**Title**
tBeam—A Fast Model to Estimate Energy Consumption Due to Pavement Structural Response User Manual



**Author**
Weissman, Shmuel L.

**Publication Date**
2021-06-01

**DOI**
10.7922/G2J964P8

**June 2021**
**Reprint Report: UCPRC-RP-2021-01**

# tBeam—
# A Fast Model to Estimate Energy Consumption Due to Pavement Structural Response
# User Manual

**Author:**
Shmuel L. Weissman, Symplectic Engineering Corporation

This report is based on research performed by the Symplectic Engineering Corporation on behalf of the University of California Pavement Research Center for the California Department of Transportation, and is reprinted here in its original form.

Work Conducted as part of University of California Research Agreement #201702289-03

**PREPARED FOR:**

California Department of Transportation
Division of Research, Innovation, and System Information
Office of Materials and Infrastructure

**PREPARED BY:**

University of California
Pavement Research Center
UC Davis, UC Berkeley

# TECHNICAL REPORT DOCUMENTATION PAGE

| 1. REPORT NUMBER<br>UCPRC-RP-2021-01 | 2. GOVERNMENT ASSOCIATION NUMBER | 3. RECIPIENT'S CATALOG NUMBER |
|---|---|---|
| 4. TITLE AND SUBTITLE<br>tBeam—A Fast Model to Estimate Energy Consumption Due to Pavement Structural Response User Manual | | 5. REPORT PUBLICATION DATE<br>June 2021<br><br>6. PERFORMING ORGANIZATION CODE |
| 7. AUTHOR(S)<br>Shmuel L. Weissman | | 8. PERFORMING ORGANIZATION REPORT NO.<br>UCPRC-RP-2021-01<br>UCD-ITS-RR-21-32 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS<br>Symplectic Engineering<br>2901 Benvenue Avenue<br>Berkeley, CA 94705 | | 10. WORK UNIT NUMBER<br><br>11. CONTRACT OR GRANT NUMBER<br>UC subcontract number #201702289-03 as part of Caltrans contract number 65A0628 |
| 12. SPONSORING AGENCY AND ADDRESS<br>California Department of Transportation<br>Division of Research, Innovation, and System Information<br>P.O. Box 942873<br>Sacramento, CA 94273-0001 | | 13. TYPE OF REPORT AND PERIOD COVERED<br>Reprint Report<br>November 2017 to July 2020<br><br>14. SPONSORING AGENCY CODE |

15. SUPPLEMENTAL NOTES
DOI:10.7922/G2J964P8

16. ABSTRACT

This document constitutes the user manual for tBeam, standalone software for the analysis of energy dissipation in pavements under moving vehicles. tBeam was developed as part of the improvement of modeling capabilities for environmental life cycle assessment of pavements being conducted by the University of California Pavement Research Center for the California Department of Transportation. tBeam is finite element based, employing multi-layered three-node Timoshenko beam elements resting on a viscoelastic Winkler foundation. It provides an approximation of the deflection bowl of pavements and the energy dissipated in pavement structures when subjected to loads moving at constant velocities. tBeam supports two loading options: a uniform pressure (per unit length) applied to a segment at the center of the beam, and a rolling rigid wheel. To achieve numerical efficiency the load-beam-foundation system is represented relative to a moving coordinate system attached to the moving load. The higher efficiency is made possible because, in this framework, an observer attached to the moving coordinate system perceives a "static" state (i.e., independent of time). The standalone tBeam software serves two purposes. First, to provide developers of pavement LCA tools a "guide" as to how to integrate tBeam technology into their program. To this end, the "main" of tBeam can be used as "guide" for integrating tBeam capabilities within the LCA tool. Second, tBeam capabilities are relevant to pavement research in general. Thus, it could represent a useful addition to the toolset for pavement viscoelastic mechanics.

| 17. KEY WORDS<br>pavement energy dissipation, viscoelastic, life cycle assessment, pavement-vehicle interaction, structural response | 18. DISTRIBUTION STATEMENT<br>No restrictions. This document is available to the public through the National Technical Information Service, Springfield, VA 22161 | |
|---|---|---|
| 19. SECURITY CLASSIFICATION (of this report)<br><br>Unclassified | 20. NUMBER OF PAGES<br>30 | 21. PRICE<br>None |

## UCPRC ADDITIONAL INFORMATION

| | |
|---|---|
| 1. DRAFT STAGE<br>Final | 2. VERSION NUMBER<br>1 |
| 3. PARTNERED PAVEMENT RESEARCH CENTER STRATEGIC PLAN ELEMENT NUMBER<br>4.73 | 4. DRISI TASK NUMBER<br>3210 |
| 5. CALTRANS TECHNICAL LEAD AND REVIEWER(S)<br>Deepak Maskey | 6. FHWA NUMBER<br>CA213210A |

7. PROPOSALS FOR IMPLEMENTATION
The code for tBeam should be implemented in some form in the pavement life cycle assessment tool eLCAP to provide the capability to analyze the environmental impacts of vehicle propulsion energy dissipation due to the structural response of the pavement under the moving vehicle.

8. RELATED DOCUMENTS
Simulation of Cumulative Annual Impact of Pavement Structural Response on Vehicle Fuel Economy for California Test Sections, UCPRC-RR-2015-05; Investigation of the Effect of Pavement Deflection on Vehicle Fuel Consumption: Field Data Collection, Data Processing and Empirical Analysis, UCPRC-RR-2021-03 (in progress)

9. LABORATORY ACCREDITATION

| | | | | |
|---|---|---|---|---|
| S. Weissman<br>**FIRST AUTHOR** | J.T. Harvey<br>**TECHNICAL REVIEW** | J.T. Harvey<br>**PRINCIPAL INVESTIGATOR** | D. Maskey<br>**CALTRANS TECH LEAD** | T.J. Holland<br>**CALTRANS CONTRACT MANAGER** |

# DISCLAIMER

This document is disseminated in the interest of information exchange. The contents of this report reflect the views of the authors who are responsible for the facts and accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the State of California or the Federal Highway Administration. This publication does not constitute a standard, specification or regulation. This report does not constitute an endorsement by the California Department of Transportation (Caltrans) of any product described herein.

For individuals with sensory disabilities, this document is available in braille, large print, audiocassette, or compact disc. To obtain a copy of this document in one of these alternate formats, please contact: the California Department of Transportation, Division of Research Innovation, and Systems Information, MS-83, P.O. Box 942873, Sacramento, CA 94273-0001.

**For more information:**

University of California Pavement Research Center, Davis
One Shields Avenue, Davis, CA 95616

University of California Pavement Research Center, Berkeley
1353 S. 46th St., Bldg. 452, Richmond, CA 94804

*www.ucprc.ucdavis.edu*

May 12, 2019

# tBeam

# A Fast Model to Estimate Energy Consumption Due to Pavement Structural Response

## User Manual

By
Shmuel L. Weissman


2901 Benvenue Ave., Berkeley, California 94705
telephone (510) 528-1251 • fax (510) 528-7102 • www.symplectic.com

# Table of Contents

# Acknowledgments

A University of California Research Agreement #201702289-03 funded this work. We wish to acknowledge the contributions by Drs. Rongzong Wu, Ali Azhar Butt, and John T. Harvey of the University of California Pavement Research Center.

# Executive Summary

This document constitutes the user manual for tBeam, standalone software for the analysis of pavements. tBeam is finite element based, employing multi-layered three-node Timoshenko beam elements resting on a viscoelastic Winkler foundation. It is intended to provide an approximation of the deflection bowl of pavements and the energy dissipated in pavement structures when subjected to loads moving at constant velocities. tBeam supports two loading options: a uniform pressure (per unit length) applied to a segment at the center of the beam, and a rolling rigid wheel.

To achieve numerical efficiency the load-beam-foundation system is represented relative to a moving coordinate system attached to the moving load. The higher efficiency is made possible because, in this framework, an observer attached to the moving coordinate system perceives a "static" state (*i.e.,* independent of time).

The motivation for tBeam development is the need for its capabilities to facilitate a sustainable pavement design procedure that accounts for energy dissipated in pavement structures. The standalone software serves two purposes. First, to provide developers of the aforementioned sustainable analysis tool a "guide" as to how to integrate tBeam technology into their program. To this end, the "main" of tBeam can be used as "guide" for integrating tBeam capabilities with the sustainable analysis tool. Second, tBeam capabilities are relevant to pavement research in general. Thus, it could represent a useful addition to their toolset.

# 1 Introduction

This user manual provides the instructions for installing and running tBeam, standalone software that simulates the response of pavement structures to a load moving at a constant velocity. tBeam represents the pavement structure as a linear viscoelastic system comprised of a one-dimensional multi-layered Timoshenko beam resting on a Winkler foundation. tBeam's theoretical manual and validation simulations are presented in Weissman and Kelly [2019]. The purpose of this user manual is to instruct on applying tBeam, as standalone software, to the analysis of pavements. This manual is organized in five sections. A short overview of tBeam is provided in Section 2. Section 3 addresses installing tBeam. The input file format is described in Section 4. Running tBeam is discussed in Section 5. Finally, Section 6 explains the output.

*Remarks:*

- The input files used to run the examples provided in the tBeam Theoretical and Validation Manual (Weissman and Kelly [2019]) are provided along with the source code (in the inFiles directory). These files provide examples for structuring tBeam simulations.
- The main objective in developing tBeam was to add structural analysis capabilities to the CalME Software developed by the UC Pavement Research Center. The version presented herein serves two objectives. First, to provide the developers of a working example of how to access tBeam capabilities. Specifically, the main function for the standalone tBeam software, contained in the CalME.cpp file, provides a simple surrogate for CalME. Second, give researchers access to the tBeam analysis capabilities outside the CalME framework.♦

# 2 tBeam overview

tBeam simulates the response of a multi-layered linear viscoelastic (shear deformable) Timoshenko beam resting on a viscoelastic Winkler foundation. (The user specifies the number of layers.) A generalized Maxwell model is used for the bending and shear beam responses, which for flexibility of usage are treated as independent of each other. The beam rests on a linear viscoelastic Winkler foundation, which is also represented as a generalized Maxwell model.

To achieve numerical efficiency, following Kelly [1962], tBeam is formulated relative to a moving coordinate system that is attached to the moving load. The load is assumed to be moving at a constant (positive) velocity. An observer attached to the moving coordinate system perceives a steady state of deformation (*i.e.*, independent of time). The implication is that the time-domain is removed, and a numerically efficient implementation is achieved. Note that although the observer perceives a static state, inertia is included in the analysis.

The beam-foundation system is subjected to one of two loading options: uniformly distributed pressure, and rolling rigid wheel. The uniformly distributed pressure (per unit length) is applied to the center section of the beam, with the length of the loaded section being specified by the user. The rolling wheel is applied to the center of the beam, and the contact zone is determined as part of the solution. In this case the user specifies the radius of the wheel and the magnitude of the applied load. Determining the contact zone is a nonlinear problem, requiring a number of iterations. Therefore, the numerical effort required for the rolling wheel solution is significantly higher than that consumed when considering a uniformly distributed pressure.

*Remark:* The formulation is relative to a moving coordinate system that is attached to the moving wheel. Consequently, the center of the beam in the model is a moving location in "real world." It is, in fact, the "current" location of the wheel. ♦

The analytical solution requires integration from $-\infty$ to $+\infty$. The solution, however, is such that at a certain distance in front of the load, the load does not yet affect the state of deformation, and at a certain distance behind the load the deformation attained an asymptotic state. tBeam is a finite element-based code. It takes advantage of the state of deformation, so that only the relevant center portion of the beam must be modeled. The user specifies the length of the finite element beam representation (*e.g.*, 20 meters in each direction from the center of the beam where the load is applied).

A detailed account of the theory underlying tBeam is provided in the tBeam Theoretical and Validation Manual (Weissman and Kelly [2019]). The Theory and Validation Manual also contains numerical examples demonstrating the performance of tBeam. (The input files used to run these examples are contained in the inFiles directory contained in the tBeam1.0 folder.)

# 3 Installing tBeam

The source files, a makefile, and a number sample input files are contained in tBeam.zip. To access these files, expand the tBeam.zip file to obtain a folder called tBeam. In this folder you will find the following file types:

- .cpp and .hpp files. These are the C++ source files for tBeam.
- A makefile. This file can be used to compile the software.
- A folder called inFiles containing the input files for the simulations presented in the tBeam Theoretical and Examples Manual (Weissman and Kelly [2019]).

The makefile is set up to work with the GNU g++ compiler. It can also be edited to use with another compiler. Once edited, the make utility can be invoked to compile the software by issuing, in a terminal window, the command: make. (Linux environment is assumed.) This would generate a few .o files and an executable file called tBeam. Once the software is installed, tBeam can be tested by running one of the provided example input files. For example, in the terminal window, while in the tBeam folder, issue the following command: ./tBeam inFiles/in5.1. It should run a simulation, using an input file called in5.1 (the input file for example 5.1 in Weissman and Kelly [2019]) located in the inFiles folder.

# 4 Input file

tBeam is set up as a standalone software requiring an input file, which is described in this Section. The input file observes the following conventions:

- tBeam converts the input to lower case, and is therefore case insensitive.
- tBeam uses space to delaminate different items. So if, for example, a certain set of material properties is the seventh set, it can be labeled as mat7, but not mat 7 (it will be assigned the name mat).
- If a certain line expects three arguments, but only two are provided, then tBeam may crash or give erroneous results. On the other hand, if tBeam expects three arguments and it is given more, then it treats the rest of the line (beyond the initial three arguments) as a comment. This facilitates adding explanatory information as is done in the sample input files.

tBeam recognizes the following keywords:

| | |
|---|---|
| cmat | enter a contact material |
| denergy | request the output of the dissipated energy for a specified tBeam object |
| editalltau | uniformly scale the characteristic period for all Maxwell elements in a pavement connected to a specified tBeam object |
| editcmat | edit a property of a contact material |
| editlt | uniformly scale the characteristic period for all Maxwell elements in a specified layer within a pavement associated with a specified tBeam object |
| editmat | edit a property in a pavement material object (influences only the copy provided to a specific pavement connected to a specified tBeam object) |
| editrw | edit properties (radius and velocity) of a rigid wheel |
| editup | edit the properties of a uniform pressure load |
| pave | request construction of the mesh for the pavement |
| pmat | enter pavement material properties |
| print | output the transverse displacement and rotation at each beam node for a pavement associated with a specified tBeam object |
| printrw | output the position of the wheel's nodes for a rigid wheel associated with a specified tBeam object |
| psolve | solve for a uniformly distributed pressure applied to the center part of the beam |
| rrwsolve | solve for a rolling rigid wheel applied to the center of the center of the beam |
| rw | set up a rigid wheel |
| solstat | control the printout of solution information (*e.g.*, residual and energy norms) |
| tbeam-rw | set up a tBeam object linking a pavement and a rigid wheel |
| tbeam-up | set up a tBeam object linking a pavement and a uniform pressure load |
| upress | enter uniformly distributed pressure to be applied to the center part of the beam |

A keyword may be followed by further information on the same line, or a command block consisting of a number of lines. To be recognized as a keyword by tBeam, the keyword must be the first argument in the input line, and this line cannot be part of a command block. A description of the functionality associated with each keyword follows.

## 4.1 cmat – Define a Contact Material

Simulating a beam loaded by a rigid wheel requires the solution of a contact problem. In tBeam the contact problem is handled as node-on-node contact. cmat tells tBeam that you are inputting properties of the penalty spring forcing the no-penetration constraint. The syntax consists of the following two lines:

*cmat name*
*K power and lRef*

Name is a user defined (unique) name for this set. tBeam supports a nonlinear penalty spring, based on the following force-penetration law: $f = K\left(\frac{\epsilon}{L_{ref}}\right)^{power-1}$. In this law $f$ is the force applied to counter the penetration; K is a user-specified stiffness value; $L_{ref}$ is a user-specified reference length (it is recommended that $L_{ref}$ be taken as the length of the beam elements connected to the node on the pavement side of the contact); $p$ is a user specified power (in the examples in Weissman and Kelly [2019] $p = 2$); and $\epsilon$ is the "length" of the penetration.

## 4.2 denergy – Request Output of the Dissipated Energy

denergy requests tBeam to output the dissipated energy in the pavement. This block consists of a single line:

*deneregy name*

In the above *name* is the name of the tBeam object for which the output is desired.

## 4.3 editalltau – Scale the Characteristic Periods of All Maxwell Elements

This one-line command is used to scale the characteristic times of all Maxwell elements of a specified pavement (beam and foundation) previously set using the *pmat* command block. The command syntax is as follows:

*editalltau tbname value*

The first argument in this command line, *editalltau*, is the keyword invoking this command. The second argument, *tbname,* is a unique string that was used to identify an instance of tBeam when it was set (either with *tbeam-rw* or *tbeam-up*). The last argument, *value*, is the scaling factor. Note this scaling factor is applied to the original value characteristic periods set when the material instance was established with the *pmat* command. This command affects only the properties used by the specified tBeam object; other pavements using the same material set are unaffected by this change.

## 4.4 editcmat – Edit One of the Contact Material's Properties

This one-line command is used to edit properties of a contact material. The command line syntax is as follows:

*editcmat name prop-id value*

The first argument of the command line, *editcmat*, is the keyword that invokes this command. The second argument, *name*, is a unique string identifying the cMat instance to be edited that was established when the cMat instance was constructed using the *cmat* keyword. The third argument is used to identify which property is being edited. It can be one of the following three options: *k*, *lref*, or *power* (see Section 4.1). The last argument, *value*, is the new value assigned to the identified property.

## 4.5 editlt – Scale the Characteristic Periods of all Maxwell Elements in a Specified Layer

This one-line command is used to edit all the Maxwell elements of a specific layer (or foundation) of a specified pavement. This command affects only the properties used by the specified tBeam object; other pavements using the same material set are unaffected by this change. The command line syntax for scaling the characteristic periods for all Maxwell elements of a beam's layer is:

*editlt tbname type num value*

For scaling all the characteristic periods of all foundation Maxwell elements, the syntax is:

*editlt tbname type value*

The first argument, *editlt*, is the keyword invoking this command. The second argument, *tbname*, is a unique string that was used to identify an instance of tBeam when it was set (either with *tbeam-rw* or *tbeam-up*). The third argument, *type*, is a flag identifying if properties are changed for a layer of the beam or the foundation. *type* can be either "*l*" for editing layers, or "*f*" for editing the foundation. The fourth argument, which appears only when scaling properties of a layer, identifies the layer number. Note that per tBeam input conventions the first layer is the one directly above the foundation, the second is the second above the foundation, *etc*. Also, tBeam employs C++ array conventions, the value for the first layer is 0, the value for the second layer is 1, *etc*. The last argument, *value*, is the scaling factor. Note this scaling factor is applied to the original value characteristic periods set when the material instance was established with the *pmat* command.

## 4.6 editmat – Edit Material Properties

This one-line command is used to scale individual properties of the pavement material, previously instantiated with a *pmat* command. In order to facilitate usage of the same material model by multiple pavement structures, the properties stored with the material instance are left

unchanged. Instead, a set of weights specific to a pavement (and stored with the pavement information), are used to scale, not replace, the original properties.

*Note:* This command cannot be used to modify all properties. Specifically, it does not facilitate editing mass density; beam width; the Timoshenko shear coefficient; the number of layers; and the number of Maxwell elements for any component (beam bending, beam shear, and foundation).♦

This command has multiple branches; addressing foundation *vs*. layer, shear *vs*. bending, and Maxwell *vs*. non-Maxwell properties. All branches share the first three arguments, which are: the keyword invoking the command (*editmat*); a unique identifier for the tBeam instance, *name*; and an identifier of the target: "*f*" for foundation, and "*l*" for layer. (The tBeam instance links a pavement and a load; it is instantiated with either *tbeam-rw* or *tbeam-up*.) The syntax for modifying a non-Maxwell property of the foundation is:

*editmat name f flag value*

The first three arguments are described above. *flag* is one of two characters: "*e*" for the elastic spring response, or "*v*" for the standalone dashpot. The last argument, *value*, is the scaling factor to be applied to the property.

The command syntax to edit a Maxwell property of the foundation is:

*editmat name f flag num value*

In this case *flag* is one of the following two-character sequence: "*me*" for scaling the property of the spring of a Maxwell element, and "*mt*" to scale the property of the characteristic period of the Maxwell element. The fifth argument is an integer identifying the Maxwell element to be edited. (tBeam follows the C++ convention of array indexing, and so the first Maxwell is numbered 0.) The last argument is the scaling factor.

Editing a beam property adds the complexity of having to identify the layer, and whether it is a bending or shear property. Accordingly, syntax for editing a non-Maxwell element property is:

*editmat name l type flag lnum value*

The first three arguments are explained above. *type* is a single character flag that can assume one of the following two values: "*b*" for bending or "*s*" for shear. Following the procedure for the foundation, *flag* assumes one of the following two single character values: "*e*" for the elastic (spring) property, or "*v*" for the standalone viscous (dashpot) property. The sixth argument, *lnum*, is an integer identifying the layer's number starting from the bottom. Following C++ conventions, the number of the bottom layer is 0. Finally, *value* is the scaling factor to be applied.

The syntax to edit a bending or shear Maxwell element property is:

*editmat name l type flag lnum mnum value*

In this case *flag* assumes one of the two two-character sequences: "*me*" for editing the elastic property of a Maxwell element, or "*mt*" to edit the characteristic time. *lnum* is layer's number, starting from the bottom layer, whose number is 0. *mnum* is the number of the Maxwell element to edit, with the first numbered 0. Lastly, *value* is the scaling factor to apply.

## 4.7 editrw – Edit Attributes of a Rigid Rolling Wheel

This one-line command allows the editing of two of the properties of a rigid wheel previously defined by the *rw* command. The editable properties are the applied force, *f*, and the velocity of the wheel, *v*. Editing of the radius of the wheel is not allowed because, if the wheel is in use, its nodes have been generated. The syntax for this command is:

*editrw name type value1 value2*

The first argument, *editrw*, is the keyword invoking this command. The second argument, *name*, is a unique string assigned to identify the rigid wheel instance when it was created with the *rw* command. The third argument, *type*, admits three values: *v* for editing the velocity; *f* for editing the force; and *vf* for editing both the velocity and force. The last two arguments are numerical values to be assigned to the attributes being modified. *value2* is used only in conjunction with type *vf* (*value1* is assigned to the velocity).

## 4.8 editup – Edit Attributes of a Uniformly Distributed Pressure Per Unit Length

This one-line command allows the editing of the properties of a uniformly distributed pressure (per unit length) previously defined by the *upress* command. The editable properties are the velocity, *v*; the pressure half length, *a* (the pressure is applied to the interval $x = [-a, +a]$); and the (line) pressure, *p*. The syntax for this command is:

*editup name type value1 value2 value3*

The first argument, *editup*, is the keyword invoking this command. The second argument, *name*, is a unique string assigned to identify the uniform pressure instance when it was created with the *upress* command. The third argument, *type*, admits four values: *a* for editing the half-length of the load; *v* for editing the velocity; *p* for editing the (uniform) pressure; and *apv* for editing the half length, force, and velocity. The last three arguments are numerical values to be assigned to the attributes being modified. *value2* and *value3* are used only in conjunction with type *apv*. In this case, *value1* is assigned to *a*, *value2* to *p*, and *value3* to *v*.

## 4.9 pave – Set Up a Pavement Structure

The *pave* command is used to generate the finite element mesh of Timoshenko beams used to represent the pavement structure (see *e.g.*, Timoshenko [1921,2]). The mesh construction requires the specification of a previously defined material set of properties (see *pmat*). The *pave* command requires a two-line block as follows:

*pave name mat*
*a b n1 n2*

The first line consists of the keyword *pave*, a unique name assigned to this pavement model for future identification, and the name of a previously defined material model (see *pmat*, Section 4.10). (tBeam supports the creation of multiple pavement instances.)

The second line provides the parameters used to generate the finite element mesh. The mesh is generated from $-b$ to $+b$ ($b$ is a length measure; it should be taken as 30-50 times the radius of the wheel). To improve accuracy, while keeping the computational effort reasonable, the mesh generated is not uniform. The user is asked to specify a second length, $a < b$, where the user can specify a finer mesh. In practice, $a$ should be about twice the radius of the wheel or the length of the applied uniform pressure. Next, the user specifies the number of beam elements. The first value, *n1*, specifies the number of elements in the interval $x \in [-b, -a]$. tBeam automatically generates a non-uniform mesh in this section, with elements getting larger as they get closer to the beam's end at $x = -b$. The reason for this non-uniform mesh is that the accuracy of finite elements diminishes with increased gradients of the state variables. The solution of the load-pavement system is such that the steepest gradients are in the vicinity of the load. Therefore, a finer mesh should be applied near the loaded section. The user has no control over the gradation of the mesh. A mirror image mesh is generated in the interval $x \in [+a, +b]$. The second value, *n2*, specifies the number of elements in the center section of the model, $x \in [-a, +a]$. The mesh in this section is uniform. To improve accuracy of the solution it is recommended that a finer mesh, relative to the exterior sections, be used for the center section, $x \in [-a, +a]$.

## 4.10 pmat – Define a Pavement Material

The *pmat* block is where properties of the pavement structure are provided. This block is organized as follows. The first line starts with the keyword *pmat*, followed by a user-specified name:

*pmat name*

The first argument, *pmat*, is the keyword invoking this command block. The second argument, *name*, is a unique string identifying this instance of *pmat* for subsequent use by a pavement structure.

The pavement is assumed to consist of a user-specified number of layers, each with its own properties, resting on a viscoelastic foundation. Accordingly, the syntax for the second line is as follows:

*b al #layers*

The first argument of the second line, *b*, specifies the width of the beam (all layers). The second argument, *al*, is the Timoshenko shear correction coefficient, which is commonly set to 5/6. The third argument, *#layers*, is an integer specifying the number of layers (above the foundation).

n blocks follow the second line, each representing a layer. (n is the number of layers in the beam.) These blocks are organized from the bottom up (*i.e.*, the first is the bottom layer, and the last is the top layer). The syntax for the first line in each layer's block is:

*rho h*

The first argument, *rho*, is the mass density for the layer. The second argument, *h*, is the thickness of the layer. Each layer requires the specification of bending and shear properties. Each model, bending and shear, is represented by a generalized Maxwell element consisting of an elastic element in parallel with a purely viscous element, in parallel with a number of Maxwell elements (the number is user specified). Accordingly, the layer's block continues with a description of the bending properties as follows: modulus of the purely elastic bending element (*E*), viscosity of the dashpot (*etab*), and the (integer) number of bending Maxwell elements (*mb*). This line is followed by *mb* lines (*mb* is the number of bending Maxwell elements), each describing the properties (modulus and characteristic time) of a bending Maxwell element. Accordingly, the bending part of the layer is given by:

*E etab mb*
*E-1 tau-1*
*E-2 tau-2*
.
.
.
*E-mb tau-mb*

The shear part of the layer follows the exact same format. With G replacing *E*, and *b* replaced by *s*. The shear blocks is given by:

*G etas ms*
*G-1 tau-1*
*G-2 tau-2*
.
.
.
*G-ms tau-ms*

Last comes the block describing the foundation. It consists of an elastic spring (*K*), in parallel with a dashpot (*etaf*), which is in parallel with *mf* Maxwell elements. (*mf* is user specified). The foundation block is given by:

*K etaf ms*
*K-1 tau-1*
*K-2 tau-2*
.
.
.
*K-mf tau-mf*

*Remarks:*

1. The number of Maxwell elements is specified individually for each layer/foundation, and within a layer is specified separately for the shear and bending parts. Thus, they do not need to be equal to each other.
2. All these blocks are specified without any line breaks.♦

The following sample example illustrates the *pmat* block with a 3 layer system (all layers are the same), and with one Maxwell element for each bending/shear/foundation system. (In this example properties are the same for different components although they need not be so.)

```
pmat mat1
0.1 0.83333333333 3      // thickness = 0.1 al = 5/6 3 layers
2300. 0.066666666667     // bottom layer rho & thickness
4.7897421e+10 0.   1     // Bending modulus, dashpot viscosity, & # of Maxwells
4.7897421e+10 0.1        // Bending Maxwell: modulus and characteristic time
1.8422085e+10 0.   1     // shear modulus, shear dashpot viscosity, & # Maxwells
1.8422085e+10 0.1        // shear Maxwell: modulus & characteristic time
2300. 0.066666666667     // middle layer rho & thickness
4.7897421e+10 0.   1     // Bending modulus, dashpot viscosity, & # of Maxwells
4.7897421e+10 0.1        // Bending Maxwell: modulus and characteristic time
1.8422085e+10 0.   1     // shear modulus, shear dashpot viscosity, & # Maxwells
1.8422085e+10 0.1        // shear Maxwell: modulus & characteristic time
2300. 0.066666666667     // top layer rho & thickness
4.7897421e+10 0.   1     // Bending modulus viscosity of dashpot # of Maxwells
4.7897421e+10 0.1        // Bending Maxwell: modulus and characteristic time
1.8422085e+10 0.  1      // shear modulus, shear dashpot viscosity, & # Maxwells
1.8422085e+10 0.1        // shear Maxwell: modulus & characteristic time
6.8424887e+06 0.   1     // foundation spring & dashpot values, & # of Maxwells
6.8424887e+06 0.1        // foundation Maxwell: stiffness & characteristic time
```

## 4.11 print – Output Pavement's Nodal Coordinates, Displacements, and Forces

This command is used to print the pavement's nodal transverse displacement, rotation, and applied force. For convenience, the node number (in the beam object), and the node's *x*-coordinate (the axis of the beam is taken along the x axis) are also reported. The *y*-coordinate of all beam nodes is taken to reside on the *x*-axis (*i.e.*, their y coordinate value is 0). The command syntax is:

*print tbname*

In the command line *print* is the keyword, and *tbname* is the name of the *tbeam* instance (generated either with *tbeam-rw* or *tbeam-up*) for which this information is required.

## 4.12 printrw – Request Output of Rigid Wheel's Current Nodal Position

This one-line command prints the x and y values of the rigid-wheel's nodes in the deformed configuration (*i.e.*, where it is when in contact with the pavement). This information allows plotting the beam and wheel to explore how well the contact properties imposed the no penetration constraint, as shown in Figure 4.1. The command syntax is:

*printrw tbname*

The first argument is the keyword invoking the command. The second argument, *tbname*, is the name of the tBeam instance previously created with the *tbeam-rw* command.

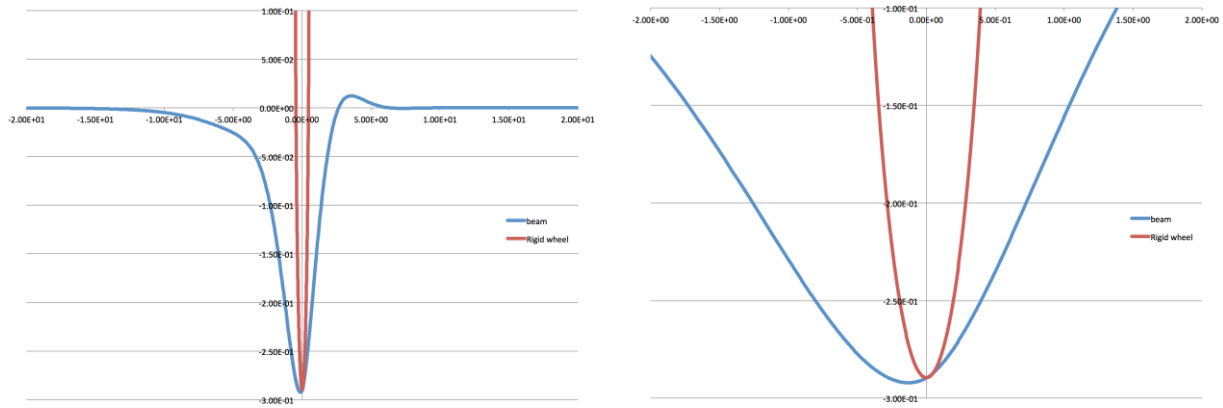


Figure 4.1: Wheel-beam contact: full beam (left), and close-up view (right).

## 4.13 psolve – Simulate a Pavement Structure Loaded by a Uniform Pressure

The *psolve* command instructs tBeam to perform a simulation using a uniformly distributed pressure (see *upress* command in Section 4.19). This analysis is linear viscoelastic. Consequently, it is very fast compared to the nonlinear rigid wheel-beam contact problem (see *rrwsolve* in Section 4.14). The command syntax for this command is:

*psolve tbname*

The first argument is the keyword invoking this command. The second argument, *tbname*, identifies a tBeam instance, previously instantiated with a *tbeam-up* command, which is to be analyzed.

## 4.14 rrwsolve – Simulate a Pavement Structure Loaded by a Rolling Rigid Wheel

The *rrwsolve* command instructs tBeam to run a simulation for the pavement-rigid wheel system identified in the specified *tbeam-rw* class instance. This solution is nonlinear because of the need to resolve the wheel-beam contact problem. Therefore, the computational effort is considerably

larger relative to that required for the simulation of a beam subjected to a uniformly distributed pressure (see *psolve*). The command line takes the form:

*rrwsolve tbname*

The first argument is the keyword invoking this command. The second argument, *tbname*, identifies a tBeam instance, previously instantiated with a *tbeam-rw* command, which is to be analyzed.

## 4.15 rw – Set Up a Rigid Wheel

The *rw* command is used to specify an instance of a rigid wheel. tBeam allows the user to specify multiple such instances, but only one can be paired with a given pavement for analysis (see *tbeam-rw*). The following two consecutive lines specify the *rw* command block:

*rw name*
*v r f cmat*

The first line consists of the *rw* keyword, and a unique string used to identify this instance of a rigid wheel. In the second line, *v* is the wheel's velocity; *r* is the radius of the wheel; *f* is the applied force, which should be negative (downwards) for the contact algorithm to work properly; and *cmat* is the name of a contact material set previously with the *cmat* command. If the user specifies the force as a positive value, tBeam automatically changes it to a negative value. (It must be a negative value for the contact to work properly.)

Note that the user does not specify the mesh for the rigid wheel. tBeam sets the mesh for the rigid wheel automatically. Specifically, it places the wheel nodes so that they are directly above the pavement's nodes, and the wheel center is placed at $x = 0$, and $y = r$.

## 4.16 solstat – Set Up a Flag for Pringing Solution Information

The *solstat* command is used to turn on and off the output of solution information such as the norm of the residual. This is a one-line command, containing the keyword and a value as follows:

*solstat flag*

solstat is the keyword, and *flag* is an integer. If *flag = 0*, the printout is suppressed. Otherwise, the solution statistics are reported. This information is useful for rigid wheel analysis because it shows if the analysis converged. If it did not, the solution is not reliable, and a change should be considered. This change would most likely involve reducing the stiffness of the contact material, *K* (see Section 4.1).

## 4.17 tbeam-rw – Set Up a tBeam Class Instance Linking a Pavement with a Rigid Wheel

This command pairs a structure with a rigid wheel, for the analysis. The command consists of a single line as follows:

*tbeam-rw name pavement wheel*

In this command line *tbeam-rw* is the keyword invoking the command; *name* is a unique string used to identify this instance of the tBeam class; pavement is the name of a pavement structure previously defined using the *pave* command; and wheel is the name of a rigid wheel previously defined using the *rw* command. Note that this command only sets the tBeam class instance. It does not actually run the simulation. Use the *rrwsolve* command to run a simulation.

## 4.18 tbeam-up – Set Up a tBeam Class Instance Linking a Pavement with a Uniform Pressure

This command pairs a structure with a uniform pressure for the analysis. The command consists of a single line as follows:

*tbeam-up name pavement press*

In this command line *tbeam-up* is the keyword invoking the command; *name* is a unique string used to identify this instance of the tBeam class; pavement is the name of a pavement structure previously defined using the *pave* command; and *press* is the name of a uniform pressure previously defined using the *upress* command. Note that this command only sets the tBeam class instance. It does not actually run the simulation. Use the *psolve* command to run a simulation.

## 4.19 upress – Define a Uniformly Distributed Pressure Per Unit Length

The *upress* command is used to specify a uniformly distributed pressure. The user can specify a number of sets, but only one will be used in the solution. The *upress* block consists of two lines as follows:

*upress name*
*v a p*

Above, *name* is a user-defined label used to identify this specific instance of *upress*. *v* is the velocity of the pressure. (Recall that the pressure represents a wheel moving at a constant velocity.) The pressure is applied to the middle section of the beam over the interval [$-a$,$+a$]. Lastly, *p* is the applied pressure (per unit length).

# 5 Running tBeam

To run tBeam simply issue the following command in a terminal window.

*./tBeam path/inFile*

The ./ in front of tBeam forces the system to look for the tBeam executable in the current directory (*i.e.*, it is assumed that you run tBeam while in the same directory as the code). If tBeam is invoked from another directory, add the path to tBeam folder to your .login file, or explicitly type the path to the tBeam executable. *path/inFile* is the path to and the name of the input file. If no input is specified, tBeam will look for a file called inFile in the current directory.

Once the command is issued, tBeam will run the simulations specified in the input file, and output the requested results. While it is running, unless suppressed with the *solstat* command, tBeam outputs information regarding the convergence (relevant only for understanding the rigid wheel analysis). Verify that the energy and norm of the right-hand side are very small (of the order of 10E-12 for energy 10E-8 for norm). That will be an indication that the analysis converged. Note that when simulating pavements subjected to uniform pressure the analysis is linear. Consequently, only one solution is performed, and the reported residual and energy norms are not expected to be small.

Finally, the various edit options (*e.g.*, *editlt* and *editalltau*) enable changing various properties of the analysis (*e.g.*, wheel velocity). Taking advantage of this capability, tBeam can be used to run an uninterrupted sequence of simulations. For the case where loading is by a rolling rigid wheel, this approach can reduce the computational effort considerably, especially if the change from one simulation to the next is relatively small (*e.g.*, when accounting for incremental damage).

# 6 Results

As detailed above (see Sections 4.2, 4.11, and 4.12), tBeam can output three different types of information. The *denergy* command outputs the energy dissipated in the pavement per unit length. The following is an example output, obtained for input file in5.4, of the dissipated energy per unit length:

Dissipated energy = 1.22098e-01

Note that there are no units. This is because tBeam does not impose units. It is up to the user to be consistent with the units.

The *print* command results in tBeam outputting the beam's nodal coordinates, displacements, and the applied external force (either due to the uniform pressure, or contact with a wheel). An example output, obtained for input file in5.4, is provided below:

```
node #   X-coor       Y-Disp        Rotation       Force
1     -1.00000e+02 -5.38157e-02  2.71008e-04  0.00000e+00
2     -9.96537e+01 -5.37316e-02  2.71008e-04  0.00000e+00
3     -9.93080e+01 -5.36647e-02  2.71008e-04  0.00000e+00
4     -9.89627e+01 -5.36112e-02  2.71008e-04  0.00000e+00
.
.
.
1198   9.89627e+01  2.68972e-04  2.71030e-04  0.00000e+00
1199   9.93080e+01  2.94257e-04  2.71030e-04  0.00000e+00
1200   9.96537e+01  3.37008e-04  2.71030e-04  0.00000e+00
1201   1.00000e+02  4.09941e-04  2.71030e-04  0.00000e+00
```

In this example (see input file in5.4 provided with the source code) the beam is represented by 1201 nodes, and the beam's domain is the closed interval $x \in [-100, +100]$.

Lastly, the *printrw* command is used to output the position of the rigid wheel. Plotting the beam and rigid wheel together (see, *e.g.*, Figure 4.1) provides a check on the performance of the contact algorithm. If penetration is detected, the stiffness of the contact spring should be increased (see cmat). The output for this command is in the following format:

```
node #    x-coor     y-coor
1      -5.00000e-01  4.35252e-01
2      -4.95000e-01  3.64719e-01
3      -4.90000e-01  3.35754e-01
4      -4.85000e-01  3.13700e-01
.
.
.
```

| | | |
|---|---|---|
| 198 | 4.85000e-01 | 3.13700e-01 |
| 199 | 4.90000e-01 | 3.35754e-01 |
| 200 | 4.95000e-01 | 3.64719e-01 |
| 201 | 5.00000e-01 | 4.35252e-01 |

# References

Kelly, J.M., [1962], "Moving Load Problems In the Theory of Viscoelasticity," Ph.D. Dissertation, Stanford University, April.

Timoshenko, S.P., [1921], "On the correction factor for shear of the differential equation for transverse vibrations of bars of uniform cross-section," *Philosophical Magazine*, 41: 744-746.

Timoshenko, S.P., [1922], "On the transverse vibrations of bars of uniform cross-section," *Philosophical Magazine*, 43: 125-131.

Weissman, S.L. and J.M. Kelly, [2019], "tBeam, A Fast Model to Estimate Energy Consumption Due to Pavement Structural Response. Theoretical and Validation Manual," Symplectic Engineering Report, May.