root.bind('<Return>', main)
```

root.mainloop()

4.7.1.6 shelxt_library.py

```
import os

"""
crystallography library
"""
def library(number):
#triclinic
    global g
    global ins_sym
    if int(number) == 1:
        g = str("P1")
        ins_sym = str("LATT -1")
    if int(number) == 2:
        g = str("P-1")
        ins_sym = str("LATT -1")
#monoclinic
    if int(number) == 3:
        g = str("P2")
        ins_sym = str("LATT -1\nSYMM -X,Y,-Z")
    if int(number) == 4:
        g = str("P2/1")
        ins_sym = str("LATT -1\nSYMM -X,Y+1/2,-Z")
    if int(number) == 5:
        g = str("C2")
        ins_sym = str("LATT -7\nSYMM -X,Y,-Z")
    if int(number) == 6:
        g = str("Pm")
        ins_sym = str("LATT -1\nSYMM X,-Y,Z")
    if int(number) == 7:
        g = str("Pc")
        ins_sym = str("LATT -1\nSYMM X,-Y,Z+1/2")
    if int(number) == 8:
        g = str("Cm")
        ins_sym = str("LATT -7\nSYMM X,-Y,Z")
    if int(number) == 9:
        g = str("Cc")
        ins_sym = str("LATT -7\nSYMM X,-Y,Z+1/2")
    if int(number) == 10:
        g = str("P2_m")
        ins_sym = str("LATT -7\nSYMM X,-Y,Z+1/2")
    if int(number) == 11:
        g = str("P2/1_m")
```

```
        ins_sym = str("LATT 1\nSYMM -X,Y+1/2,-Z")
    if int(number) == 12:
        g = str("C2_m")
        ins_sym = str("LATT 7\nSYMM -X,Y,-Z")
    if int(number) == 13:
        g = str("P2_c")
        ins_sym = str("LATT 1\nSYMM -X,Y,-Z+1/2")
    if int(number) == 14:
        g = str("P2/1_c")
        ins_sym = str("LATT 1\nSYMM -X,Y+1/2,-Z+1/2")
    if int(number) == 15:
        g = str("C2_c")
        ins_sym = str("LATT 7\nSYMM -X,Y,-Z+1/2")
#orthorhombic
    if int(number) == 16:
        g = str("P222")
        ins_sym = str("LATT -1\nSYMM X,-Y,-Z\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z")
    if int(number) == 17:
        g = str("P222/1")
        ins_sym =str("LATT -1\nSYMM X,-Y,-Z\nSYMM -X,Y,-Z+1/2\nSYMM -X,-Y,Z+1/2")
    if int(number) == 18:
        g = str("P2/12/12")
        ins_sym = str("LATT -1\nSYMM X+1/2,-Y+1/2,-Z\nSYMM -X+1/2,Y+1/2,-Z" +
                "\nSYMM -X,-Y,Z")
    if int(number) == 19:
        g = str("P2/12/12/1")
        ins_sym = str("LATT -1\nSYMM X+1/2,-Y+1/2,-Z\nSYMM -X,Y+1/2,-Z+1/2" +
                "\nSYMM -X+1/2,-Y,Z+1/2")
    if int(number) == 20:
        g = str("C222/1")
        ins_sym = str("LATT -7\nSYMM X,-Y,-Z\nSYMM -X,Y,-Z+1/2\nSYMM -X,-Y,Z+1/2")
    if int(number) == 21:
        g = str("C222")
        ins_sym = str("LATT -7\nSYMM X,-Y,-Z\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z")
    if int(number) == 22:
        g = str("F222")
        ins_sym = str("LATT -4\nSYMM X,-Y,-Z\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z")
    if int(number) == 23:
        g = str("I222")
        ins_sym = str("LATT -2\nSYMM X,-Y,-Z\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z")
    if int(number) == 24:
        g = str("I2/12/12/1")
        ins_sym = str("LATT -2\nSYMM X,-Y,-Z+1/2\nSYMM -X+1/2,Y,-Z" +
                "\nSYMM -X,-Y+1/2,Z")
    if int(number) == 25:
        g = str("Pmm2")
```

```
        ins_sym = str("LATT -1\nSYMM -X,-Y,Z\nSYMM -X,Y,Z\nSYMM X,-Y,Z")
    if int(number) == 26:
        g = str("Pmc2/1")
        ins_sym = str("LATT -1\nSYMM -X,-Y,Z+1/2\nSYMM -X,Y,Z\nSYMM X,-Y,Z+1/2")
    if int(number) == 27:
        g = str("Pcc2")
        ins_sym = str("LATT -1\nSYMM -X,-Y,Z\nSYMM -X,Y,Z+1/2\nSYMM X,-Y,Z+1/2")
    if int(number) == 28:
        g = str("Pma2")
        ins_sym = str("LATT -1\nSYMM -X,-Y,Z\nSYMM -X+1/2,Y,Z\nSYMM X+1/2,-Y,Z")
    if int(number) == 29:
        g = str("Pca2/1")
        ins_sym = str("LATT -1\nSYMM -X,-Y,Z+1/2\nSYMM -X+1/2,Y,Z+1/2\nSYMM X+1/2,-
Y,Z")
    if int(number) == 30:
        g = str("Pnc2")
        ins_sym = str("LATT -1\nSYMM -X,-Y,Z\nSYMM -X,Y+1/2,Z+1/2\nSYMM X,-
Y+1/2,Z+1/2")
    if int(number) == 31:
        g = str("Pmn2/1")
        ins_sym = str("LATT -1\nSYMM -X+1/2,-Y,Z+1/2\nSYMM -X,Y,Z\nSYMM X+1/2,-
Y,Z+1/2")
    if int(number) == 32:
        g = str("Pba2")
        ins_sym = str("LATT -1\nSYMM -X,-Y,Z\nSYMM -X+1/2,Y+1/2,Z\nSYMM X+1/2,-
Y+1/2,Z")
    if int(number) == 33:
        g = str("Pna2/1")
        ins_sym = str("LATT -1\nSYMM -X,-Y,Z+1/2\nSYMM -X+1/2,Y+1/2,Z+1/2\nSYMM
X+1/2,-Y+1/2,Z")
    if int(number) == 34:
        g = str("Pnn2")
        ins_sym = str("LATT -1\nSYMM -X,-Y,Z\nSYMM -X+1/2,Y+1/2,Z+1/2\nSYMM X+1/2,-
Y+1/2,Z+1/2")
    if int(number) == 35:
        g = str("Cmm2")
        ins_sym = str("LATT -7\nSYMM -X,-Y,Z\nSYMM -X,Y,Z\nSYMM X,-Y,Z")
    if int(number) == 36:
        g = str("Cmc2/1")
        ins_sym = str("LATT -7\nSYMM -X,-Y,Z+1/2\nSYMM -X,Y,Z\nSYMM X,-Y,Z+1/2")
    if int(number) == 37:
        g = str("Ccc2")
        ins_sym = str("LATT -7\nSYMM -X,-Y,Z\nSYMM -X,Y,Z+1/2\nSYMM X,-Y,Z+1/2")
    if int(number) == 38:
        g = str("Amm2")
        ins_sym = str("LATT -5\nSYMM -X,-Y,Z\nSYMM -X,Y,Z\nSYMM X,-Y,Z")
```

```
    if int(number) == 39:
        g = str("Aem2")
        ins_sym = str("LATT -5\nSYMM -X,-Y,Z\nSYMM -X,Y+1/2,Z\nSYMM X,-Y+1/2,Z")
    if int(number) == 40:
        g = str("Ama2")
        ins_sym = str("LATT -5\nSYMM -X,-Y,Z\nSYMM -X+1/2,Y,Z\nSYMM X+1/2,-Y,Z")
    if int(number) == 41:
        g = str("Aea2")
        ins_sym = str("LATT -5\nSYMM -X,-Y,Z\nSYMM -X+1/2,Y+1/2,Z\nSYMM X+1/2,-
Y+1/2,Z")
    if int(number) == 42:
        g = str("Fmm2")
        ins_sym = str("LATT -4\nSYMM -X,-Y,Z\nSYMM -X,Y,Z\nSYMM X,-Y,Z")
    if int(number) == 43:
        g = str("Fdd2")
        ins_sym = str("LATT -4\nSYMM -X,-Y,Z\nSYMM -X+1/4,Y+1/4,Z+1/4" +
                  "\nSYMM X+1/4,-Y+1/4,Z+1/4")
    if int(number) == 44:
        g = str("Imm2")
        ins_sym = str("LATT -2\nSYMM -X,-Y,Z\nSYMM -X,Y,Z\nSYMM X,-Y,Z")
    if int(number) == 45:
        g = str("Iba2")
        ins_sym = str("LATT -2\nSYMM -X,-Y,Z\nSYMM -X,Y,Z+1/2\nSYMM X,-Y,Z+1/2")
    if int(number) == 46:
        g = str("Ima2")
        ins_sym = str("LATT -2\nSYMM -X,-Y,Z\nSYMM -X+1/2,Y,Z\nSYMM X+1/2,-Y,Z")
    if int(number) == 47:
        g = str("Pmmm")
        ins_sym = str("LATT 1\nSYMM X,-Y,-Z\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z")
    if int(number) == 48:
        g = str("Pnnn")
        ins_sym = str("LATT 1\nSYMM X,-Y+1/2,-Z+1/2\nSYMM -X+1/2,Y,-Z+1/2" +
                  "\nSYMM -X+1/2,-Y+1/2,Z")
    if int(number) == 49:
        g = str("Pccm")
        ins_sym = str("LATT 1\nSYMM X,-Y,-Z+1/2\nSYMM -X,Y,-Z+1/2\nSYMM -X,-Y,Z")
    if int(number) == 50:
        g = str("Pban")
        ins_sym = str("LATT 1\nSYMM X,-Y+1/2,-Z\nSYMM -X+1/2,Y,-Z" +
                  "\nSYMM -X+1/2,-Y+1/2,Z")
    if int(number) == 51:
        g = str("Pmma")
        ins_sym = str("LATT 1\nSYMM X+1/2,-Y,-Z\nSYMM -X,Y,-Z\nSYMM -X+1/2,-Y,Z")
    if int(number) == 52:
        g = str("Pnna")
        ins_sym = str("LATT 1\nSYMM X,-Y+1/2,-Z+1/2\nSYMM -X+1/2,Y+1/2,-Z+1/2" +
```

```
            "\nSYMM -X+1/2,-Y,Z")
if int(number) == 53:
   g = str("Pmna")
   ins_sym = str("LATT 1\nSYMM X,-Y,-Z\nSYMM -X+1/2,Y,-Z+1/2" +
            "\nSYMM -X+1/2,-Y,Z+1/2")
if int(number) == 54:
   g = str("Pcca")
   ins_sym = str("LATT 1\nSYMM X+1/2,-Y,-Z+1/2\nSYMM -X,Y,-Z+1/2" +
            "\nSYMM -X+1/2,-Y,Z")
if int(number) == 55:
   g = str("Pbam")
   ins_sym = str("LATT 1\nSYMM X+1/2,-Y+1/2,-Z\nSYMM -X+1/2,Y+1/2,-Z" +
            "\nSYMM -X,-Y,Z")
if int(number) == 56:
   g = str("Pccn")
   ins_sym = str("LATT 1\nSYMM X+1/2,-Y,-Z+1/2\nSYMM -X,Y+1/2,-Z+1/2" +
            "\nSYMM -X+1/2,-Y+1/2,Z")
if int(number) == 57:
   g = str("Pbcm")
   ins_sym = str("LATT 1\nSYMM X,-Y+1/2,-Z\nSYMM -X,Y+1/2,-Z+1/2" +
            "\nSYMM -X,-Y,Z+1/2")
if int(number) == 58:
   g = str("Pnnm")
   ins_sym = str("LATT 1\nSYMM X+1/2,-Y+1/2,-Z+1/2\nSYMM -X+1/2,Y+1/2,-Z+1/2"
            + "\nSYMM -X,-Y,Z")
if int(number) == 59:
   g = str("Pmmn")
   ins_sym = str("LATT 1\n X+1/2,-Y,-Z\nSYMM -X,Y+1/2,-Z" +
            "\nSYMM -X+1/2,-Y+1/2,Z")
if int(number) == 60:
   g = str("Pbcn")
   ins_sym = str("LATT 1\nSYMM X+1/2,-Y+1/2,-Z\nSYMM -X,Y,-Z+1/2" +
            "\nSYMM -X+1/2,-Y+1/2,Z+1/2")
if int(number) == 61:
   g = str("Pbca")
   ins_sym = str("LATT 1\nSYMM X+1/2,-Y+1/2,-Z\nSYMM -X,Y+1/2,-Z+1/2" +
            "\nSYMM -X+1/2,-Y,Z+1/2")
if int(number) == 62:
   g = str("Pnma")
   ins_sym = str("LATT 1\nSYMM X+1/2,-Y+1/2,-Z+1/2\nSYMM -X,Y+1/2,-Z" +
            "\nSYMM -X+1/2,-Y,Z+1/2")
if int(number) == 63:
   g = str("Cmcm")
   ins_sym = str("LATT 7\nSYMM X,-Y,-Z\nSYMM -X,Y,-Z+1/2" +
            "\nSYMM -X,-Y,Z+1/2")
if int(number) == 64:
```

```
        g = str("Cmca")
        ins_sym = str("LATT 7\nSYMM X,-Y,-Z\nSYMM -X+1/2,Y,-Z+1/2" +
                "\nSYMM -X+1/2,-Y,Z+1/2")
    if int(number) == 65:
        g = str("Cmmm")
        ins_sym = str("LATT 7\nSYMM X,-Y,-Z\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z")
    if int(number) == 66:
        g = str("Cccm")
        ins_sym = str("LATT 7\nSYMM X,-Y,-Z+1/2\nSYMM -X,Y,-Z+1/2" +
                "\nSYMM -X,-Y,Z")
    if int(number) == 67:
        g = str("Cmma")
        ins_sym = str("LATT 7\nSYMM X,-Y,-Z\nSYMM -X+1/2,Y,-Z" +
                "\nSYMM -X+1/2,-Y,Z")
    if int(number) == 68:
        g = str("Ccca")
        ins_sym = str("LATT 7\nSYMM X+1/2,-Y,-Z+1/2\nSYMM -X,Y,-Z+1/2" +
                "\nSYMM -X+1/2,-Y,Z")
    if int(number) == 69:
        g = str("Fmmm")
        ins_sym = str("LATT 4\nSYMM X,-Y,-Z\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z")
    if int(number) == 70:
        g = str("Fddd")
        ins_sym = str("LATT 4\nSYMM X,-Y+1/4,-Z+1/4\nSYMM -X+1/4,Y,-Z+1/4" +
                "\nSYMM -X+1/4,-Y+1/4,Z")
    if int(number) == 71:
        g = str("Immm")
        ins_sym = str("LATT 2\nSYMM X,-Y,-Z\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z")
    if int(number) == 72:
        g = str("Ibam")
        ins_sym = str("LATT 2\nSYMM X,-Y,-Z+1/2\nSYMM -X,Y,-Z+1/2\nSYMM -X,-Y,Z")
    if int(number) == 73:
        g = str("Ibca")
        ins_sym = str("LATT 2\nSYMM X,-Y,-Z+1/2\nSYMM -X+1/2,Y,-Z\nSYMM -X,-
Y+1/2,Z")
    if int(number) == 74:
        g = str("Imma")
        ins_sym = str("LATT 2\nSYMM X,-Y,-Z\nSYMM -X,Y+1/2,-Z\nSYMM -X,-Y+1/2,Z")
#tetragonal
    if int(number) == 75:
        g = str("P4")
        ins_sym = str("LATT -1\nSYMM -Y,X,Z\nSYMM Y,-X,Z\nSYMM -X,-Y,Z")
    if int(number) == 76:
        g = str("P4/1")
        ins_sym = str("LATT -1\nSYMM -Y,X,Z+1/4" +
                "\nSYMM Y,-X,Z+3/4\nSYMM -X,-Y,Z+1/2")
```

```
if int(number) == 77:
   g = str("P4/2")
   ins_sym = str("LATT -1\nSYMM -Y,X,Z+1/2\nSYMM Y,-X,Z+1/2" +
             "\nSYMM -X,-Y,Z")
if int(number) == 78:
   g = str("P4/3")
   ins_sym = str("LATT -1\nSYMM -Y,X,Z+3/4\nSYMM Y,-X,Z+1/4" +
             "\nSYMM -X,-Y,Z+1/2")
if int(number) == 79:
   g = str("I4")
   ins_sym = str("LATT -2\nSYMM -Y,X,Z\nSYMM Y,-X,Z\nSYMM -X,-Y,Z")
if int(number) == 80:
   g = str("I4/1")
   ins_sym = str("LATT -2\nSYMM -Y+1/2,X,Z+3/4\nSYMM Y+1/2,-X,Z+3/4" +
             "\nSYMM -X,-Y,Z")
if int(number) == 81:
   g = str("P-4")
   ins_sym = str("LATT -1\nSYMM Y,-X,-Z\nSYMM -Y,X,-Z\nSYMM -X,-Y,Z")
if int(number) == 82:
   g = str("I-4")
   ins_sym = str("LATT -2\nSYMM Y,-X,-Z\nSYMM -Y,X,-Z\nSYMM -X,-Y,Z")
if int(number) == 83:
   g = str("P4_m")
   ins_sym = str("LATT 1\nSYMM -Y,X,Z\nSYMM Y,-X,Z\nSYMM -X,-Y,Z")
if int(number) == 84:
   g = str("P4/2_m")
   ins_sym = str("LATT 1\nSYMM -Y,X,Z+1/2\nSYMM Y,-X,Z+1/2" +
             "\nSYMM -X,-Y,Z")
if int(number) == 85:
   g = str("P4_n")
   ins_sym = str("LATT 1\nSYMM -Y+1/2,X,Z\nSYMM Y,-X+1/2,Z" +
             "\nSYMM -X+1/2,-Y+1/2,Z")
if int(number) == 86:
   g = str("P4/2_n")
   ins_sym = str("LATT 1\nSYMM -Y,X+1/2,Z+1/2\nSYMM Y+1/2,-X,Z+1/2"
             "\nSYMM -X+1/2,-Y+1/2,Z")
if int(number) == 87:
   g = str("I4_m")
   ins_sym = str("LATT 2\nSYMM -Y,X,Z\nSYMM Y,-X,Z\nSYMM -X,-Y,Z")
if int(number) == 88:
   g = str("I4/1_a")
   ins_sym = str("LATT 2\nSYMM -Y+1/4,X+3/4,Z+3/4\nSYMM Y+1/4,-X+1/4,Z+1/4" +
             "\nSYMM -X,-Y+1/2,Z")
if int(number) == 89:
   g = str("P422")
   ins_sym = str("LATT -1\nSYMM -Y,X,Z\nSYMM Y,-X,Z\nSYMM X,-Y,-Z" +
```

```
            "\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z\nSYMM Y,X,-Z\nSYMM -Y,-X,-Z")
if int(number) == 90:
   g = str("P42(1)2")
   ins_sym = str("LATT -1\nSYMM -Y+1/2,X+1/2,Z\nSYMM Y+1/2,-X+1/2,Z" +
            "\nSYMM X+1/2,-Y+1/2,-Z\nSYMM -X+1/2,Y+1/2,-Z\nSYMM -X,-Y,Z"
            + "\nSYMM Y,X,-Z\nSYMM -Y,-X,-Z")
if int(number) == 91:
   g = str("P4(1)22")
   ins_sym = str("LATT -1\nSYMM -Y,X,Z+1/4\nSYMM Y,-X,Z+3/4" +
            "\nSYMM X,-Y,-Z+1/2\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z+1/2" +
            "SYMM Y,X,-Z+3/4\nSYMM -Y,-X,-Z+1/4")
if int(number) == 92:
   g = str("P4(1)2(1)2")
   ins_sym = str("LATT -1\nSYMM -Y+1/2,X+1/2,Z+1/4\nSYMM Y+1/2,-X+1/2,Z+3/4"
            + "\nSYMM X+1/2,-Y+1/2,-Z+3/4\nSYMM -X+1/2,Y+1/2,-Z+1/4" +
            "\nSYMM -X,-Y,Z+1/2\nSYMM Y,X,-Z\nSYMM -Y,-X,-Z+1/2")
if int(number) == 93:
   g = str("P4(2)22")
   ins_sym = str("LATT -1\nSYMM -Y,X,Z+1/2\nSYMM Y,-X,Z+1/2" +
            "\nSYMM X,-Y,-Z\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z" +
            "\nSYMM Y,X,-Z+1/2\nSYMM -Y,-X,-Z+1/2")
if int(number) == 94:
   g = str("P4(2)2(1)2")
   ins_sym = str("LATT -1\nSYMM -Y+1/2,X+1/2,Z+1/2\nSYMM Y+1/2,-X+1/2,Z+1/2" +
            "\nSYMM X+1/2,-Y+1/2,-Z+1/2\nSYMM -X+1/2,Y+1/2,-Z+1/2" +
            "\nSYMM -X,-Y,Z\nSYMM Y,X,-Z\nSYMM -Y,-X,-Z")
if int(number) == 95:
   g = str("P4(3)22")
   ins_sym = str("LATT -1\nSYMM -Y,X,Z+3/4\nSYMM Y,-X,Z+1/4" +
            "\nSYMM X,-Y,-Z+1/2\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z+1/2" +
            "\nSYMM Y,X,-Z+1/4\nSYMM -Y,-X,-Z+3/4")
if int(number) == 96:
   g = str("P4(3)2(1)2")
   ins_sym = str("LATT -1\nSYMM -Y+1/2,X+1/2,Z+3/4" +
            "\nSYMM Y+1/2,-X+1/2,Z+1/4\nSYMM X+1/2,-Y+1/2,-Z+1/4" +
            "\nSYMM -X+1/2,Y+1/2,-Z+3/4\nSYMM -X,-Y,Z+1/2" +
            "\nSYMM Y,X,-Z\nSYMM -Y,-X,-Z+1/2")
if int(number) == 97:
   g = str("I422")
   ins_sym = str("LATT -2\nSYMM -Y,X,Z\nSYMM Y,-X,Z\nSYMM X,-Y,-Z" +
            "\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z\nSYMM Y,X,-Z\nSYMM -Y,-X,-Z")
if int(number) == 98:
   g = str("I4(1)22")
   ins_sym = str("LATT -2\nSYMM -Y+1/2,X,Z+3/4\nSYMM Y+1/2,-X,Z+3/4" +
            "\nSYMM X+1/2,-Y,-Z+3/4\nSYMM -X+1/2,Y,-Z+3/4" +
            "\nSYMM -X,-Y,Z\nSYMM Y,X,-Z\nSYMM -Y,-X,-Z")
```

```
if int(number) == 99:
   g = str("P4mm")
   ins_sym = str("LATT -1\nSYMM -Y,X,Z\nSYMM Y,-X,Z\nSYMM -X,-Y,Z" +
             "\nSYMM -X,Y,Z\nSYMM X,-Y,Z\nSYMM -Y,-X,Z\nSYMM Y,X,Z")
if int(number) == 100:
   g = str("P4bm")
   ins_sym = str("LATT -1\nSYMM -Y,X,Z\nSYMM Y,-X,Z\nSYMM -X,-Y,Z" +
             "\nSYMM -X+1/2,Y+1/2,Z\nSYMM X+1/2,-Y+1/2,Z" +
             "\nSYMM -Y+1/2,-X+1/2,Z\nSYMM Y+1/2,X+1/2,Z")
if int(number) == 101:
   g = str("P4(2)cm")
   ins_sym = str("LATT -1\nSYMM -Y,X,Z+1/2\nSYMM Y,-X,Z+1/2" +
             "\nSYMM -X,-Y,Z\nSYMM -X,Y,Z+1/2\nSYMM X,-Y,Z+1/2" +
             "\nSYMM -Y,-X,Z\nSYMM Y,X,Z")
if int(number) == 102:
   g = str("P4(2)nm")
   ins_sym = str("LATT -1\nSYMM -Y+1/2,X+1/2,Z+1/2\nSYMM Y+1/2,-X+1/2,Z+1/2"
             + "\nSYMM -X,-Y,Z\nSYMM -X+1/2,Y+1/2,Z+1/2" +
             "\nSYMM X+1/2,-Y+1/2,Z+1/2\nSYMM -Y,-X,Z\nSYMM Y,X,Z")
if int(number) == 103:
   g = str("P4cc")
   ins_sym = str("LATT -1\nSYMM -Y,X,Z\nSYMM Y,-X,Z\nSYMM -X,-Y,Z" +
             "\nSYMM -X,Y,Z+1/2\nSYMM X,-Y,Z+1/2\nSYMM -Y,-X,Z+1/2" +
             "\nSYMM Y,X,Z+1/2")
if int(number) == 104:
   g = str("P4nc")
   ins_sym = str("LATT -1\nSYMM -Y,X,Z\nSYMM Y,-X,Z\nSYMM -X,-Y,Z" +
             "\nSYMM -X+1/2,Y+1/2,Z+1/2\nSYMM X+1/2,-Y+1/2,Z+1/2" +
             "\nSYMM -Y+1/2,-X+1/2,Z+1/2\nSYMM Y+1/2,X+1/2,Z+1/2")
if int(number) == 105:
   g = str("P4(2)mc")
   ins_sym = str("LATT -1\nSYMM -Y,X,Z+1/2\nSYMM Y,-X,Z+1/2\nSYMM -X,-Y,Z"
             + "\nSYMM -X,Y,Z\nSYMM X,-Y,Z\nSYMM -Y,-X,Z+1/2" +
             "\nSYMM Y,X,Z+1/2")
if int(number) == 106:
   g = str("P4(2)bc")
   ins_sym = str("LATT -1\nSYMM -Y,X,Z+1/2\nSYMM Y,-X,Z+1/2" +
             "\nSYMM -X,-Y,Z\nSYMM -X+1/2,Y+1/2,Z\nSYMM X+1/2,-Y+1/2,Z"
             + "\nSYMM -Y+1/2,-X+1/2,Z+1/2\nSYMM Y+1/2,X+1/2,Z+1/2")
if int(number) == 107:
   g = str("I4mm")
   ins_sym = str("LATT -2\nSYMM -Y,X,Z\nSYMM Y,-X,Z\nSYMM -X,-Y,Z" +
             "\nSYMM -X,Y,Z\nSYMM X,-Y,Z\nSYMM -Y,-X,Z\nSYMM Y,X,Z")
if int(number) == 108:
   g = str("I4cm")
   ins_sym = str("LATT -2\nSYMM -Y,X,Z\nSYMM Y,-X,Z\nSYMM -X,-Y,Z" +
```

```
                "\nSYMM -X,Y,Z+1/2\nSYMM X,-Y,Z+1/2\nSYMM -Y,-X,Z+1/2" +
                "\nSYMM Y,X,Z+1/2")
    if int(number) == 109:
        g = str("I4(1)md")
        ins_sym = str("LATT -2\nSYMM -Y+1/2,X,Z+3/4\nSYMM Y+1/2,-X,Z+3/4" +
                "\nSYMM -X,-Y,Z\nSYMM -X,Y,Z\nSYMM X,-Y,Z" +
                "\nSYMM -Y+1/2,-X,Z+3/4\nSYMM Y+1/2,X,Z+3/4")
    if int(number) == 110:
        g = str("I4(1)cd")
        ins_sym = str("LATT -2\nSYMM -Y+1/2,X,Z+3/4\nSYMM Y+1/2,-X,Z+3/4" +
                "\nSYMM -X,-Y,Z\nSYMM -X,Y,Z+1/2\nSYMM X,-Y,Z+1/2" +
                "\nSYMM -Y+1/2,-X,Z+1/4\nSYMM Y+1/2,X,Z+1/4")
    if int(number) == 111:
        g = str("P-42m")
        ins_sym = str("LATT -1\nSYMM Y,-X,-Z\nSYMM -Y,X,-Z\nSYMM X,-Y,-Z" +
                "\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z\nSYMM -Y,-X,Z\nSYMM Y,X,Z")
    if int(number) == 112:
        g = str("P-42c")
        ins_sym = str("LATT -1\nSYMM Y,-X,-Z\nSYMM -Y,X,-Z\nSYMM X,-Y,-Z+1/2" +
                "\nSYMM -X,Y,-Z+1/2\nSYMM -X,-Y,Z\nSYMM -Y,-X,Z+1/2" +
                "\nSYMM Y,X,Z+1/2")
    if int(number) == 113:
        g = str("P-42(1)m")
        ins_sym = str("LATT -1\nSYMM Y,-X,-Z\nSYMM -Y,X,-Z\nSYMM X+1/2,-Y+1/2,-Z"
                +"\nSYMM -X+1/2,Y+1/2,-Z\nSYMM -X,-Y,Z" +
                "\nSYMM -Y+1/2,-X+1/2,Z\nSYMM Y+1/2,X+1/2,Z")
    if int(number) == 114:
        g = str("P-42(1)c")
        ins_sym = str("LATT -1\nSYMM Y,-X,-Z\nSYMM -Y,X,-Z\nSYMM X+1/2,-Y+1/2,-
Z+1/2"
                + "\nSYMM -X+1/2,Y+1/2,-Z+1/2\nSYMM -X,-Y,Z" +
                "\nSYMM -Y+1/2,-X+1/2,Z+1/2\nSYMM Y+1/2,X+1/2,Z+1/2")
    if int(number) == 115:
        g = str("P-4m2")
        ins_sym = str("LATT -1\nSYMM Y,-X,-Z\nSYMM -Y,X,-Z\nSYMM -X,-Y,Z" +
                "\nSYMM Y,X,-Z\nSYMM -Y,-X,-Z\nSYMM -X,Y,Z\nSYMM X,-Y,Z")
    if int(number) == 116:
        g = str("P-4c2")
        ins_sym = str("LATT -1\nSYMM Y,-X,-Z\nSYMM -Y,X,-Z\nSYMM -X,-Y,Z" +
                "\nSYMM Y,X,-Z+1/2\nSYMM -Y,-X,-Z+1/2\nSYMM -X,Y,Z+1/2"
                + "\nSYMM X,-Y,Z+1/2")
    if int(number) == 117:
        g = str("P-4b2")
        ins_sym = str("LATT -1\nSYMM Y,-X,-Z\nSYMM -Y,X,-Z\nSYMM -X,-Y,Z" +
                "\nSYMM Y+1/2,X+1/2,-Z\nSYMM -Y+1/2,-X+1/2,-Z" +
                "\nSYMM -X+1/2,Y+1/2,Z\nSYMM X+1/2,-Y+1/2,Z")
```

```
if int(number) == 118:
   g = str("P-4n2")
   ins_sym = str("LATT -1\nSYMM Y,-X,-Z\nSYMM -Y,X,-Z\nSYMM -X,-Y,Z" +
             "\nSYMM Y+1/2,X+1/2,-Z+1/2\nSYMM -Y+1/2,-X+1/2,-Z+1/2" +
             "\nSYMM -X+1/2,Y+1/2,Z+1/2\nSYMM X+1/2,-Y+1/2,Z+1/2")
if int(number) == 119:
   g = str("I-4m2")
   ins_sym = str("LATT -2\nSYMM Y,-X,-Z\nSYMM -Y,X,-Z\nSYMM -X,-Y,Z" +
             "\nSYMM Y,X,-Z\nSYMM -Y,-X,-Z\nSYMM -X,Y,Z\nSYMM X,-Y,Z")
if int(number) == 120:
   g = str("I-4c2")
   ins_sym = str("LATT -2\nSYMM Y,-X,-Z\nSYMM -Y,X,-Z\nSYMM -X,-Y,Z" +
             "\nSYMM Y,X,-Z+1/2\nSYMM -Y,-X,-Z+1/2" +
             "\nSYMM -X,Y,Z+1/2\nSYMM X,-Y,Z+1/2")
if int(number) == 121:
   g = str("I-42m")
   ins_sym = str("LATT -2\nSYMM Y,-X,-Z\nSYMM -Y,X,-Z\nSYMM X,-Y,-Z" +
             "\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z\nSYMM -Y,-X,Z\nSYMM Y,X,Z")
if int(number) == 122:
   g = str("I-42d")
   ins_sym = str("LATT -2\nSYMM Y,-X,-Z\nSYMM -Y,X,-Z" +
             "\nSYMM X+1/2,-Y,-Z+3/4\nSYMM -X+1/2,Y,-Z+3/4" +
             "\nSYMM -X,-Y,Z\nSYMM -Y+1/2,-X,Z+3/4\nSYMM Y+1/2,X,Z+3/4")
if int(number) == 123:
   g = str("P4_mmm")
   ins_sym = str("LATT 1\nSYMM -Y,X,Z\nSYMM Y,-X,Z\nSYMM X,-Y,-Z" +
             "\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z\nSYMM Y,X,-Z\nSYMM -Y,-X,-Z")
if int(number) == 124:
   g = str("P4_mcc")
   ins_sym = str("LATT 1\nSYMM -Y,X,Z\nSYMM Y,-X,Z\nSYMM X,-Y,-Z+1/2" +
             "\nSYMM -X,Y,-Z+1/2\nSYMM -X,-Y,Z\nSYMM Y,X,-Z+1/2" +
             "\nSYMM -Y,-X,-Z+1/2")
if int(number) == 125:
   g = str("P4_nbm")
   ins_sym = str("LATT 1\nSYMM -Y+1/2,X,Z\nSYMM Y,-X+1/2,Z\nSYMM X,-Y+1/2,-Z"
             +"\nSYMM -X+1/2,Y,-Z\nSYMM -X+1/2,-Y+1/2,Z\nSYMM Y,X,-Z"+
             "\nSYMM -Y+1/2,-X+1/2,-Z")
if int(number) == 126:
   g = str("P4_nnc")
   ins_sym = str("LATT 1\nSYMM -Y+1/2,X,Z\nSYMM Y,-X+1/2,Z" +
             "\nSYMM X,-Y+1/2,-Z+1/2\nSYMM -X+1/2,Y,-Z+1/2" +
             "\nSYMM -X+1/2,-Y+1/2,Z\nSYMM Y,X,-Z+1/2" +
             "\nSYMM -Y+1/2,-X+1/2,-Z+1/2")
if int(number) == 127:
   g = str("P4_mbm")
   ins_sym = str("LATT 1\nSYMM -Y,X,Z\nSYMM Y,-X,Z\nSYMM X+1/2,-Y+1/2,-Z" +
```

```
                "\nSYMM -X+1/2,Y+1/2,-Z\nSYMM -X,-Y,Z" +
                "\nSYMM Y+1/2,X+1/2,-Z\nSYMM -Y+1/2,-X+1/2,-Z")
    if int(number) == 128:
        g = str("P4_mnc")
        ins_sym = str("LATT 1\nSYMM -Y,X,Z\nSYMM Y,-X,Z\nSYMM X+1/2,-Y+1/2,-Z+1/2"
                + "\nSYMM -X+1/2,Y+1/2,-Z+1/2\nSYMM -X,-Y,Z"
                + "\nSYMM Y+1/2,X+1/2,-Z+1/2\nSYMM -Y+1/2,-X+1/2,-Z+1/2")
    if int(number) == 129:
        g = str("P4_nmm")
        ins_sym = str("LATT 1\nSYMM -Y+1/2,X,Z\nSYMM Y,-X+1/2,Z\nSYMM X+1/2,-Y,-Z"
                +"\nSYMM -X,Y+1/2,-Z\nSYMM -X+1/2,-Y+1/2,Z" +
                "\nSYMM Y+1/2,X+1/2,-Z\nSYMM -Y,-X,-Z")
    if int(number) == 130:
        g = str("P4_ncc")
        ins_sym = str("LATT 1\nSYMM -Y+1/2,X,Z\nSYMM Y,-X+1/2,Z" +
                "\nSYMM X+1/2,-Y,-Z+1/2\nSYMM -X,Y+1/2,-Z+1/2" +
                "\nSYMM -X+1/2,-Y+1/2,Z\nSYMM Y+1/2,X+1/2,-Z+1/2" +
                "\nSYMM -Y,-X,-Z+1/2")
    if int(number) == 131:
        g = str("P4(2)_mmc")
        ins_sym = str("LATT 1\nSYMM -Y,X,Z+1/2\nSYMM Y,-X,Z+1/2\nSYMM X,-Y,-Z" +
                "\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z\nSYMM Y,X,-Z+1/2" +
                "\nSYMM -Y,-X,-Z+1/2")
    if int(number) == 132:
        g = str("P4(2)_mcm")
        ins_sym = str("LATT 1\nSYMM -Y,X,Z+1/2\nSYMM Y,-X,Z+1/2" +
                "\nSYMM X,-Y,-Z+1/2\nSYMM -X,Y,-Z+1/2\nSYMM -X,-Y,Z" +
                "\nSYMM Y,X,-Z\nSYMM -Y,-X,-Z")
    if int(number) == 133:
        g = str("P4(2)_nbc")
        ins_sym = str("LATT 1\nSYMM -Y+1/2,X,Z+1/2\nSYMM Y,-X+1/2,Z+1/2" +
                "\nSYMM X,-Y+1/2,-Z\nSYMM -X+1/2,Y,-Z\nSYMM -X+1/2,-Y+1/2,Z"
                "\nSYMM Y,X,-Z+1/2\nSYMM -Y+1/2,-X+1/2,-Z+1/2")
    if int(number) == 134:
        g = str("P4(2)_nnm")
        ins_sym = str("LATT 1\nSYMM -Y+1/2,X,Z+1/2\nSYMM Y,-X+1/2,Z+1/2" +
                "\nSYMM X,-Y+1/2,-Z+1/2\nSYMM -X+1/2,Y,-Z+1/2" +
                "\nSYMM -X+1/2,-Y+1/2,Z\nSYMM Y,X,-Z\nSYMM -Y+1/2,-X+1/2,-Z")
    if int(number) == 135:
        g = str("P4(2)_mbc")
        ins_sym = str("LATT 1\nSYMM -Y,X,Z+1/2\nSYMM Y,-X,Z+1/2\nSYMM X+1/2,-
Y+1/2,-Z"
                + "\nSYMM -X+1/2,Y+1/2,-Z\nSYMM -X,-Y,Z" +
                "\nSYMM Y+1/2,X+1/2,-Z+1/2\nSYMM -Y+1/2,-X+1/2,-Z+1/2")
    if int(number) == 136:
        g = str("P4(2)_mnm")
```

```
        ins_sym = str("LATT 1\nSYMM -Y+1/2,X+1/2,Z+1/2\nSYMM Y+1/2,-X+1/2,Z+1/2" +
                  "\nSYMM X+1/2,-Y+1/2,-Z+1/2\nSYMM -X+1/2,Y+1/2,-Z+1/2" +
                  "\nSYMM -X,-Y,Z\nSYMM Y,X,-Z\nSYMM -Y,-X,-Z")
    if int(number) == 137:
        g = str("P4(2)_nmc")
        ins_sym = str("LATT 1\nSYMM -Y+1/2,X,Z+1/2\nSYMM Y,-X+1/2,Z+1/2" +
                  "\nSYMM X+1/2,-Y,-Z\nSYMM -X,Y+1/2,-Z\nSYMM -X+1/2,-Y+1/2,Z"
                  + "\nSYMM Y+1/2,X+1/2,-Z+1/2\nSYMM -Y,-X,-Z+1/2")
    if int(number) == 138:
        g = str("P4(2)_ncm")
        ins_sym = str("LATT 1\nSYMM -Y+1/2,X,Z+1/2\nSYMM Y,-X+1/2,Z+1/2" +
                  "\nSYMM X+1/2,-Y,-Z+1/2\nSYMM -X,Y+1/2,-Z+1/2" +
                  "\nSYMM -X+1/2,-Y+1/2,Z\nSYMM Y+1/2,X+1/2,-Z" +
                  "\nSYMM -Y,-X,-Z")
    if int(number) == 139:
        g = str("I4_mmm")
        ins_sym = str("LATT 2\nSYMM -Y,X,Z\nSYMM Y,-X,Z\nSYMM X,-Y,-Z" +
                  "\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z\nSYMM Y,X,-Z\nSYMM -Y,-X,-Z")
    if int(number) == 140:
        g = str("I4_mcm")
        ins_sym = str("LATT 2\nSYMM -Y,X,Z\nSYMM Y,-X,ZSYMM X,-Y,-Z+1/2" +
                  "\nSYMM -X,Y,-Z+1/2\nSYMM -X,-Y,Z\nSYMM Y,X,-Z+1/2" +
                  "\nSYMM -Y,-X,-Z+1/2")
    if int(number) == 141:
        g = str("I4(1)_amd")
        ins_sym = str("LATT 2\nSYMM -Y+1/4,X+3/4,Z+1/4\nSYMM Y+1/4,-X+1/4,Z+3/4"
                  +"\nSYMM X,-Y,-Z\nSYMM -X,Y+1/2,-Z\nSYMM -X,-Y+1/2,Z" +
                  "\nSYMM Y+1/4,X+3/4,-Z+1/4\nSYMM -Y+1/4,-X+1/4,-Z+3/4")
    if int(number) == 142:
        g = str("I4(1)/acd")
        ins_sym = str("LATT 2\nSYMM -Y+1/4,X+3/4,Z+1/4\nSYMM Y+1/4,-X+1/4,Z+3/4"
                  +"\nSYMM X,-Y,-Z+1/2\nSYMM -X+1/2,Y,-Z\nSYMM -X,-Y+1/2,Z"+
                  "\nSYMM Y+1/4,X+3/4,-Z+3/4\nSYMM -Y+1/4,-X+1/4,-Z+1/4")
#trigonal
    if int(number) == 143:
        g = str("P3")
        ins_sym = str("LATT -1\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z")
    if int(number) == 144:
        g = str("P3_1")
        ins_sym = str("LATT -1\nSYMM -Y,X-Y,Z+1/3\nSYMM -X+Y,-X,Z+2/3")
    if int(number) == 145:
        g = str("P3_2")
        ins_sym = str("LATT -1\nSYMM -Y,X-Y,Z+2/3\nSYMM -X+Y,-X,Z+1/3")
    if int(number) == 146:
        g = str("R3")
        ins_sym = str("LATT -3\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z")
```

```
if int(number) == 147:
   g = str("P-3")
   ins_sym = str("LATT 1\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z")
if int(number) == 148:
   g = str("R-4")
   ins_sym = str("LATT 3\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z")
if int(number) == 149:
   g = str("P312")
   ins_sym = str("LATT -1\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM -Y,-X,-Z" +
             "\nSYMM -X+Y,Y,-Z\nSYMM X,X-Y,-Z")
if int(number) == 150:
   g = str("P321")
   ins_sym = str("LATT -1\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM X-Y,-Y,-Z"+
             "\nSYMM -X,-X+Y,-Z\nSYMM Y,X,-Z")
if int(number) == 151:
   g = str("P3(1)12")
   ins_sym = str("LATT -1\nSYMM -Y,X-Y,Z+1/3\nSYMM -X+Y,-X,Z+2/3" +
             "\nSYMM -Y,-X,-Z+2/3\nSYMM -X+Y,Y,-Z+1/3\nSYMM X,X-Y,-Z")
if int(number) == 152:
   g = str("P3(1)21")
   ins_sym = str("LATT -1\nSYMM -Y,X-Y,Z+1/3\nSYMM -X+Y,-X,Z+2/3" +
             "\nSYMM X-Y,-Y,-Z+2/3\nSYMM -X,-X+Y,-Z+1/3\nSYMM Y,X,-Z")
if int(number) == 153:
   g = str("P3(2)12")
   ins_sym = str("LATT -1\nSYMM -Y,X-Y,Z+2/3\nSYMM -X+Y,-X,Z+1/3" +
             "\nSYMM -Y,-X,-Z+1/3\nSYMM -X+Y,Y,-Z+2/3" +
             "\nSYMM X,X-Y,-Z")
if int(number) == 154:
   g = str("P3(2)21")
   ins_sym = str("LATT -1\nSYMM -Y,X-Y,Z+2/3\nSYMM -X+Y,-X,Z+1/3" +
             "\nSYMM X-Y,-Y,-Z+1/3\nSYMM -X,-X+Y,-Z+2/3" +
             "\nSYMM Y,X,-Z")
if int(number) == 155:
   g = str("R32")
   ins_sym = str("LATT -3\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z" +
             "\nSYMM X-Y,-Y,-Z\nSYMM -X,-X+Y,-Z\nSYMM Y,X,-Z")
if int(number) == 156:
   g = str("P3m1")
   ins_sym = str("LATT -1\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM -X+Y,Y,Z" +
             "\nSYMM X,X-Y,Z\nSYMM -Y,-X,Z")
if int(number) == 157:
   g = str("P31m")
   ins_sym = str("LATT -1\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM Y,X,Z" +
             "\nSYMM X-Y,-Y,Z\nSYMM -X,-X+Y,Z")
if int(number) == 158:
   g = str("P3c1")
```

```
        ins_sym = str("LATT -1\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z" +
                "\nSYMM -X+Y,Y,Z+1/2\nSYMM X,X-Y,Z+1/2\nSYMM -Y,-X,Z+1/2")
    if int(number) == 159:
        g = str("P31c")
        ins_sym = str("LATT -1\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM Y,X,Z+1/2"+
                "\nSYMM X-Y,-Y,Z+1/2\nSYMM -X,-X+Y,Z+1/2")
    if int(number) == 160:
        g = str("R3m")
        ins_sym = str("LATT -3\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM -X+Y,Y,Z" +
                "\nSYMM X,X-Y,Z\nSYMM -Y,-X,Z")
    if int(number) == 161:
        g = str("R3c")
        ins_sym = str("LATT -3\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM -X+Y,Y,Z+1/2"
                + "\nSYMM X,X-Y,Z+1/2\nSYMM -Y,-X,Z+1/2")
    if int(number) == 162:
        g = str("P-31m")
        ins_sym = str("LATT 1\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM -Y,-X,-Z" +
                "\nSYMM -X+Y,Y,-Z\nSYMM X,X-Y,-Z")
    if int(number) == 163:
        g = str("P-31c")
        ins_sym = str("LATT 1\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM -Y,-X,-Z+1/2"
                + "\nSYMM -X+Y,Y,-Z+1/2\nSYMM X,X-Y,-Z+1/2")
    if int(number) == 164:
        g = str("P-3m1")
        ins_sym = str("LATT 1\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM X-Y,-Y,-Z" +
                "\nSYMM -X,-X+Y,-Z\nSYMM Y,X,-Z")
    if int(number) == 165:
        g = str("P-3c1")
        ins_sym = str("LATT 1\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM X-Y,-Y,-Z+1/2"
                + "\nSYMM -X,-X+Y,-Z+1/2\nSYMM Y,X,-Z+1/2")
    if int(number) == 166:
        g = str("R-3m")
        ins_sym = str("LATT 3\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM X-Y,-Y,-Z" +
                "\nSYMM -X,-X+Y,-Z\nSYMM Y,X,-Z")
    if int(number) == 167:
        g = str("R-3c")
        ins_sym = str("LATT 3\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM X-Y,-Y,-Z+1/2"
                + "\nSYMM -X,-X+Y,-Z+1/2\nSYMM Y,X,-Z+1/2")
#hexagonal
    if int(number) == 168:
        g = str("P6")
        ins_sym = str("LATT -1\nSYMM X-Y,X,Z\nSYMM Y,-X+Y,Z\nSYMM -Y,X-Y,Z" +
                "\nSYMM -X+Y,-X,Z\nSYMM -X,-Y,Z")
    if int(number) == 169:
        g = str("P6(1)")
        ins_sym = str("LATT -1\nSYMM X-Y,X,Z+1/6\nSYMM Y,-X+Y,Z+5/6" +
```

```
            "\nSYMM -Y,X-Y,Z+1/3\nSYMM -X+Y,-X,Z+2/3" +
            "\nSYMM -X,-Y,Z+1/2")
if int(number) == 170:
   g = str("P6(5)")
   ins_sym = str("LATT -1\nSYMM X-Y,X,Z+5/6\nSYMM Y,-X+Y,Z+1/6" +
            "\nSYMM -Y,X-Y,Z+2/3\nSYMM -X+Y,-X,Z+1/3" +
            "\nSYMM -X,-Y,Z+1/2")
if int(number) == 171:
   g = str("P6(2)")
   ins_sym = str("LATT -1\nSYMM X-Y,X,Z+1/3\nSYMM Y,-X+Y,Z+2/3" +
            "\nSYMM -Y,X-Y,Z+2/3\nSYMM -X+Y,-X,Z+1/3\nSYMM -X,-Y,Z")
if int(number) == 172:
   g = str("P6(4)")
   ins_sym = str("LATT -1\nSYMM X-Y,X,Z+2/3\nSYMM Y,-X+Y,Z+1/3" +
            "\nSYMM -Y,X-Y,Z+1/3\nSYMM -X+Y,-X,Z+2/3\nSYMM -X,-Y,Z")
if int(number) == 173:
   g = str("P6(3)")
   ins_sym = str("LATT -1\nSYMM X-Y,X,Z+1/2\nSYMM Y,-X+Y,Z+1/2" +
            "\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM -X,-Y,Z+1/2")
if int(number) == 174:
   g = str("P-6")
   ins_sym = str("LATT -1\nSYMM -X+Y,-X,-Z\nSYMM -Y,X-Y,-Z" +
            "\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM X,Y,-Z")
if int(number) == 175:
   g = str("P6_m")
   ins_sym = str("LATT 1\nSYMM X-Y,X,Z\nSYMM Y,-X+Y,Z\nSYMM -Y,X-Y,Z" +
            "\nSYMM -X+Y,-X,Z\nSYMM -X,-Y,Z")
if int(number) == 176:
   g = str("P6(3)_m")
   ins_sym = str("LATT 1\nSYMM X-Y,X,Z+1/2\nSYMM Y,-X+Y,Z+1/2" +
            "\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM -X,-Y,Z+1/2")
if int(number) == 177:
   g = str("P622")
   ins_sym = str("LATT -1\nSYMM X-Y,X,Z\nSYMM Y,-X+Y,Z\nSYMM -Y,X-Y,Z" +
            "\nSYMM -X+Y,-X,Z\nSYMM X-Y,-Y,-Z\nSYMM -X,-X+Y,-Z" +
            "\nSYMM -X,-Y,Z\nSYMM Y,X,-Z\nSYMM -Y,-X,-Z" +
            "\nSYMM -X+Y,Y,-Z\nSYMM X,X-Y,-Z")
if int(number) == 178:
   g = str("P6(1)22")
   ins_sym = str("LATT -1\nSYMM X-Y,X,Z+1/6\nSYMM Y,-X+Y,Z+5/6" +
            "\nSYMM -Y,X-Y,Z+1/3\nSYMM -X+Y,-X,Z+2/3" +
            "\nSYMM X-Y,-Y,-Z\nSYMM -X,-X+Y,-Z+2/3\nSYMM -X,-Y,Z+1/2"+
            "\nSYMM Y,X,-Z+1/3\nSYMM -Y,-X,-Z+5/6" +
            "\nSYMM -X+Y,Y,-Z+1/2\nSYMM X,X-Y,-Z+1/6")
if int(number) == 179:
   g = str("P6(5)22")
```

```
    ins_sym = str("LATT -1\nSYMM X-Y,X,Z+5/6\nSYMM Y,-X+Y,Z+1/6" +
                  "\nSYMM -Y,X-Y,Z+2/3\nSYMM -X+Y,-X,Z+1/3\nSYMM X-Y,-Y,-Z"+
                  "\nSYMM -X,-X+Y,-Z+1/3\nSYMM -X,-Y,Z+1/2\nSYMM Y,X,-Z+2/3"+
                  "\nSYMM -Y,-X,-Z+1/6\nSYMM -X+Y,Y,-Z+1/2" +
                  "\nSYMM X,X-Y,-Z+5/6")
if int(number) == 180:
    g = str("P6(2)22")
    ins_sym = str("LATT -1\nSYMM X-Y,X,Z+1/3\nSYMM Y,-X+Y,Z+2/3" +
                  "\nSYMM -Y,X-Y,Z+2/3\nSYMM -X+Y,-X,Z+1/3\nSYMM X-Y,-Y,-Z"+
                  "\nSYMM -X,-X+Y,-Z+1/3\nSYMM -X,-Y,Z\nSYMM Y,X,-Z+2/3" +
                  "\nSYMM -Y,-X,-Z+2/3\nSYMM -X+Y,Y,-Z\nSYMM X,X-Y,-Z+1/3")
if int(number) == 181:
    g = str("P6(4)22")
    ins_sym = str("LATT -1\nSYMM X-Y,X,Z+2/3\nSYMM Y,-X+Y,Z+1/3" +
                  "\nSYMM -Y,X-Y,Z+1/3\nSYMM -X+Y,-X,Z+2/3\nSYMM X-Y,-Y,-Z"+
                  "\nSYMM -X,-X+Y,-Z+2/3\nSYMM -X,-Y,Z\nSYMM Y,X,-Z+1/3" +
                  "\nSYMM -Y,-X,-Z+1/3\nSYMM -X+Y,Y,-Z\nSYMM X,X-Y,-Z+2/3")
if int(number) == 182:
    g = str("P6(3)22")
    ins_sym = str("LATT -1\nSYMM X-Y,X,Z+1/2\nSYMM Y,-X+Y,Z+1/2" +
                  "\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM X-Y,-Y,-Z" +
                  "\nSYMM -X,-X+Y,-Z\nSYMM -X,-Y,Z+1/2\nSYMM Y,X,-Z" +
                  "\nSYMM -Y,-X,-Z+1/2\nSYMM -X+Y,Y,-Z+1/2\nSYMM X,X-Y,-Z+1/2")
if int(number) == 183:
    g = str("P6mm")
    ins_sym = str("LATT -1\nSYMM X-Y,X,Z\nSYMM Y,-X+Y,Z\nSYMM -Y,X-Y,Z" +
                  "\nSYMM -X+Y,-X,Z\nSYMM -X,-Y,Z\nSYMM -X+Y,Y,Z" +
                  "\nSYMM X,X-Y,Z\nSYMM -Y,-X,Z\nSYMM Y,X,Z" +
                  "\nSYMM X-Y,-Y,Z\nSYMM -X,-X+Y,Z")
if int(number) == 184:
    g = str("P6cc")
    ins_sym = str("LATT -1\nSYMM X-Y,X,Z\nSYMM Y,-X+Y,Z\nSYMM -Y,X-Y,Z" +
                  "\nSYMM -X+Y,-X,Z\nSYMM -X,-Y,Z\nSYMM -X+Y,Y,Z+1/2" +
                  "\nSYMM X,X-Y,Z+1/2\nSYMM -Y,-X,Z+1/2\nSYMM Y,X,Z+1/2" +
                  "\nSYMM X-Y,-Y,Z+1/2\nSYMM -X,-X+Y,Z+1/2")
if int(number) == 185:
    g = str("P6(3)cm")
    ins_sym = str("LATT -1\nSYMM X-Y,X,Z+1/2\nSYMM Y,-X+Y,Z+1/2" +
                  "\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM -X,-Y,Z+1/2" +
                  "\nSYMM -X+Y,Y,Z+1/2\nSYMM X,X-Y,Z+1/2\nSYMM -Y,-X,Z+1/2"
                  +"\nSYMM Y,X,Z\nSYMM X-Y,-Y,Z\nSYMM -X,-X+Y,Z")
if int(number) == 186:
    g = str("P6(3)mc")
    ins_sym = str("LATT -1\nSYMM X-Y,X,Z+1/2\nSYMM Y,-X+Y,Z+1/2" +
                  "\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM -X,-Y,Z+1/2" +
                  "\nSYMM -X+Y,Y,Z\nSYMM X,X-Y,Z\nSYMM -Y,-X,Z" +
```

```
            "\nSYMM Y,X,Z+1/2\nSYMM X-Y,-Y,Z+1/2\nSYMM -X,-X+Y,Z+1/2")
if int(number) == 187:
   g = str("P-6m2")
   ins_sym = str("LATT -1\nSYMM -X+Y,-X,-Z\nSYMM -Y,X-Y,-Z" +
            "\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM -Y,-X,-Z" +
            "\nSYMM -X+Y,Y,-Z\nSYMM X,X-Y,-Z\nSYMM -X+Y,Y,Z" +
            "\nSYMM X,X-Y,Z\nSYMM X,Y,-Z\nSYMM -Y,-X,Z")
if int(number) == 188:
   g = str("P-6c2")
   ins_sym = str("LATT -1\nSYMM -X+Y,-X,-Z+1/2\nSYMM -Y,X-Y,-Z+1/2" +
            "\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM -Y,-X,-Z" +
            "\nSYMM -X+Y,Y,-Z\nSYMM X,X-Y,-Z\nSYMM -X+Y,Y,Z+1/2" +
            "\nSYMM X,X-Y,Z+1/2\nSYMM X,Y,-Z+1/2\nSYMM -Y,-X,Z+1/2")
if int(number) == 189:
   g = str("P-62m")
   ins_sym = str("LATT -1\nSYMM -X+Y,-X,-Z\nSYMM -Y,X-Y,-Z\nSYMM -Y,X-Y,Z"+
            "\nSYMM -X+Y,-X,Z\nSYMM X-Y,-Y,-Z\nSYMM -X,-X+Y,-Z" +
            "\nSYMM Y,X,-Z\nSYMM X,Y,-Z\nSYMM Y,X,Z\nSYMM X-Y,-Y,Z" +
            "\nSYMM -X,-X+Y,Z")
if int(number) == 190:
   g = str("P-62c")
   ins_sym = str("LATT -1\nSYMM -X+Y,-X,-Z+1/2\nSYMM -Y,X-Y,-Z+1/2" +
            "\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM X-Y,-Y,-Z" +
            "\nSYMM -X,-X+Y,-Z\nSYMM Y,X,-Z\nSYMM X,Y,-Z+1/2" +
            "\nSYMM Y,X,Z+1/2\nSYMM X-Y,-Y,Z+1/2\nSYMM -X,-X+Y,Z+1/2")
if int(number) == 191:
   g = str("P6_mmm")
   ins_sym = str("LATT 1\nSYMM X-Y,X,Z\nSYMM Y,-X+Y,Z\nSYMM -Y,X-Y,Z" +
            "\nSYMM -X+Y,-X,Z\nSYMM X-Y,-Y,-Z\nSYMM -X,-X+Y,-Z" +
            "\nSYMM -X,-Y,Z\nSYMM Y,X,-Z\nSYMM -Y,-X,-Z" +
            "\nSYMM -X+Y,Y,-Z\nSYMM X,X-Y,-Z")
if int(number) == 192:
   g = str("P6_mcc")
   ins_sym = str("LATT 1\nSYMM X-Y,X,Z\nSYMM Y,-X+Y,Z\nSYMM -Y,X-Y,Z" +
            "\nSYMM -X+Y,-X,Z\nSYMM X-Y,-Y,-Z+1/2" +
            "\nSYMM -X,-X+Y,-Z+1/2\nSYMM -X,-Y,Z\nSYMM Y,X,-Z+1/2" +
            "\nSYMM -Y,-X,-Z+1/2\nSYMM -X+Y,Y,-Z+1/2\nSYMM X,X-Y,-Z+1/2")
if int(number) == 193:
   g = str("P6_mcm")
   ins_sym = str("LATT 1\nSYMM X-Y,X,Z+1/2\nSYMM Y,-X+Y,Z+1/2" +
            "\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM X-Y,-Y,-Z+1/2" +
            "\nSYMM -X,-X+Y,-Z+1/2\nSYMM -X,-Y,Z+1/2\nSYMM Y,X,-Z+1/2"+
            "\nSYMM -Y,-X,-Z\nSYMM -X+Y,Y,-Z\nSYMM X,X-Y,-Z")
if int(number) == 194:
   g = str("P6_mmc")
   ins_sym = str("LATT 1\nSYMM X-Y,X,Z+1/2\nSYMM Y,-X+Y,Z+1/2" +
```

```
                "\nSYMM -Y,X-Y,Z\nSYMM -X+Y,-X,Z\nSYMM X-Y,-Y,-Z" +
                "\nSYMM -X,-X+Y,-Z\nSYMM -X,-Y,Z+1/2\nSYMM Y,X,-Z" +
                "\nSYMM -Y,-X,-Z+1/2\nSYMM -X+Y,Y,-Z+1/2\nSYMM X,X-Y,-Z+1/2")
#cubic
   if int(number) == 195:
      g = str("P23")
      ins_sym = str("LATT -1\nSYMM Z,X,Y\nSYMM Y,Z,X\nSYMM -Y,-Z,X" +
                "\nSYMM Z,-X,-Y\nSYMM -Y,Z,-X\nSYMM -Z,-X,Y\nSYMM -Z,X,-Y" +
                "\nSYMM Y,-Z,-X\nSYMM X,-Y,-Z\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z")
   if int(number) == 196:
      g = str("F23")
      ins_sym = str("LATT -4\nSYMM Z,X,Y\nSYMM Y,Z,X\nSYMM -Y,-Z,X" +
                "\nSYMM Z,-X,-Y\nSYMM -Y,Z,-X\nSYMM -Z,-X,Y\nSYMM -Z,X,-Y" +
                "\nSYMM Y,-Z,-X\nSYMM X,-Y,-Z\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z")
   if int(number) == 197:
      g = str("I23")
      ins_sym = str("LATT -2\nSYMM Z,X,Y\nSYMM Y,Z,X\nSYMM -Y,-Z,X" +
                "\nSYMM Z,-X,-Y\SYMM -Y,Z,-X\nSYMM -Z,-X,Y\nSYMM -Z,X,-Y" +
                "\nSYMM Y,-Z,-X\nSYMM X,-Y,-Z\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z")
   if int(number) == 198:
      g = str("P2(1)3")
      ins_sym = str("LATT -1\nSYMM Z,X,Y\nSYMM Y,Z,X\nSYMM -Y+1/2,-Z,X+1/2" +
                "\nSYMM Z+1/2,-X+1/2,-Y\nSYMM -Y,Z+1/2,-X+1/2" +
                "\nSYMM -Z+1/2,-X,Y+1/2\nSYMM -Z,X+1/2,-Y+1/2" +
                "\nSYMM Y+1/2,-Z+1/2,-X\nSYMM X+1/2,-Y+1/2,-Z" +
                "\nSYMM -X,Y+1/2,-Z+1/2\nSYMM -X+1/2,-Y,Z+1/2")
   if int(number) == 199:
      g = str("I2(1)3")
      ins_sym = str("LATT -2\nSYMM Z,X,Y\nSYMM Y,Z,X\nSYMM -Y,-Z+1/2,X" +
                "\nSYMM Z,-X,-Y+1/2\nSYMM -Y+1/2,Z,-X\nSYMM -Z,-X+1/2,Y" +
                "\nSYMM -Z+1/2,X,-Y\nSYMM Y,-Z,-X+1/2\nSYMM X,-Y,-Z+1/2" +
                "\nSYMM -X+1/2,Y,-Z\nSYMM -X,-Y+1/2,Z")
   if int(number) == 200:
      g = str("Pm-3")
      ins_sym = str("LATT 1\nSYMM Z,X,Y\nSYMM Y,Z,X\nSYMM -Y,-Z,X\nSYMM Z,-X,-
Y"+
                "\nSYMM -Y,Z,-X\nSYMM -Z,-X,Y\nSYMM -Z,X,-Y\nSYMM Y,-Z,-X" +
                "\nSYMM X,-Y,-Z\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z")
   if int(number) == 201:
      g = str("Pn-3")
      ins_sym = str("LATT 1\nSYMM Z,X,Y\nSYMM Y,Z,X\nSYMM -Y+1/2,-Z+1/2,X" +
                "\nSYMM Z,-X+1/2,-Y+1/2\nSYMM -Y+1/2,Z,-X+1/2" +
                "\nSYMM -Z+1/2,-X+1/2,Y\nSYMM -Z+1/2,X,-Y+1/2" +
                "\nSYMM Y,-Z+1/2,-X+1/2\nSYMM X,-Y+1/2,-Z+1/2" +
                "\nSYMM -X+1/2,Y,-Z+1/2\nSYMM -X+1/2,-Y+1/2,Z")
   if int(number) == 202:
```

```
    g = str("Fm-3")
    ins_sym = str("LATT 4\nSYMM Z,X,Y\nSYMM Y,Z,X\nSYMM -Y,-Z,X" +
                  "\nSYMM Z,-X,-Y\nSYMM -Y,Z,-X\nSYMM -Z,-X,Y" +
                  "\nSYMM -Z,X,-Y\nSYMM Y,-Z,-X\nSYMM X,-Y,-Z" +
                  "\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z")
if int(number) == 203:
    g = str("Fd-3")
    ins_sym = str("LATT 4\nSYMM Z,X,Y\nSYMM Y,Z,X\nSYMM -Y+1/4,-Z+1/4,X" +
                  "\nSYMM Z,-X+1/4,-Y+1/4\nSYMM -Y+1/4,Z,-X+1/4" +
                  "\nSYMM -Z+1/4,-X+1/4,Y\nSYMM -Z+1/4,X,-Y+1/4" +
                  "\nSYMM Y,-Z+1/4,-X+1/4\nSYMM X,-Y+1/4,-Z+1/4" +
                  "\nSYMM -X+1/4,Y,-Z+1/4\nSYMM -X+1/4,-Y+1/4,Z")
if int(number) == 204:
    g = str("Im-3")
    ins_sym = str("LATT 2\nSYMM Z,X,Y\nSYMM Y,Z,X\nSYMM -Y,-Z,X" +
                  "\nSYMM Z,-X,-Y\nSYMM -Y,Z,-X\nSYMM -Z,-X,Y" +
                  "\nSYMM -Z,X,-Y\nSYMM Y,-Z,-X\nSYMM X,-Y,-Z" +
                  "\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z")
if int(number) == 205:
    g = str("Pa-3")
    ins_sym = str("LATT 1\nSYMM Z,X,Y\nSYMM Y,Z,X\nSYMM -Y+1/2,-Z,X+1/2" +
                  "\nSYMM Z+1/2,-X+1/2,-Y\nSYMM -Y,Z+1/2,-X+1/2" +
                  "\nSYMM -Z+1/2,-X,Y+1/2\nSYMM -Z,X+1/2,-Y+1/2" +
                  "\nSYMM Y+1/2,-Z+1/2,-X\nSYMM X+1/2,-Y+1/2,-Z" +
                  "\nSYMM -X,Y+1/2,-Z+1/2\nSYMM -X+1/2,-Y,Z+1/2")
if int(number) == 206:
    g = str("Ia-3")
    ins_sym = str("LATT 2\nSYMM Z,X,Y\nSYMM Y,Z,X\nSYMM -Y,-Z+1/2,X" +
                  "\nSYMM Z,-X,-Y+1/2\nSYMM -Y+1/2,Z,-X\nSYMM -Z,-X+1/2,Y" +
                  "\nSYMM -Z+1/2,X,-Y\nSYMM Y,-Z,-X+1/2\nSYMM X,-Y,-Z+1/2" +
                  "\nSYMM -X+1/2,Y,-Z\nSYMM -X,-Y+1/2,Z")
if int(number) == 207:
    g = str("P432")
    ins_sym = str("LATT -1\nSYMM X,-Z,Y\nSYMM X,Z,-Y\nSYMM Z,Y,-X" +
                  "\nSYMM -Z,Y,X\nSYMM -Y,X,Z\nSYMM Y,-X,Z\nSYMM Z,X,Y" +
                  "\nSYMM Y,Z,X\nSYMM -Y,-Z,X\nSYMM Z,-X,-Y\nSYMM -Y,Z,-X" +
                  "\nSYMM -Z,-X,Y\nSYMM -Z,X,-Y\nSYMM Y,-Z,-X\nSYMM X,-Y,-Z" +
                  "\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z\nSYMM Y,X,-Z\nSYMM -Y,-X,-Z" +
                  "\nSYMM Z,-Y,X\nSYMM -Z,-Y,-X\nSYMM -X,Z,Y\nSYMM -X,-Z,-Y")
if int(number) == 208:
    g = str("P4(2)32")
    ins_sym = str("LATT -1\nSYMM X+1/2,-Z+1/2,Y+1/2" +
                  "\nSYMM X+1/2,Z+1/2,-Y+1/2\nSYMM Z+1/2,Y+1/2,-X+1/2" +
                  "\nSYMM -Z+1/2,Y+1/2,X+1/2\nSYMM -Y+1/2,X+1/2,Z+1/2" +
                  "\nSYMM Y+1/2,-X+1/2,Z+1/2\nSYMM Z,X,Y\nSYMM Y,Z,X" +
                  "\nSYMM -Y,-Z,X\nSYMM Z,-X,-Y\nSYMM -Y,Z,-X\nSYMM -Z,-X,Y" +
```

```
                "\nSYMM -Z,X,-Y\nSYMM Y,-Z,-X\nSYMM X,-Y,-Z" +
                "\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z\nSYMM Y+1/2,X+1/2,-Z+1/2" +
                "\nSYMM -Y+1/2,-X+1/2,-Z+1/2\nSYMM Z+1/2,-Y+1/2,X+1/2" +
                "\nSYMM -Z+1/2,-Y+1/2,-X+1/2\nSYMM -X+1/2,Z+1/2,Y+1/2" +
                "\nSYMM -X+1/2,-Z+1/2,-Y+1/2")
    if int(number) == 209:
        g = str("F432")
        ins_sym = str("LATT -4\nSYMM X,-Z,Y\nSYMM X,Z,-Y\nSYMM Z,Y,-X" +
                "\nSYMM -Z,Y,X\nSYMM -Y,X,Z\nSYMM Y,-X,Z\nSYMM Z,X,Y" +
                "\nSYMM Y,Z,X\nSYMM -Y,-Z,X\nSYMM Z,-X,-Y\nSYMM -Y,Z,-X" +
                "\nSYMM -Z,-X,Y\nSYMM -Z,X,-Y\nSYMM Y,-Z,-X\nSYMM X,-Y,-Z" +
                "\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z\nSYMM Y,X,-Z\nSYMM -Y,-X,-Z" +
                "\nSYMM Z,-Y,X\nSYMM -Z,-Y,-X\nSYMM -X,Z,Y\nSYMM -X,-Z,-Y")
    if int(number) == 210:
        g = str("F4(1)32")
        ins_sym = str("LATT -4\nSYMM X+1/4,-Z+1/4,Y+1/4" +
                "\nSYMM X+1/4,Z+1/4,-Y+1/4\nSYMM Z+1/4,Y+1/4,-X+1/4" +
                "\nSYMM -Z+1/4,Y+1/4,X+1/4\nSYMM -Y+1/4,X+1/4,Z+1/4" +
                "\nSYMM Y+1/4,-X+1/4,Z+1/4\nSYMM Z,X,Y\nSYMM Y,Z,X" +
                "\nSYMM -Y,-Z,X\nSYMM Z,-X,-Y\nSYMM -Y,Z,-X\nSYMM -Z,-X,Y" +
                "\nSYMM -Z,X,-Y\nSYMM Y,-Z,-X\nSYMM X,-Y,-Z\nSYMM -X,Y,-Z" +
                "\nSYMM -X,-Y,Z\nSYMM Y+1/4,X+1/4,-Z+1/4" +
                "\nSYMM -Y+1/4,-X+1/4,-Z+1/4\nSYMM Z+1/4,-Y+1/4,X+1/4" +
                "\nSYMM -Z+1/4,-Y+1/4,-X+1/4\nSYMM -X+1/4,Z+1/4,Y+1/4" +
                "\nSYMM -X+1/4,-Z+1/4,-Y+1/4")
    if int(number) == 211:
        g = str("I432")
        ins_sym = str("LATT -2\nSYMM X,-Z,Y\nSYMM X,Z,-Y\nSYMM Z,Y,-X\nSYMM -
Z,Y,X" +
                "\nSYMM -Y,X,Z\nSYMM Y,-X,Z\nSYMM Z,X,Y\nSYMM Y,Z,X" +
                "\nSYMM -Y,-Z,X\nSYMM Z,-X,-Y\nSYMM -Y,Z,-X\nSYMM -Z,-X,Y" +
                "\nSYMM -Z,X,-Y\nSYMM Y,-Z,-X\nSYMM X,-Y,-Z\nSYMM -X,Y,-Z" +
                "\nSYMM -X,-Y,Z\nSYMM Y,X,-Z\nSYMM -Y,-X,-Z\nSYMM Z,-Y,X" +
                "\nSYMM -Z,-Y,-X\nSYMM -X,Z,Y\nSYMM -X,-Z,-Y")
    if int(number) == 212:
        g = str("P4(3)32")
        ins_sym = str("LATT -1\nSYMM X+3/4,-Z+3/4,Y+1/4" +
                "\nSYMM X+1/4,Z+3/4,-Y+3/4\nSYMM Z+1/4,Y+3/4,-X+3/4" +
                "\nSYMM -Z+3/4,Y+1/4,X+3/4\nSYMM -Y+3/4,X+1/4,Z+3/4" +
                "\nSYMM Y+3/4,-X+3/4,Z+1/4\nSYMM Z,X,Y\nSYMM Y,Z,X" +
                "\nSYMM -Y+1/2,-Z,X+1/2\nSYMM Z+1/2,-X+1/2,-Y" +
                "\nSYMM -Y,Z+1/2,-X+1/2\nSYMM -Z+1/2,-X,Y+1/2" +
                "\nSYMM -Z,X+1/2,-Y+1/2\nSYMM Y+1/2,-Z+1/2,-X" +
                "\nSYMM X+1/2,-Y+1/2,-Z\nSYMM -X,Y+1/2,-Z+1/2" +
                "\nSYMM -X+1/2,-Y,Z+1/2\nSYMM Y+1/4,X+3/4,-Z+3/4" +
                "\nSYMM -Y+1/4,-X+1/4,-Z+1/4\nSYMM Z+3/4,-Y+3/4,X+1/4" +
```

```
            "\nSYMM -Z+1/4,-Y+1/4,-X+1/4\nSYMM -X+3/4,Z+1/4,Y+3/4" +
            "\nSYMM -X+1/4,-Z+1/4,-Y+1/4")
if int(number) == 213:
   g = str("P4(1)32")
   ins_sym = str("LATT -1\nSYMM X+1/4,-Z+1/4,Y+3/4" +
            "\nSYMM X+3/4,Z+1/4,-Y+1/4\nSYMM Z+3/4,Y+1/4,-X+1/4" +
            "\nSYMM -Z+1/4,Y+3/4,X+1/4\nSYMM -Y+1/4,X+3/4,Z+1/4" +
            "\nSYMM Y+1/4,-X+1/4,Z+3/4\nSYMM Z,X,Y\nSYMM Y,Z,X" +
            "\nSYMM -Y+1/2,-Z,X+1/2\nSYMM Z+1/2,-X+1/2,-Y" +
            "\nSYMM -Y,Z+1/2,-X+1/2\nSYMM -Z+1/2,-X,Y+1/2" +
            "\nSYMM -Z,X+1/2,-Y+1/2\nSYMM Y+1/2,-Z+1/2,-X" +
            "\nSYMM X+1/2,-Y+1/2,-Z\nSYMM -X,Y+1/2,-Z+1/2" +
            "\nSYMM -X+1/2,-Y,Z+1/2\nSYMM Y+3/4,X+1/4,-Z+1/4" +
            "\nSYMM -Y+3/4,-X+3/4,-Z+3/4\nSYMM Z+1/4,-Y+1/4,X+3/4" +
            "\nSYMM -Z+3/4,-Y+3/4,-X+3/4\nSYMM -X+1/4,Z+3/4,Y+1/4" +
            "\nSYMM -X+3/4,-Z+3/4,-Y+3/4")
if int(number) == 214:
   g = str("I4(1)32")
   ins_sym = str("LATT -2\nSYMM X+1/4,-Z+1/4,Y+3/4" +
            "\nSYMM X+1/4,Z+3/4,-Y+3/4\nSYMM Z+1/4,Y+3/4,-X+3/4" +
            "\nSYMM -Z+1/4,Y+3/4,X+1/4\nSYMM -Y+1/4,X+3/4,Z+1/4" +
            "\nSYMM Y+1/4,-X+1/4,Z+3/4\nSYMM Z,X,Y\nSYMM Y,Z,X" +
            "\nSYMM -Y,-Z+1/2,X\nSYMM Z,-X,-Y+1/2\nSYMM -Y+1/2,Z,-X" +
            "\nSYMM -Z,-X+1/2,Y\nSYMM -Z+1/2,X,-Y\nSYMM Y,-Z,-X+1/2" +
            "\nSYMM X,-Y,-Z+1/2\nSYMM -X+1/2,Y,-Z\nSYMM -X,-Y+1/2,Z" +
            "\nSYMM Y+1/4,X+3/4,-Z+3/4\nSYMM -Y+1/4,-X+1/4,-Z+1/4" +
            "\nSYMM Z+1/4,-Y+1/4,X+3/4\nSYMM -Z+1/4,-Y+1/4,-X+1/4" +
            "\nSYMM -X+1/4,Z+3/4,Y+1/4\nSYMM -X+1/4,-Z+1/4,-Y+1/4")
if int(number) == 215:
   g = str("P-43m")
   ins_sym = str("LATT -1\nSYMM -X,Z,-Y\nSYMM -X,-Z,Y\nSYMM -Z,-Y,X" +
            "\nSYMM Z,-Y,-X\nSYMM Y,-X,-Z\nSYMM -Y,X,-Z\nSYMM Z,X,Y"+
            "\nSYMM Y,Z,X\nSYMM -Y,-Z,X\nSYMM Z,-X,-Y\nSYMM -Y,Z,-X"+
            "\nSYMM -Z,-X,Y\nSYMM -Z,X,-Y\nSYMM Y,-Z,-X\nSYMM X,-Y,-Z"+
            "\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z\nSYMM -Y,-X,Z\nSYMM Y,X,Z"+
            "\nSYMM -Z,Y,-X\nSYMM Z,Y,X\nSYMM X,-Z,-Y\nSYMM X,Z,Y")
if int(number) == 216:
   g = str("F-43m")
   ins_sym = str("LATT -4\nSYMM -X,Z,-Y\nSYMM -X,-Z,Y\nSYMM -Z,-Y,X" +
            "\nSYMM Z,-Y,-X\nSYMM Y,-X,-Z\nSYMM -Y,X,-Z\nSYMM Z,X,Y" +
            "\nSYMM Y,Z,X\nSYMM -Y,-Z,X\nSYMM Z,-X,-Y\nSYMM -Y,Z,-X" +
            "\nSYMM -Z,-X,Y\nSYMM -Z,X,-Y\nSYMM Y,-Z,-X\nSYMM X,-Y,-Z" +
            "\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z\nSYMM -Y,-X,Z\nSYMM Y,X,Z" +
            "\nSYMM -Z,Y,-X\nSYMM Z,Y,X\nSYMM X,-Z,-Y\nSYMM X,Z,Y")
if int(number) == 217:
   g = str("I-43m")
```

```
        ins_sym = str("LATT -2\nSYMM -X,Z,-Y\nSYMM -X,-Z,Y\nSYMM -Z,-Y,X" +
                      "\nSYMM Z,-Y,-X\nSYMM Y,-X,-Z\nSYMM -Y,X,-Z\nSYMM Z,X,Y" +
                      "\nSYMM Y,Z,X\nSYMM -Y,-Z,X\nSYMM Z,-X,-Y\nSYMM -Y,Z,-X" +
                      "\nSYMM -Z,-X,Y\nSYMM -Z,X,-Y\nSYMM Y,-Z,-X\nSYMM X,-Y,-Z" +
                      "\nSYMM -X,Y,-Z\nSYMM -X,-Y,Z\nSYMM -Y,-X,Z\nSYMM Y,X,Z" +
                      "\nSYMM -Z,Y,-X\nSYMM Z,Y,X\nSYMM X,-Z,-Y\nSYMM X,Z,Y")


if __name__ == "__main__":
    main()
```

4.7.2 Automated MicroED Data Collection Programs

4.7.2.1 Automated Data Collection Procedure

On a ThermoFisher Scientific Talos F200C equipped with a Ceta-D detector, TEMScripting, and Advanced Scripting, with Python and supporting libraries (can be found in the script's "import" text), a properly mapped MUI (utilizing PyAutoGUI) is controlled by executing the program (Lowest_Dose.py or IncDif_v1.py) in Python. A support computer simultaneously runs CrystalEyesA_v3.py and monitors data files generated by the microscope computer.

4.7.2.2 Lowest_Dose.py

```
"""
Allows for streamlined Low Dose screening
"""
import os
import sys
import time
import pyautogui
from threading import Thread
from tkinter import *
import tkinter as tk
from tkinter import ttk
from win32com import client
global MyTEM
global sp

MyTEM = client.Dispatch("TEMScripting.Instrument")

sp = []

def push_button():
    #check status of beam, blank if not already done
    beam_stat = MyTEM.Illumination
    beam_stat.BeamBlanked = True
    mode = MyTEM.Projection
    #ensure we're in imaging mode before going forward
    if str(mode.SubModeString) == "SA":
```

```
        original_x, original_y = pyautogui.position()
        pyautogui.click(x=115, y=38) #ensure that camera flap is selected
        doublecheck_1 = beam_stat.BeamBlanked
        pyautogui.click(x=47, y=209)
        doublecheck_2 = beam_stat.Beamblanked
        pyautogui.click(x=577, y=59)
        time.sleep(1)
        if doublecheck_1 != doublecheck_2:
            SA_value2 = SA_value.get()
            if str(SA_value2) == "100":
                pyautogui.click(x=110, y=871) #dropdown SA menu
                pyautogui.click(x=116, y=901) #select 100 SA aperture
            if str(SA_value2) == "40":
                pyautogui.click(x=110, y=871) #dropdown SA menu
                pyautogui.click(x=116, y=916) #select 40 SA aperture
            pyautogui.click(x=193, y=395) #exposure mode
            time.sleep(9) #wait 10 seconds for normalization
            beam_stat.BeamBlanked = False #unblank so you can take a peek
    else:
        print("Make sure you're in SA first!")
        sp.clear()
def push_button_2():
    beam_stat = MyTEM.Illumination
    pyautogui.click(x=115, y=38) #ensure that camera flap is selected
    doublecheck_1 = beam_stat.BeamBlanked
    pyautogui.click(x=47, y=209)
    doublecheck_2 = beam_stat.Beamblanked
    time.sleep(1)
    if doublecheck_1 != doublecheck_2:
        beam_stat.BeamBlanked = False #blank the beam again
        original_x, original_y = pyautogui.position()
        screen_access = MyTEM.Camera
        screen_access.MainScreen = 2 # raise up screen
        time.sleep(1) #wait 2 seconds
        pyautogui.click(x=195, y=237) # acquire image
        time.sleep(9) #wait for image
        beam_stat.BeamBlanked = True #blank the beam again
def push_button_3():
    beam_stat = MyTEM.Illumination
    pyautogui.click(x=115, y=38) #ensure that camera flap is selected
    doublecheck_1 = beam_stat.BeamBlanked
    pyautogui.click(x=47, y=209)
    time.sleep(1)
    doublecheck_2 = beam_stat.Beamblanked
    if doublecheck_1 != doublecheck_2:
        original_x, original_y = pyautogui.position()
```

```
        cam_access = MyTEM.Camera
        cam_access.MainScreen = 3 # move flu screen down
        pyautogui.click(x=47, y=866) #move SA aperture out
        pyautogui.click(x=623, y=56) #beamstop out
        time.sleep(1)
        pyautogui.click(x=38, y=396) #back to search mode
        time.sleep(9) #wait 10 seconds
        beam_stat.BeamBlanked = False #unblank

root = Tk()
root.title("LowestDose")
root.columnconfigure(0,weight=1)
root.rowconfigure(0,weight=1)
f1 = ttk.Frame(root, padding="12 12 12 12")
f1.grid(column=0, row=0, sticky=(N, W, E, S))
ttk.Button(f1, text="Diffraction\n Mode", command=lambda:
Thread(target=push_button).start()).grid(
        column=0, row=0,sticky=E,pady=5)
ttk.Button(f1, text="Shoot\n  It", command=lambda: Thread(target=push_button_2).start()).grid(
        column=1, row=0, sticky=E,pady=5)
ttk.Button(f1, text="Imaging\n Mode", command=lambda:
Thread(target=push_button_3).start()).grid(
        column=2, row=0, sticky=E,pady=5)
SA_value = StringVar()
SA_value.set("100")
ttk.Label(f1, text="SA aperture:",
justify=LEFT).grid(column=0,row=1,columnspan=2,sticky=W)
tk.OptionMenu(f1, SA_value, "100", "40").grid(column=1,row=1,sticky=W)
root.mainloop()
```

4.7.2.3 IncDif_v1.py

```
"""
Completed Incident Diffraction Program
"""
import os
import sys
import time
import pyautogui
import ctypes
from tkinter import *
import tkinter as tk
from tkinter import ttk
from tkinter import filedialog
from win32com import client
global MyTEM
global coords

MyTEM = client.Dispatch("TEMScripting.Instrument")

coords = []


def grid_map():
    stage = MyTEM.Stage
    map_coords = ["0.1,0.1","0.1,-157","-157,-157","-157,0.1",
                  "0.1,157","157,157","157,0.1","314,0.1",
                  "0.1,314","0.1,-314","-314,0.1","314,314","-314,-314",
                  "-314,314","314,-314"]
    if str(stage.Status) == "0":
        for line in map_coords:
            c = line.split(",")
            x = str(c[0])
            y = str(c[1])
            pyautogui.tripleClick(209,35) #Open "Auto" tab
            pyautogui.click(163,35) #Open "Auto" tab
            pyautogui.doubleClick(447,86) # !!!!! Left arrow button of the stage side flap
            pyautogui.click(426,88) #!!! Click the settings button
            time.sleep(1)
            if x != "0":
                pyautogui.doubleClick(329,140) #click X
                pyautogui.press('backspace')
                pyautogui.press('backspace')
                pyautogui.press('delete')
                pyautogui.press('delete')
```

```
    pyautogui.press('delete')
    pyautogui.typewrite(x)
if y != "0":
    pyautogui.doubleClick(329,164) #click X
    pyautogui.press('backspace')
    pyautogui.press('backspace')
    pyautogui.press('delete')
    pyautogui.press('delete')
    pyautogui.press('delete')
    pyautogui.typewrite(y)
pyautogui.click(289,302) #click go to
time.sleep(2)
while str(stage.Status) == "3":
    print("waiting for stage to stabilize")
    time.sleep(2)
    stage.Status
    if str(stage.Status) == "0":
        print("stop")
        break
beam_stat = MyTEM.Illumination
beam_stat.BeamBlanked = True
mode = MyTEM.Projection
#ensure we're in imaging mode before going forward
if str(mode.SubModeString) == "LM":
    pyautogui.click(x=115, y=38) #ensure that camera flap is selected
    doublecheck_1 = beam_stat.BeamBlanked
    pyautogui.click(x=47, y=209)
    doublecheck_2 = beam_stat.BeamBlanked
    if doublecheck_1 != doublecheck_2:
        screen_access = MyTEM.Camera
        screen_access.MainScreen = 2 # raise up screen
        time.sleep(1) #wait 2 seconds
        pyautogui.click(x=195, y=237) # acquire image
        print("pic")
        time.sleep(9) #wait for image
        beam_stat.BeamBlanked = True #blank the beam again
        cam_access = MyTEM.Camera
        cam_access.MainScreen = 3 # move flu screen down
        time.sleep(3) #wait 10 seconds
        beam_stat.BeamBlanked = False #unblank
        ctypes.windll.user32.SetCursorPos(2500,700)
        pyautogui.click()
        pyautogui.keyDown('ctrl')
        pyautogui.press('e')
        pyautogui.keyUp('ctrl')
        pyautogui.typewrite(x + "x_" + y + "y_" + "LM_380")
```

```
                pyautogui.press('enter')
                time.sleep(2)

destPath = []




#import list of grid coordinates from a csv file
def auto_screen():
    stage = MyTEM.Stage
    coord_file = str(coords).strip("])([,'")
    if os.path.isfile(coord_file) == True:
        pyautogui.tripleClick(209,35) # !!!! Left arrow on the upper GUI
        f = open(coord_file,"r")
        line = f.readline()
        for line in f:
            c = line.split(",")
            if len(c) == 4:
                x = str(c[0]).strip("'")
                y = str(c[1]).strip("'")
                z = str(c[2]).strip("'")
                a = str(c[3]).strip("'")
                pyautogui.click(163,35) #Open "Auto" tab
                pyautogui.doubleClick(447,86) # !!!!! Left arrow button of the stage side flap
                pyautogui.click(426,88) #!!! Click the settings button
                time.sleep(1)
                if x != "0":
                    pyautogui.doubleClick(329,140) #click X
                    pyautogui.press('backspace')
                    pyautogui.press('backspace')
                    pyautogui.press('delete')
                    pyautogui.press('delete')
                    pyautogui.press('delete')
                    pyautogui.typewrite(x)
                if y != "0":
                    pyautogui.doubleClick(329,164) #click X
                    pyautogui.press('backspace')
                    pyautogui.press('backspace')
                    pyautogui.press('delete')
                    pyautogui.press('delete')
                    pyautogui.press('delete')
                    pyautogui.typewrite(y)
                if z != "0":
```

```
    pyautogui.doubleClick(329,186) #click X
    pyautogui.press('backspace')
    pyautogui.press('backspace')
    pyautogui.press('delete')
    pyautogui.press('delete')
    pyautogui.press('delete')
    pyautogui.typewrite(z)
if a != "0":
    pyautogui.doubleClick(329,209) #click X
    pyautogui.press('backspace')
    pyautogui.press('backspace')
    pyautogui.press('delete')
    pyautogui.press('delete')
    pyautogui.press('delete')
    pyautogui.typewrite(a)
pyautogui.click(289,302) #click go to
time.sleep(1)
if str(stage.Status) == "0":
    pyautogui.click(223,82) #Stage side flap button!!
    pyautogui.doubleClick(447,86) # !!!!! Left arrow button of the stage side flap
    pyautogui.click(426,88) #!!! Click the settings button
    time.sleep(1)
    if x != "0":
        pyautogui.doubleClick(329,140) #click X
        pyautogui.press('backspace')
        pyautogui.press('backspace')
        pyautogui.press('delete')
        pyautogui.press('delete')
        pyautogui.press('delete')
        pyautogui.typewrite(x)
    if y != "0":
        pyautogui.doubleClick(329,164) #click X
        pyautogui.press('backspace')
        pyautogui.press('backspace')
        pyautogui.press('delete')
        pyautogui.press('delete')
        pyautogui.press('delete')
        pyautogui.typewrite(y)
    if z != "0":
        pyautogui.doubleClick(329,186) #click X
        pyautogui.press('backspace')
        pyautogui.press('backspace')
        pyautogui.press('delete')
        pyautogui.press('delete')
        pyautogui.press('delete')
        pyautogui.typewrite(z)
```

```
if a != "0":
    pyautogui.doubleClick(329,209) #click X
    pyautogui.press('backspace')
    pyautogui.press('backspace')
    pyautogui.press('delete')
    pyautogui.press('delete')
    pyautogui.press('delete')
    pyautogui.typewrite(a)
pyautogui.click(289,302) #click go to
time.sleep(1)
if str(stage.Status) == "0":
    print("Cannot move stage :(")
    break
elif str(stage.Status) == "3":
    while str(stage.Status) == "3":
        time.sleep(2)
        stage.Status
        if str(stage.Status) == "0":
            break
#check status of beam, blank if not already done
beam_stat = MyTEM.Illumination
beam_stat.BeamBlanked = True
mode = MyTEM.Projection
#ensure we're in imaging mode before going forward
if str(mode.SubModeString) == "SA":
    pyautogui.click(x=115, y=38) #ensure that camera flap is selected
    doublecheck_1 = beam_stat.BeamBlanked
    pyautogui.click(x=47, y=209)
    doublecheck_2 = beam_stat.BeamBlanked
    if doublecheck_1 != doublecheck_2:
        screen_access = MyTEM.Camera
        screen_access.MainScreen = 2 # raise up screen
        time.sleep(1) #wait 2 seconds
        pyautogui.click(x=195, y=237) # acquire image
        print("pic")
        time.sleep(9) #wait for image
        beam_stat.BeamBlanked = True #blank the beam again
        cam_access = MyTEM.Camera
        cam_access.MainScreen = 3 # move flu screen down
        time.sleep(3) #wait 3 seconds
        beam_stat.BeamBlanked = False #unblank
        ctypes.windll.user32.SetCursorPos(2500,700)
        pyautogui.click()
        pyautogui.keyDown('ctrl')
        pyautogui.press('e')
        pyautogui.keyUp('ctrl')
```

```
                pyautogui.typewrite(x + "x_" + y + "y_" + "hm")
                pyautogui.press('enter')
                time.sleep(6)
                c = coord_file.split("/")
                sep = "\\"
                path = sep.join(c[:-1])
                os.chdir(path)
                os.getcwd()
                adj_file = str(x + "x_" + y + "y.csv")
                if os.path.isfile(adj_file) == True:
                    f = open(adj_file,"r")
                    line = f.readline()
                    for line in f:
                        c = line.split(",")
                        if len(c) == 4:
                            x = str(c[0]).strip("'")
                            y = str(c[1]).strip("'")
                            z = str(c[2]).strip("'")
                            a = str(c[3]).strip("'")
                            pyautogui.click(163,35) #Open "Auto" tab
                            pyautogui.doubleClick(447,86) # !!!!! Left arrow button of the stage side
flap
                            pyautogui.click(426,88) #!!! Click the settings button
                            time.sleep(1)
                            if x != "0":
                                pyautogui.doubleClick(329,140) #click X
                                pyautogui.press('backspace')
                                pyautogui.press('backspace')
                                pyautogui.press('delete')
                                pyautogui.press('delete')
                                pyautogui.press('delete')
                                pyautogui.typewrite(x)
                            if y != "0":
                                pyautogui.doubleClick(329,164) #click X
                                pyautogui.press('backspace')
                                pyautogui.press('backspace')
                                pyautogui.press('delete')
                                pyautogui.press('delete')
                                pyautogui.press('delete')
                                pyautogui.typewrite(y)
                            if z != "0":
                                pyautogui.doubleClick(329,186) #click X
                                pyautogui.press('backspace')
                                pyautogui.press('backspace')
                                pyautogui.press('delete')
                                pyautogui.press('delete')
```

```
            pyautogui.press('delete')
            pyautogui.typewrite(z)
        if a != "0":
            pyautogui.doubleClick(329,209) #click X
            pyautogui.press('backspace')
            pyautogui.press('backspace')
            pyautogui.press('delete')
            pyautogui.press('delete')
            pyautogui.press('delete')
            pyautogui.typewrite(a)
        pyautogui.click(289,302) #click go to
        time.sleep(1)
        if str(stage.Status) == "0":
            pyautogui.click(223,82) #Stage side flap button!!
            pyautogui.doubleClick(447,86) # !!!!! Left arrow button of the stage
side flap
            pyautogui.click(426,88) #!!! Click the settings button
            time.sleep(1)
            if x != "0":
                pyautogui.doubleClick(329,140) #click X
                pyautogui.press('backspace')
                pyautogui.press('backspace')
                pyautogui.press('delete')
                pyautogui.press('delete')
                pyautogui.press('delete')
                pyautogui.typewrite(x)
            if y != "0":
                pyautogui.doubleClick(329,164) #click X
                pyautogui.press('backspace')
                pyautogui.press('backspace')
                pyautogui.press('delete')
                pyautogui.press('delete')
                pyautogui.press('delete')
                pyautogui.typewrite(y)
            if z != "0":
                pyautogui.doubleClick(329,186) #click X
                pyautogui.press('backspace')
                pyautogui.press('backspace')
                pyautogui.press('delete')
                pyautogui.press('delete')
                pyautogui.press('delete')
                pyautogui.typewrite(z)
            if a != "0":
                pyautogui.doubleClick(329,209) #click X
                pyautogui.press('backspace')
                pyautogui.press('backspace')
```

```
            pyautogui.press('delete')
            pyautogui.press('delete')
            pyautogui.press('delete')
            pyautogui.typewrite(a)
        pyautogui.click(289,302) #click go to
        time.sleep(1)
    if str(stage.Status) == "0":
        print("Cannot move stage :(")
        break
    elif str(stage.Status) == "3":
        while str(stage.Status) == "3":
            time.sleep(2)
            stage.Status
            if str(stage.Status) == "0":
                break
        #check status of beam, blank if not already done
        beam_stat = MyTEM.Illumination
        beam_stat.BeamBlanked = True
        mode = MyTEM.Projection
        if str(mode.SubModeString) == "SA":
            original_x, original_y = pyautogui.position()
            pyautogui.click(x=115, y=38) #ensure that camera flap is selected
            doublecheck_1 = beam_stat.BeamBlanked
            pyautogui.click(x=47, y=209)
            doublecheck_2 = beam_stat.BeamBlanked
            pyautogui.click(x=577, y=59) #beamstop in y=59 on scope
            time.sleep(1)
            if doublecheck_1 != doublecheck_2:
                SA_value2 = SA_value.get()
                if str(SA_value2) == "100":
                    #print("I would've put in the SA aperture...")
                    pyautogui.click(x=110, y=871) #dropdown SA menu
                    pyautogui.click(x=116, y=901) #select 100 SA aperture
                if str(SA_value2) == "40":
                    #print("I would've put in the SA aperture...")
                    pyautogui.click(x=110, y=871) #dropdown SA menu
                    pyautogui.click(x=116, y=916) #select 40 SA
                pyautogui.click(x=193, y=395) #exposure mode
                time.sleep(9) #wait 10 seconds for normalization
                beam_stat.BeamBlanked = False #unblank so you can take a peek
                time.sleep(3)
                screen_access = MyTEM.Camera
                screen_access.MainScreen = 2 # raise up screen
                time.sleep(1) #wait 1 seconds
                pyautogui.click(x=195, y=237) # acquire image
```

```
                                time.sleep(9) #wait for image (need a way to check for continuous
diff)
                                beam_stat.BeamBlanked = True #blank the beam again
                                ctypes.windll.user32.SetCursorPos(2500,700)
                                pyautogui.click()
                                pyautogui.keyDown('ctrl')
                                pyautogui.press('e')
                                pyautogui.keyUp('ctrl')
                                pyautogui.typewrite(x + "x_" + y + "y_" +
                                                    "incdif")
                                pyautogui.press('enter')
                                time.sleep(1)
                                cam_access = MyTEM.Camera
                                cam_access.MainScreen = 3 # move flu screen down
                                pyautogui.click(x=47, y=866) #move SA aperture out
                                pyautogui.click(x=623, y=56) #beamstop out
                                time.sleep(1)
                                pyautogui.click(x=38, y=396) #back to search mode
                                time.sleep(9) #wait 10 seconds
                                beam_stat.BeamBlanked = False #unblank
                                #save the image somewhere yay
                            else:
                                print("Make sure you're in SA imaging mode first!")
                            time.sleep(2)
                            print("collected at " + x + " " + y + " " + z + " " + a)
                    if os.path.isfile(adj_file) == False:
                        print(c)
                        print(coord_file)
                        print(str(os.getcwd()))
            elif str(stage.Status) == "3":
                while str(stage.Status) == "3":
                    time.sleep(2)
                    stage.Status
                    if str(stage.Status) == "0":
                        break
                beam_stat = MyTEM.Illumination
                beam_stat.BeamBlanked = True
                mode = MyTEM.Projection
                #ensure we're in imaging mode before going forward
                if str(mode.SubModeString) == "SA":
                    pyautogui.click(x=115, y=38) #ensure that camera flap is selected
                    doublecheck_1 = beam_stat.BeamBlanked
                    pyautogui.click(x=47, y=209)
                    doublecheck_2 = beam_stat.BeamBlanked
                    if doublecheck_1 != doublecheck_2:
                        screen_access = MyTEM.Camera
```

```
screen_access.MainScreen = 2 # raise up screen
time.sleep(1) #wait 2 seconds
pyautogui.click(x=195, y=237) # acquire image
print("pic")
time.sleep(9) #wait for image
beam_stat.BeamBlanked = True #blank the beam again
cam_access = MyTEM.Camera
cam_access.MainScreen = 3 # move flu screen down
time.sleep(3) #wait 10 seconds
beam_stat.BeamBlanked = False #unblank
ctypes.windll.user32.SetCursorPos(2500,700)
pyautogui.click()
pyautogui.keyDown('ctrl')
pyautogui.press('e')
pyautogui.keyUp('ctrl')
pyautogui.typewrite(x + "x_" + y + "y_" + "hm")
pyautogui.press('enter')
time.sleep(6)
c = coord_file.split("/")
sep = "\\"
path = sep.join(c[:-1])
os.chdir(path)
os.getcwd()
adj_file = str(x + "x_" + y + "y.csv")
if os.path.isfile(adj_file) == True:
    f = open(adj_file,"r")
    line = f.readline()
    for line in f:
        c = line.split(",")
        if len(c) == 4:
            x = str(c[0]).strip("'")
            y = str(c[1]).strip("'")
            z = str(c[2]).strip("'")
            a = str(c[3]).strip("'")
            pyautogui.click(163,35) #Open "Auto" tab
            pyautogui.doubleClick(447,86) # !!!!! Left arrow button of the stage side
flap
            pyautogui.click(426,88) #!!! Click the settings button
            time.sleep(1)
            if x != "0":
                pyautogui.doubleClick(329,140) #click X
                pyautogui.press('backspace')
                pyautogui.press('backspace')
                pyautogui.press('delete')
                pyautogui.press('delete')
                pyautogui.press('delete')
```

```
            pyautogui.typewrite(x)
        if y != "0":
            pyautogui.doubleClick(329,164) #click X
            pyautogui.press('backspace')
            pyautogui.press('backspace')
            pyautogui.press('delete')
            pyautogui.press('delete')
            pyautogui.press('delete')
            pyautogui.typewrite(y)
        if z != "0":
            pyautogui.doubleClick(329,186) #click X
            pyautogui.press('backspace')
            pyautogui.press('backspace')
            pyautogui.press('delete')
            pyautogui.press('delete')
            pyautogui.press('delete')
            pyautogui.typewrite(z)
        if a != "0":
            pyautogui.doubleClick(329,209) #click X
            pyautogui.press('backspace')
            pyautogui.press('backspace')
            pyautogui.press('delete')
            pyautogui.press('delete')
            pyautogui.press('delete')
            pyautogui.typewrite(a)
        pyautogui.click(289,302) #click go to
        time.sleep(1)
        if str(stage.Status) == "0":
            pyautogui.click(223,82) #Stage side flap button!!
            pyautogui.doubleClick(447,86) # !!!!! Left arrow button of the stage
side flap
            pyautogui.click(426,88) #!!! Click the settings button
            time.sleep(1)
            if x != "0":
                pyautogui.doubleClick(329,140) #click X
                pyautogui.press('backspace')
                pyautogui.press('backspace')
                pyautogui.press('delete')
                pyautogui.press('delete')
                pyautogui.press('delete')
                pyautogui.typewrite(x)
            if y != "0":
                pyautogui.doubleClick(329,164) #click X
                pyautogui.press('backspace')
                pyautogui.press('backspace')
                pyautogui.press('delete')
```

```
            pyautogui.press('delete')
            pyautogui.press('delete')
            pyautogui.typewrite(y)
        if z != "0":
            pyautogui.doubleClick(329,186) #click X
            pyautogui.press('backspace')
            pyautogui.press('backspace')
            pyautogui.press('delete')
            pyautogui.press('delete')
            pyautogui.press('delete')
            pyautogui.typewrite(z)
        if a != "0":
            pyautogui.doubleClick(329,209) #click X
            pyautogui.press('backspace')
            pyautogui.press('backspace')
            pyautogui.press('delete')
            pyautogui.press('delete')
            pyautogui.press('delete')
            pyautogui.typewrite(a)
        pyautogui.click(289,302) #click go to
        time.sleep(1)
    if str(stage.Status) == "0":
            print("Cannot move stage :(")
            break
    elif str(stage.Status) == "3":
        while str(stage.Status) == "3":
            time.sleep(2)
            stage.Status
            if str(stage.Status) == "0":
                break
        #check status of beam, blank if not already done
        beam_stat = MyTEM.Illumination
        beam_stat.BeamBlanked = True
        mode = MyTEM.Projection
        if str(mode.SubModeString) == "SA":
            original_x, original_y = pyautogui.position()
            pyautogui.click(x=115, y=38) #ensure that camera flap is selected
            doublecheck_1 = beam_stat.BeamBlanked
            pyautogui.click(x=47, y=209)
            doublecheck_2 = beam_stat.BeamBlanked
            pyautogui.click(x=577, y=59) #beamstop in y=59 on scope
            time.sleep(1)
            if doublecheck_1 != doublecheck_2:
                SA_value2 = SA_value.get()
                if str(SA_value2) == "100":
                    #print("I would've put in the SA aperture...")
```

```
                        pyautogui.click(x=110, y=871) #dropdown SA menu scope is
871
                        pyautogui.click(x=116, y=901) #select 100 SA aperture scope
is 901
                    if str(SA_value2) == "40":
                        #print("I would've put in the SA aperture...")
                        pyautogui.click(x=110, y=871) #dropdown SA menu
                        pyautogui.click(x=116, y=916) #select 40 SA aperture scope
                    pyautogui.click(x=193, y=395) #exposure mode
                    time.sleep(9) #wait 10 seconds for normalization
                    beam_stat.BeamBlanked = False #unblank so you can take a peek
                    time.sleep(3)
                    screen_access = MyTEM.Camera
                    screen_access.MainScreen = 2 # raise up screen
                    time.sleep(1) #wait 1 seconds
                    pyautogui.click(x=195, y=237) # acquire image
                    time.sleep(9) #wait for image
                    beam_stat.BeamBlanked = True #blank the beam again
                    ctypes.windll.user32.SetCursorPos(2500,700)
                    pyautogui.click()
                    pyautogui.keyDown('ctrl')
                    pyautogui.press('e')
                    pyautogui.keyUp('ctrl')
                    pyautogui.typewrite(x + "x_" + y + "y_" +
                                "incdif")
                    pyautogui.press('enter')
                    time.sleep(1)
                    cam_access = MyTEM.Camera
                    cam_access.MainScreen = 3 # move flu screen down
                    pyautogui.click(x=47, y=866) #move SA aperture out
                    pyautogui.click(x=623, y=56) #beamstop out
                    time.sleep(1)
                    pyautogui.click(x=38, y=396) #back to search mode
                    time.sleep(9) #wait 10 seconds
                    beam_stat.BeamBlanked = False #unblank
                    #save the image somewhere yay
                else:
                    print("Make sure you're in SA imaging mode first!")
                time.sleep(2)
                print("collected at " + x + " " + y + " " + z + " " + a)
            if os.path.isfile(adj_file) == False:
                print(c)
                print(coord_file)
                print(str(os.getcwd()))

    else:
```

```
        print("Need to import coordinates!")


def fileadd():
    coords.clear()
    files = filedialog.askopenfilename(title="Select data files",filetypes =
                                ((("ser files","*.csv"),
                                 ("all files","*.*")),multiple=False)
    var = root.tk.splitlist(files)
    coords.append(var)
#def stop():
#    print("main")

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Start                                    of
GUI~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
root = Tk()
#photo = PhotoImage(file = "icon.gif")
root.title("CrystalEyesEM")
#root.iconphoto(False,photo)
root.columnconfigure(0,weight=1)
root.rowconfigure(0,weight=1)
n = ttk.Notebook(root)
n.grid(column=0,row=0)
ttk.Label(root, text="Status:",justify=LEFT).grid(column=0,row=1,sticky=W,pady=5)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~First
page~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
f1 = ttk.Frame(n, padding="12 12 12 12")
f1.grid(column=2, row=1, sticky=(N, W, E, S))
f1.columnconfigure(0, weight=1)
f1.rowconfigure(0, weight=1)
sf1 = ttk.Frame(f1, borderwidth=5, relief=GROOVE)
sf1.grid(column=1,row=0,sticky=(N, W, E, S),pady=3)
sf1.columnconfigure(0, weight=1)
sf1.columnconfigure(0, weight=1)
s1f1 = ttk.Frame(f1, borderwidth=5, relief=GROOVE)
s1f1.grid(column=2,row=0,sticky=(N, W, E, S),pady=3)
s1f1.columnconfigure(0, weight=1)
s1f1.columnconfigure(0, weight=1)
ttk.Label(sf1, text="TEM Settings").grid(column=0,row=1,pady=3)
ttk.Label(sf1, text="Magnification:").grid(column=0,row=2)
mag = StringVar()
mag.set("LM 380x")
sc = tk.OptionMenu(sf1, mag, "LM 380x").grid(column=1,row=2)
mon = StringVar()
mon.set("4x4")
```

```
mnt = tk.OptionMenu(sf1, mon, "4x4", "4x4").grid(column=1,row=3)
ttk.Label(sf1, text="LM Map Size: ").grid(column=0,row=3)

ttk.Label(sf1, text="Ensure TIA is exporting images\n" +
        "to correct location and camera is\n" +
        "not in continuous collection mode").grid(column=0,row=4,
                                    columnspan=2)
ttk.Button(sf1, text="Start",command=grid_map).grid(
         column=0, row=5,columnspan=1,sticky=E)
#ttk.Button(sf1, text="Stop",command=stop).grid(
#          column=1, row=5,columnspan=1,sticky=W)

#ttk.Label(s1f1, text= "         to be replaced by images       \n\n\n").grid(
#    column=1,row=1)




#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Second
page~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
f2 = ttk.Frame(n, padding="12 12 12 12") # second page
f2.grid(column=0, row=0, sticky=(N, W, E, S))
f2.columnconfigure(0, weight=1)
f2.rowconfigure(0, weight=1)


sf2 = ttk.Frame(f2, borderwidth=5, relief=GROOVE)
sf2.grid(column=1,row=0,sticky=(N, W, E, S),pady=3)
sf2.columnconfigure(0, weight=1)
sf2.columnconfigure(0, weight=1)
s1f2 = ttk.Frame(f2, borderwidth=5, relief=GROOVE)
s1f2.grid(column=2,row=0,sticky=(N, W, E, S),pady=3)
s1f2.columnconfigure(0, weight=1)
s1f2.columnconfigure(0, weight=1)
ttk.Label(sf2, text="TEM Settings").grid(column=0,row=1,pady=3)
ttk.Label(sf2, text="Magnification:").grid(column=0,row=2)
hmag = StringVar()
hmag.set("SA 2600x")
hmagz = tk.OptionMenu(sf2, hmag, "SA 2600x").grid(column=1,row=2)
isi = StringVar()
isi.set("2k x 2k")
isize = tk.OptionMenu(sf2, isi, "2k x 2k").grid(column=1,row=3)
ttk.Label(sf2, text="Final Image Size: ").grid(column=0,row=3)
SA_value = StringVar()
```

```
SA_value.set("100")
ttk.Label(sf2, text="SA aperture:", justify=LEFT).grid(column=0,row=4,columnspan=2,sticky=W)
tk.OptionMenu(sf2, SA_value, "100", "40").grid(column=1,row=4,sticky=W)
ttk.Button(sf2, text='   Import coordinate *.csv file', command=fileadd).grid(column=0, row=5,
        sticky=E,columnspan=2)
ttk.Button(sf2, text="Start",command=auto_screen).grid(
         column=0, row=6,columnspan=1,sticky=E)
#ttk.Button(sf2, text="Stop",command=stop).grid(
#          column=1, row=6,columnspan=1,sticky=W)

#ttk.Label(s1f2, text= "         to be replaced by images       \n\n\n").grid(
#    column=1,row=1)



n.add(f1, text='LM Imaging')
n.add(f2, text='Diffraction Screening')
n.select(f1)
n.enable_traversal()
#root.bind('<Return>', main)
root.mainloop()
```

4.7.2.4 CrystalEyes_v3.py

```
import os
import cv2 as cv
import imutils
global coordinates
global new_coordinates
import numpy as np
from collections import Counter
from decimal import Decimal
import time
from watchdog.observers import Observer
from watchdog.events import FileSystemEventHandler
import random


coordinates = []
new_coordinates = []


def analyze(file):
    path = os.getcwd()
    name = str(file)
    if name.endswith("_LM_380.tif"):
        img = cv.imread(name)
        img2 = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
        ret3,th3 = cv.threshold(img2,0,255,cv.THRESH_BINARY+cv.THRESH_OTSU)
        th3 = cv.erode(th3, None, iterations=2)
        th3 = cv.dilate(th3, None, iterations=4)
        th3 = cv.medianBlur(th3,5)
        edged = cv.Canny(th3, 30, 200)
        contours = cv.findContours(edged, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
        cnts = imutils.grab_contours(contours)
        fname = str(name + "_collectedpts.tif")
        cv.drawContours(img, cnts, -1, (255,0,0), 3)
        for contour in cnts:
            x, y, w, h = cv.boundingRect(contour)
            if int(x) and int(y) != 0:
                if int(w) and int(h) < 200:
                    extLeft = tuple(contour[contour[:, :, 0].argmin()][0])
                    x1 = extLeft[0]
                    y1 = round(int(extLeft[1])/10)*10
                    coo = x1,y1
                    if coo not in coordinates:
                            coordinates.append(coo)
                            cv.circle(img, coo, 8, (0, 255, 0), -1)
```

```
                label = str(coo)
                cv.putText(img, label, (x1, y1), cv.FONT_HERSHEY_PLAIN, 2, (0,0,255), 2)
cv.imwrite(fname, img)
f = open("coord_file.csv","a")
conv_coord = []
for x,y in coordinates:
   center = (1024,1024)
   coo = (x,y)
   if x < 1024 and y < 1024:
      ynm = abs(x - 1024) * 0.0766
      xnm = abs(y - 1024) * 0.0766
   if x < 1024 and y > 1024:
      ynm = abs(x - 1024) * 0.0766
      xnm = abs(y - 1024) * -0.0766
   if x > 1024 and y > 1024:
      ynm = abs(x - 1024) * -0.0766
      xnm = abs(y - 1024) * -0.0766
   if x > 1024 and y < 1024:
      ynm = abs(x - 1024) * -0.0766
      xnm = abs(y - 1024) * 0.0766
   origxy = name.split("_")
   x0 = str(origxy[0]).strip("x")
   y0 = str(origxy[1]).strip("y")
   x_coord = float(x0) + xnm
   y_coord = float(y0) + ynm
   nx = Decimal(x_coord).quantize(Decimal("1.00"))
   ny = Decimal(y_coord).quantize(Decimal("1.00"))
   conv_coord.append(str(nx) + "," + str(ny))
parsed_coord = []
for i in range(len(conv_coord)):
   x1,y1 = conv_coord[i].split(",")
   if i+1 < len(conv_coord):
      x2,y2 = conv_coord[i+1].split(",")
      if abs(float(x1) - float(x2)) and abs(float(y1) - float(y2)) < 12: #opt this value
         pass
      else:
         parsed_coord.append(str(x1) + "," + str(y1))
   else:
      pass
for item in parsed_coord:
   f.write(str(item) + ",0,0\n")
coordinates.clear()
conv_coord.clear()
parsed_coord.clear()
f.close()
```

```
def incdif(file):
    name = str(file)
    if name.endswith("incdif.tif"):
        #this is the first round of image processing
        img = cv.imread(name)
        img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
        crop_img1 = img[25:1033, 45:2048]
        blur1 = cv.medianBlur(crop_img1,7)
        crop_img2 = img[1033:2041,45:2048]
        fl_crop_img2 = cv.flip(crop_img2, -1)
        blur2 = cv.medianBlur(fl_crop_img2,7)
        dst = cv.addWeighted(blur1, 0.5, blur2, 0.5, 0.0)
        ret3,th3 = cv.threshold(dst,0,255,cv.THRESH_BINARY+cv.THRESH_OTSU)
        x,y,w,h = 800,800,450,500
        cv.rectangle(th3, (x, x), (x + w, y + h), (0,0,0), -1)
        th3 = cv.erode(th3, None, iterations=2)
        th3 = cv.dilate(th3, None, iterations=3)
        th3 = cv.medianBlur(th3,3)
        crop_th3 = th3[0:690,0:2003]
        c = crop_th3.flatten().tolist()
        c1 = crop_th3.flatten().tolist()
        x = 255
        d = Counter(c1)
        d2 = Counter(c)
        num = d[x]
        fid = open("inc_dif.csv","a")
        if num == 0:
            fid.write(name + ",0\n")
        if 10000 > num > 0:
            params = cv.SimpleBlobDetector_Params()
            params.blobColor = 255
            params.filterByCircularity = True
            params.minCircularity = 0.7
            params.minDistBetweenBlobs = 10
            detector = cv.SimpleBlobDetector_create(params)
            keypoints = detector.detect(crop_th3)
            kpt_image = cv.drawKeypoints(crop_th3, keypoints, np.array([]),
                    (0,0,255),cv.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
            c = True
            if len(keypoints) > 3:
                pts = [k.pt for k in keypoints]
                center = (1001, 1008) # This should be changed to the center of your image
                distances = []
                for pt in pts:
```

```
            dis = distance2(center,pt)
            distances.append(int(dis))

        for elem in distances:
            if distances.count(elem) > 2:
                fid.write(name + ",0\n")
                c = False
                break
    if c == True:
        fid.write(name + ",1\n")

if 40000 >= num >= 10000:
    img = cv.imread(name)
    img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
    crop_img1 = img[25:1033, 44:1046]
    blur1 = cv.medianBlur(crop_img1,7)
    crop_img2 = img[25:1033, 1046:2048]
    fl_crop_img = cv.flip(crop_img2, 1)
    blur2 = cv.medianBlur(fl_crop_img,7)
    crop_img3 = img[1033:2041,44:1046]
    blur3 = cv.medianBlur(crop_img3,5)
    crop_img4 = img[1033:2041,1046:2048]
    blur4 = cv.medianBlur(crop_img4,7)
    fl_crop_img1 = cv.flip(blur3, 0)
    fl_crop_img2 = cv.flip(blur4, -1)
    dst3 = cv.addWeighted(blur1, 0.5, blur2, 0.5, 0.0)
    dst2 = cv.addWeighted(fl_crop_img1, 0.5,
                          fl_crop_img2, 0.5, 0.0)
    dst = cv.addWeighted(dst3, 0.5, dst2, 0.5, 0.0)
    blur = cv.medianBlur(dst3,7)
    ret3,th3 = cv.threshold(blur,0,255,cv.THRESH_BINARY+cv.THRESH_OTSU)
    crop_th3 = th3[0:690,0:1002] #690
    crop2_th3 = th3[0:400,0:1002]
    c = crop_th3.flatten().tolist()
    c2 = crop2_th3.flatten().tolist()
    x = 255
    d = Counter(c)
    d2 = Counter(c2)
    num = d[x]
    num2 = d2[x]
    if 1000 < num < 10000 and num2 < 50:
        fid.write(name + ",2\n")
    else:
        fid.write(name + ",0\n")
if num > 40000:
    fid.write(name + ",0\n")
```

```
        fid.close()

def hmag(file):
    name = str(file)
    if name.endswith("hm.tif"):
        img = cv.imread(name)
        img2 = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
        blur = cv.medianBlur(img2,9)
        th3 = cv.adaptiveThreshold(blur, 255,
              cv.ADAPTIVE_THRESH_MEAN_C, cv.THRESH_BINARY, 17, 10)
        kernel = np.ones((6,6),np.uint8)
        opening = cv.morphologyEx(th3,cv.MORPH_OPEN,kernel, iterations = 2)
        blur = cv.medianBlur(opening,7)
        contours = cv.findContours(blur, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
        cnts = imutils.grab_contours(contours)
        for contour in cnts:
            area = cv.contourArea(contour)
            x, y, w, h = cv.boundingRect(contour)
            if x > 20:
                if area > 40.0:
                    cv.drawContours(img, contour, -1, (255,0,0), 4)
                    extLeft = tuple(contour[contour[:, :, 0].argmin()][0])
                    x1 = extLeft[0]
                    y1 = round(int(extLeft[1])/10)*10
                    coo = x1,y1
                    if coo not in new_coordinates:
                        new_coordinates.append(coo)
        file2 = name.split("/")
        end = str(file2[-1]).split("_")
        filename = str(end[0].strip("][',") + "_" + end[1].strip("][',"))
        filt_pts = []
        if len(new_coordinates) != 0:
            fname = str(filename + ".csv")
            f = open(fname,"a")
            f.write("x,y,z,a\n")
            for i in range(len(new_coordinates)-1):
                x,y = new_coordinates[i]
                x2,y2 = new_coordinates[i+1]
                if abs(y - y2) > 50 and abs(x - x2) > 50:
                    if x < 1024 and y < 1024:
                        ynm = abs(x - 1024) * 0.0105
                        xnm = abs(y - 1024) * 0.0105
                    if x < 1024 and y > 1024:
                        ynm = abs(x - 1024) * -0.0105
                        xnm = abs(y - 1024) * 0.0105
                    if x > 1024 and y > 1024:
```

```
                ynm = abs(x - 1024) * -0.0105
                xnm = abs(y - 1024) * -0.0105
            if x > 1024 and y < 1024:
                ynm = abs(x - 1024) * 0.0105
                xnm = abs(y - 1024) * -0.0105
            origxy = name.split("_")
            x0 = str(origxy[0]).strip("x")
            y0 = str(origxy[1]).strip("y")
            x_coord = float(x0) + xnm
            y_coord = float(y0) + ynm
            nx = float(Decimal(x_coord).quantize(Decimal("1.00")))
            ny = float(Decimal(y_coord).quantize(Decimal("1.00")))
            nxny = str(nx) + "," + str(ny)
            n = int(x),int(y)
            cv.circle(img, n, 8, (0, 255, 0), -1)
            cv.putText(img, nxny, (n), cv.FONT_HERSHEY_PLAIN, 2, (0,0,255), 2)
            filt_pts.append(nxny)
    if len(filt_pts) != 0 and len(filt_pts) > 3:
        pts = random.sample(filt_pts, 3)
        for point in pts:
            x1,y1 = point.split(",")
            f.write(str(x1) + "," + str(y1) + ",0,0\n")
    if len(filt_pts) != 0 and len(filt_pts) <= 3:
        for point in filt_pts:
            x1,y1 = point.split(",")
            f.write(str(x1) + "," + str(y1) + ",0,0\n")

    nfilename = str(filename + "_collectedpts.tif")
    cv.imwrite(nfilename, img)
    new_coordinates.clear()
    filt_pts.clear()
    f.close()
    cv.namedWindow('image',cv.WINDOW_NORMAL)
    cv.resizeWindow('image',800,800)
    cv.imshow('image',img)
    cv.waitKey(0)
    cv.destroyAllWindows()


def distance(kpt1, kpt2):
    arr = np.array([kpt1.pt, kpt2.pt])
    return np.sqrt(np.sum((arr[0]-arr[1])**2))

def distance2(kpt1, kpt2):
    arr = np.array([kpt1, kpt2])
    return np.sqrt(np.sum((arr[0]-arr[1])**2))
```

```
class MyHandler(FileSystemEventHandler):
    def on_created(self,event):
        file = str({event.src_path}).strip("}{'").split("/")
        end = str(file[-1])
        end2 = end.split("_")
        filename = str(end2[-1]).strip("]'")
        if filename == "380.tif":
            sep = "/"
            path = sep.join(file[:-1])
            os.chdir(path)
            time.sleep(3)
            analyze(end)
        if filename == "incdif.tif":
            sep = "/"
            path = sep.join(file[:-1])
            os.chdir(path)
            time.sleep(3)
            incdif(end)

        if filename == "hm.tif":
            sep = "/"
            path = sep.join(file[:-1])
            os.chdir(path)
            time.sleep(3)
            hmag(end)



if __name__ == "__main__":
    print("Keep me open for live monitoring of data files!")
    event_handler = MyHandler()
    observer = Observer()
    observer.schedule(event_handler, path='/mnt/c/', recursive=True)
    observer.start()
    try:
        while True:
            time.sleep(1)
    except KeyboardInterrupt:
        observer.stop()
    observer.join()
```

4.7.2.4 Incident Diffraction Screening Table

Particle Recognition Method	Particles found in LM 380x	Coordinates driven to 2600x	Diffraction patterns collected	No Diffraction	Percent Diffracting Particles
CrystalEyes_v1.py	245	65	66	33	50%
CrystalEyes_v2.py	236	177	141	90	36%
CrystalEyes_v3.py	255	41	86	19	78%
Manually Determined	n/a	n/a	38	3	92%

Table 4.1 Optimization of image recognition scripts based on percentage of diffracting particles.