

FTire/calc

FE-Based FTire Parameter Calculation
Documentation and User's Guide

Contents

1	<i>FTire/calc</i>: Scope and Mode of Operation	1
2	<i>FTire/calc</i> Workbench	2
2.1	Columns, Buttons, File Selection Lists	2
2.2	Input and Output Files	3
2.3	Input File Parameterization	3
3	Launching and Controlling <i>FTire/calc</i>	4
4	Tire Design Data File (.tdd-File)	5
4.1	Data Required by <i>FTire/calc</i>	5
4.2	A Sample .tdd-File	9
4.3	Description of Tire Design Data in .tdd-File	14
5	Additional Tools	18
5.1	Analysis of Static Load-Cases	19
5.2	Modal Analysis	20

General Remarks

This documentation describes the tire simulation tool *FTire/calc*, which can be used to generate data files for the Flexible Ring Tire Model (*FTire*) on basis of tire design data. For more material about *FTire*, and other tire simulation tools, please refer to the respective cosin documentation chapters, or visit cosin.eu.

1 *FTire/calc*: Scope and Mode of Operation

FTire/calc is a tool to calculate (or estimate in a qualified way), input data for *FTire*. This is done by generating and analyzing a coarse finite-element mesh on basis of certain tire design data. These data, in turn, can be parameterized using very few key values of the tire design, like for example

- tire and rim size in standard notation,
- few dimensionless x/y-data pairs describing tire cross-section (automatically scaled by tire size),
- belt and tread width relative to tire width,
- tire total mass,
- estimated relative mass shares of bead, side-walls, belt, and tread,
- belt chord angle,
- belt chord thread diameter,
- belt chord thread density,
- young's modulus of carcass chord thread,
- carcass chord thread diameter,
- carcass chord thread density,
- young's modulus of tread rubber,
- tread depth,
- estimated tread rubber friction coefficients, etc.

FTire/calc uses the coarse FE model to automatically calculate some modal properties and to solve a series of non-linear static load-cases. The results of these 'virtual measurements' determine the structural stiffness and damping properties of *FTire*. *FTire/calc* uses them to automatically generate a ready-to-run *FTire* data file.

Some parameters of *FTire*, however, cannot be calculated in such an easy way, like the friction properties of tread rubber on road surface, for example. These parameters will just be copied from the *FTire/calc* input file to the *FTire* data file.

FTire/calc also provides some means to analyze the design data prior to generating an *FTire* data file. It can

- display eigen-frequencies and animated mode shapes of the internally generated FE model,
- calculate single static load-cases, and
- display distorted FE grids, stresses, and ground pressure distribution on flat surface and special obstacles.

After having completed the generation of the *FTire* data file, *FTire/calc* can be used

- to check the file for '*FTire* integrity' (that is, whether or not *FTire* pre-processing runs successfully),
- to analyze and process the file by the *FTire/tools* suite,
- or to launch stand-alone *FTire* simulations with *FTire/sim*.

2 FTire/calc Workbench

Like all other *cosin* products, *FTire/calc* is launched using a Tcl/Tk-based workbench, see fig. 1. This workbench has nearly the same appearance under all supported platforms (which are at present Win98/NT/2000/XP. HP-UX, Linux, Sun Solaris, and Irix solutions can be made available upon request).

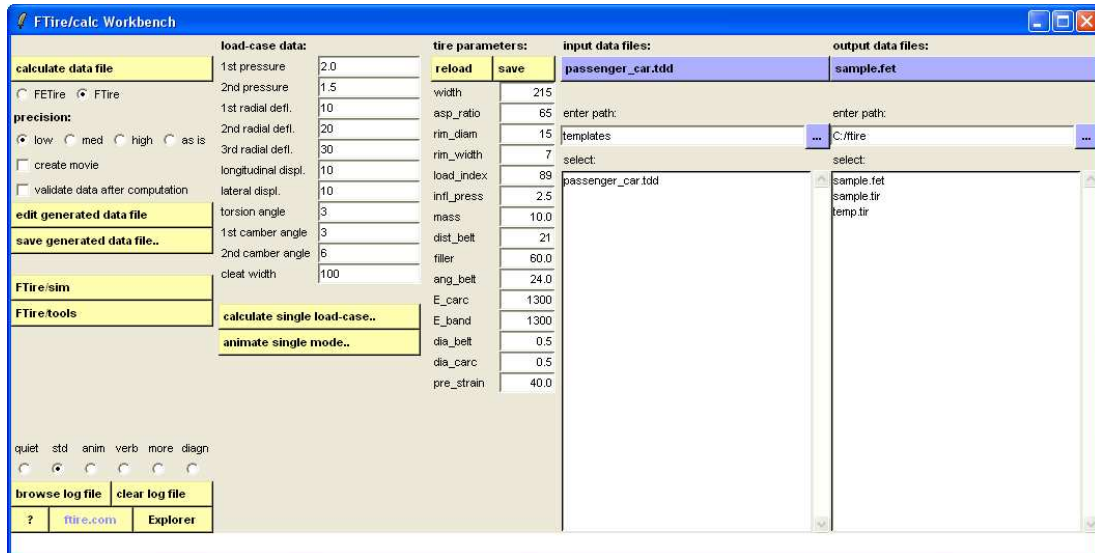


Figure 1: FTire/calc work-bench and its 5 columns

2.1 Columns, Buttons, File Selection Lists

The workbench consists of 5 columns:

1. the **action** column (red ellipse in figure 1), which is used to launch *FTire/calc*, and to study the results,
2. the **control and analysis** column (blue ellipse in figure 1), being used to control some input data to *FTire/calc*, and to analyze these input data prior to generation of the *FTire* data file, in a more detailed way than it is necessary for the generation of an *FTire* data set,
3. the **parameterization** column (grey ellipse in figure 1), to specify some key tire design data values used in the tire design data file (the choice of these values can be easily customized),
4. the **input files** column (green ellipse in figure 1), being used to select and edit the tire design data input file, and
5. the **output files** column (yellow ellipse in figure 1), which is used to browse output files from previous runs, if any available.

As in all *cosin* work-benches, there are two types of **buttons**.

Any **yellow** button is called an **action** button. Action buttons can be left-clicked to perform the respective action described in the button text.

The **blue** buttons are **analysis** buttons, serving to open a data file in a way which is appropriate for the type of the file. The name of the file appears as button text and is dynamically refreshed.

Analysis buttons always appear on top of a **file selection list**. The files being listed here all have extensions taken from a predefined internal list, and are all located in the same folder. This folder is remembered by *FTire/calc* from the most recent invocation, and can be changed using the '**enter path**' entry field. *FTire/calc* accepts both Windows and Unix-style path descriptions. In Windows, you can use forward or backward slashes, with or without drive letter and colon. Path names may contain blank spaces.

Under Windows, through clicking the blue '**...**'-**button** on the right-hand side of the entry field, the path choice as well as the activation of a file can be done using Windows's file selection window.

As soon as the user enters a new path, after hitting the '**enter**' or '**return**' key, *FTire/calc* automatically collects and displays all relevant files in this path. **Activating** (which means '**selecting for further analysis**') of such a file is simply done by left-clicking its entry in the list. Only one file can be active at a time, being indicated by the text in the analysis button.

Any file in the file selection list can be **duplicated**, by first activating the file with a single click, and then hitting the **space bar**. The copy of the file will share its name, appended to the prefix '**copy_of_**'. Under Windows, it can be renamed using Explorer, either by clicking the respective action button, or by right-clicking into the 'enter path'-field.

2.2 Input and Output Files

FTire/calc provides two file selection lists in the **input** and the **output** file columns.

The **input** files are those with extensions

- **.tdd** (tire design data), the input file type for *FTire/calc*, and
- **.fet** (*FETire* data-files, *not* considered here).

The analysis button in the input files column will **edit** the active file, using the user's preferred editor. Default editor is Notepad, but can be changed in the *cosin/products* root menu. This is the menu which appears immediately after installation of *cosin/products*.

The **output** files are those with extensions

- **.tir**, **.ft** (that is, all *FTire* data files in all supported formats),
- **.rt**, **.bi**, **.bs**, **.fet** (other types of tire model data files which are not considered here).

Again, the analysis button in the output files column will **edit** the active file.

2.3 Input File Parameterization

If the active input file is **parameterized** (cf. *cosin/io* documentation), indicated by the presence of the `$parameters` data-block, all parameters will be made accessible for easy change in the **parameterization column**.

The parameterization column is automatically refreshed when the active file changes. The parameter values displayed in the entry fields will be used in the next calculation, **instead** of the respective values in the file, which can be seen when editing the file.

Moving the mouse pointer over or into an entry field will display the meaning and dimension of the respective parameter in the white **message line** at the bottom of the work-bench. More precise: it is the comment string after the exclamation mark in the definition line of the parameter which will be shown as message. This comment is completely free to changes by the user, using the text editor.

Please note: any **change in an entry field** of the parameterization column

- will be used in the next calculation, instead of the value in the file,
- but will not change the value in the file itself, and thus will not be permanent.

On the other hand: any **change in the input file**

- **will** of course be permanent when leaving the work-bench,
- but **will not** be automatically taken over into the respective field of the parameterization column, and thus will **not** be automatically used in the next calculation.

In order to make changes in an entry field permanent, or undo them, use the **reload** and **save** action buttons on the top of the column. **Reload** will refresh the entry field values, using those actually stored in the file, whereas **save** will update the file, using the actually displayed values in the entry fields.

As described in [cosin/io](#) documentation, parameter values can be **numerical** values as well as very general **arithmetic expressions**, using other parameters previously defined.

3 Launching and Controlling *FTire/calc*

An *FTire/calc* run can be easily launched by clicking the '**calculate data file**' button in the action menu. Before doing this, please make sure that

- a **valid tire design data file** (.tdd-file) is activated as input file,
- '**FTire**' is checked in the **first** 'radio-button' line of the action menu,
- the **precision** (grid mesh size) of the internally used FE model is chosen to be one of **low**, **medium**, or **high** in the second radio-button line,
- the **format** of the *FTire* data file to be created is chosen to be either MSC.ADAMS/TeimOrbit or *cosin/io* (by clicking the respective radio-button in the **third** radio-button line),
- '**verbose**' is checked if you want to watch the progress of the calculation in greater detail in the message window (usually not needed),
- and '**check FTire data integrity**' is checked if the generated *FTire* data file is to be automatically pre-processed after calculation.

Depending on these choices, and the properties of the tire data, the calculation will last from several seconds to several minutes. Because the actual time needed for computation is hard to be foreseen,

it does not make much sense to show a progress bar. Several large and highly nonlinear systems of equations are solved iteratively. The speed of convergence, and thus the overall computation time, extremely depends on tire design data and on actual values of control variables.

During calculation, experienced users might wish to watch the progress, indicated by several intermediate results which are shown in a message window. The message window can be scrolled (however, trying to do so might become a bit frustrating: whenever new results are put out, the scroll position will jump to the last message).

This window is closed shortly after completion of the calculations. However, the **'browse log file'** button will allow to browse and print all intermediate results and all other messages of the most recent program run, even if the message window had been closed already.

Moreover, the log file can already be browsed during a running calculation, rather than trying to scroll the message window. You will see all results produced up to the very moment when opening the log file. Closing and re-opening will refresh the display.

The action button **'edit generated data file'** can be used to browse or edit the most recently generated *FTire* data file. This file is called temp.tir and resides in the output data folder, specified in the entry field of the output data column.

Clicking **'save generated data file'** allows to specifying a new name of the data file to copy the temporary data file into.

The two action buttons described above will only be visible if an *FTire* data file had been calculated already in the current session.

For deeper reaching analyses of the generated *FTire* data file, you might want to change to the [FTire/sim](#) work-bench, being initiated by the respective action button **'FTire/sim'**. *FTire/sim* is the starting point for stand-alone simulations with *FTire*, optionally connected to a very general non-linear 'quarter-car' model. The design of the *FTire/sim* work-bench is quite similar to that of *FTire/calc*.

Moreover, the button **'FTire/tools'** provides access to yet another work-bench with more [FTire utilities](#), like

- mode-shape animation for unloaded and loaded tire,
- statics calculation on flat surface and on several special obstacles,
- linearization,
- data file format change between all supported formats, including Tydex format,
- estimation of valid ranges for certain structural parameters, etc.

4 Tire Design Data File (.tdd-File)

4.1 Data Required by *FTire/calc*

The tire design data file (.tdd-file) is the input file to *FTire/calc*. It is given in [cosin/io](#) format, and consists of several data-blocks, each beginning with a \$-sign, followed by a predefined data-block name.

The data contained in a .tdd-file can be roughly distinguished into three different types:

- generally available basic data, like tire size, rim size, and load-index,
- data that is to be measured, and that will be copied directly to the *FTire* data file without any processing in *FTire/calc*, like tread rubber stiffness, damping, and friction properties, and
- structural data of the tire belt and carcass layers. These data are used to create a coarse FE model, which in turn is used to calculate those modal and static tire properties that are expected in the *FTire* data file.

The structural data give a description of the tire carcass and belt layers, separately for side-walls and belt region. This description is made of several lists, one for each typical position in side-wall and belt.

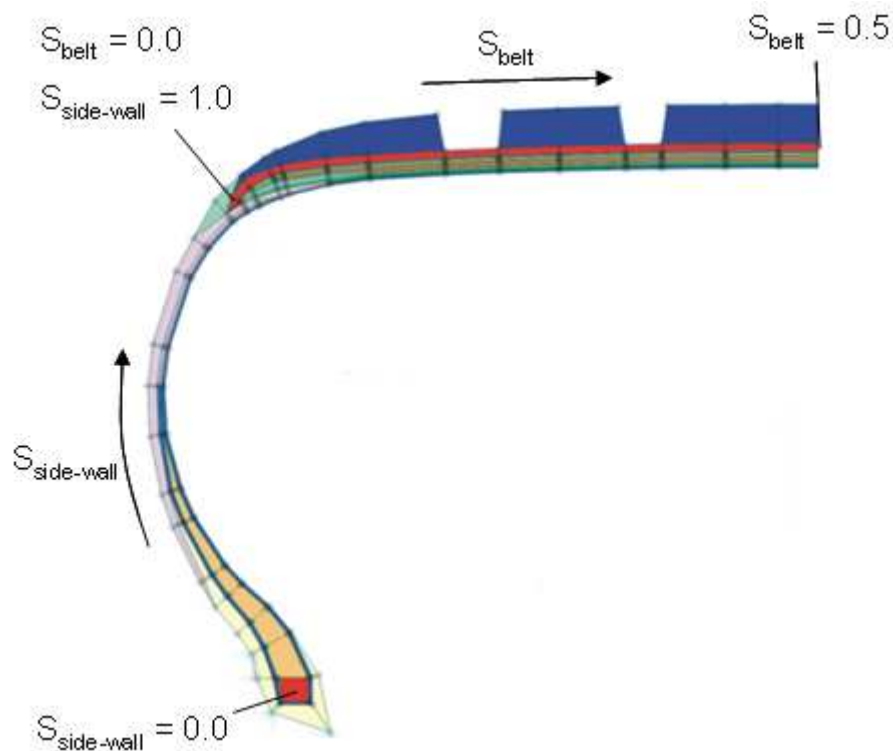


Figure 2: tire cross-section

To uniquely locate these positions, side-wall and belt are both parameterized, using a dimensionless curvilinear co-ordinate s . The value of this variable ranges from 0 to 1, roughly following the tire carcass in transversal direction (see figure 2. The FE grid used in this figure is **not** the grid of *FTire/calc*). For the side-walls, '0' means center of bead coil, '1' means start of the first belt layer. For the belt, '0' is leftmost begin of a belt layer, and '1' is rightmost end of such a layer. For symmetrical tires, 0.5 is the so-called belt zenith.

A single layer list refers to a certain value of $s_{side-wall}$ or s_{belt} , respectively. It starts with the inner-most tire layer, the 'inner liner', and ends with the outermost rubber layer. For the belt region, the tread layer, which is underlying wear, is taken into account not here, but elsewhere. That means, a belt layer list ends with the 'protector rubber', which is defined to be the tread rubber starting at steel belt surface, and ranging only up to the **bottom of the tread grooves**.

Any layer list may consist of arbitrarily many layers. Each layer in turn is described by one line of a data-block that is assigned to the layer list. Each such line contains 6 numbers, either as numerical value, as a parameter, or as an arithmetic expression using none, one, or several parameters.

These 6 numbers are:

- the **cross section area of one thread** of the reinforcing cord ply (carcass or belt) [mm²]
- the **cord density**, given by the number of threads per length unit, in the direction perpendicular to the cord direction [1/mm]
- the **layer height** [mm]
- the **cord direction** (0 deg means circumferential direction, 90 deg means transversal direction) [deg]
- **Young's modulus of the cord material** [N/mm²]
- **Young's modulus of the matrix (rubber) material** [N/mm²].

All other data contained in a .tdd-file are described in detail below.

It is the user's decision how detailed the tire is described, in terms of the number of layer lists. *FTire/calc* will interpolate the information contained herein in an appropriate way, to calculate element stiffnesses of a 'single-layer' coarse Finite-Element model. The user can choose between three different complexity levels of this highly non-linear model, having **3990**, **6000**, or **8370** degrees of freedom respectively.

The low-level model places **13** nodes on each cross-section, the mid-level one **17**, whereas the high-level one uses **21**. Clearly, these numbers could be easily increased upon request, if longer computation times are acceptable. CPU-time will roughly grow only with the square of the number of degrees of freedom.

Actually, *FTire/calc* uses two different versions of the FE model:

- one for **modal calculations** (*FETire/modal*), based upon the determination of the generalized eigenvalues and eigenvectors of the linearized mass, stiffness, and damping matrix. In order to speed up calculation, this version takes full advantage of the tire's axisymmetry. This is achieved by placing the grid segments in an equally spaced manner around the tire circumference;
- another one for **non-linear statics calculations** (*FETire/static*). Here, for each load-case a non-linear system of equilibrium conditions is solved, using Newton-Raphson's method, combined with sparse linear system solvers. In order to optimize accuracy where it is needed, the mesh now is refined near the contact patch, see figure 3.

Modal and static calculations are performed for two different inflation pressures. For both pressure values, the statics is solved for a predefined list of 15 different cases ('load-cases'). These load-cases are defined as certain combinations of

- three different tire deflection values,
- three different camber angles,
- two different longitudinal displacement values,
- two different lateral tire displacement values,

- two different tire torsion angles about the vertical axis,
- and three different obstacle types.

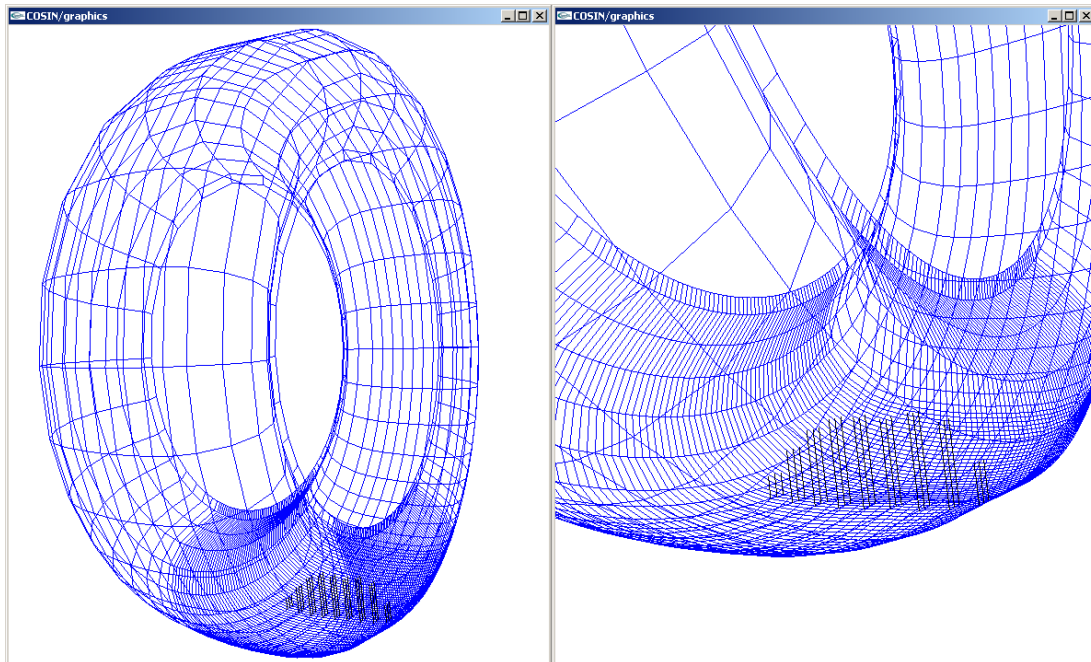


Figure 3: *FTire/static* load-case: tire deflection on transversal cleat with 6 deg camber angle

Each load-case either uses high or low surface friction coefficients.

The actual values of inflation pressure, tire deflection, rim displacement, rim rotation, and camber angle for these load-cases are predefined, but can be changed by the user. They are accessible through the entry fields of the 'control and analysis' column. Any change in these fields will be remembered by *FTire/calc* when being launched the next time.

However, for inexperienced users it is recommended not to modify the values. They are optimized to cover the most important operating conditions of a typical passenger car tire. In the next release, it will be possible to request optimized values for further classes of tires.

Note that the deflection values for the load cases are not necessarily the same values as those used in *FTire* to define the vertical stiffness characteristics. This is in contrast to the inflation pressure. These values are directly copied into the *FTire* data-file, and of course at the same time are used in the calculation of the pressure dependent stiffness data.

All the respective computation results (eigenfrequencies, spindle forces and moments, foot-print geometry etc.) are processed by *FTire/calc*, being sufficient to fill in a new *FTire* file completely.

There are several different 'philosophies' how a user can organize his own collection of .tdd-files. The most convenient mode of operation might be to consider a .tdd-file as nothing more but a **template**. Such a template describes the typical layer structure of a member of a tire family ('high-speed passenger-car tire', say, or 'heavy-duty truck tire'), but keeps all important and relevant numbers in the parameter block. This is what is done in the sample data file listed below.

The sample data file is parameterized for a 205/65 R 15 6.5 J summer tire, but the layer structure can be used for a wide variety of other passenger car tires as well.

To summarize, the rather complex data flow in *FTire/calc*, as described so far, is shown in figure 4.

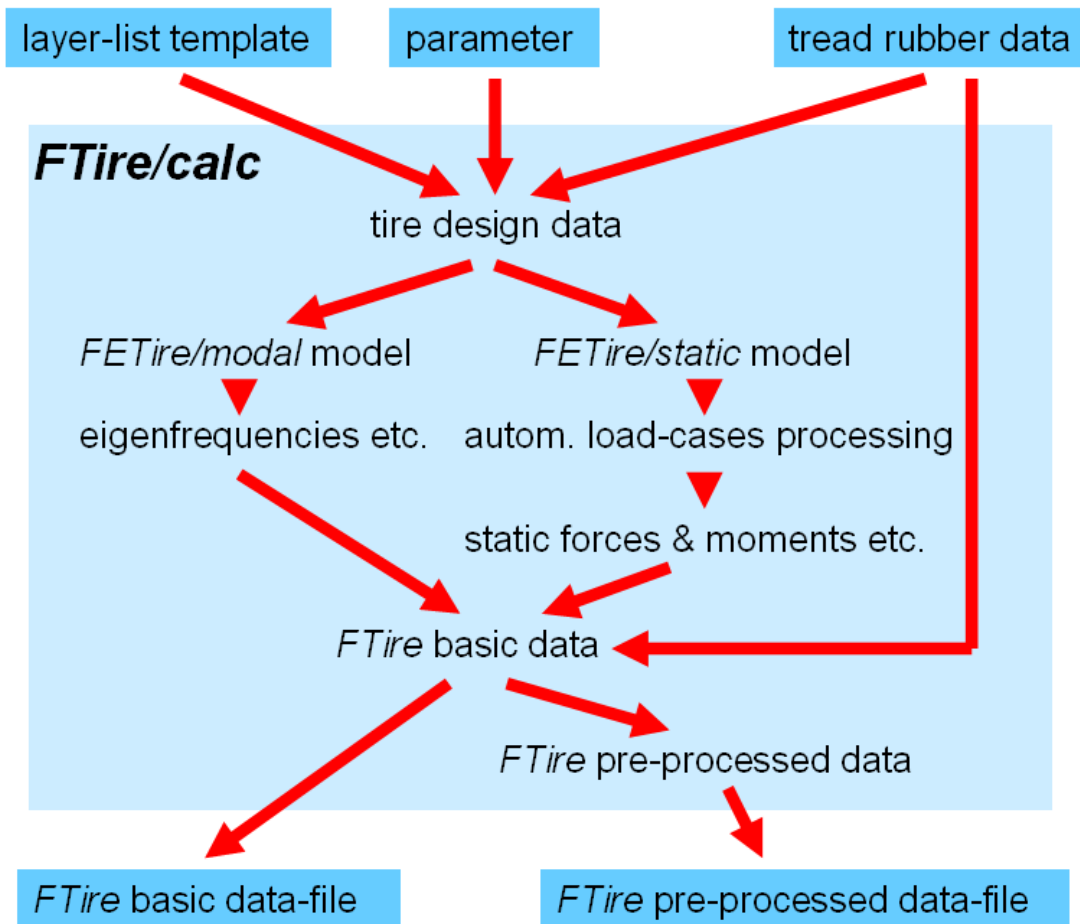


Figure 4: *FTire/calc* data flow

4.2 A Sample .tdd-File

The following is a listing of the sample tdd-file `passenger_car_tire.tdd`, which is shipped together with *FTire/calc*. The contents of the first three data-blocks, `$parameters`, `$more_parameters`, and `$auxiliary_parameters`, is completely customizable, and for that reason needn't be described here. All entries in data-block `$parameters` will appear in the parameterization column of the work-bench, whereas those in data-block `$more_parameters` are 'hidden' parameters. They are treated in the same way as the non-hidden ones, with the only exception of not being shown in the work-bench. Changing them can only be done by editing the .tdd-file. The optional data-block `$auxiliary_parameters` is meant to introduce certain arithmetic expressions and functions of parameters, to facilitate introduction of parameter dependencies.

Apart from these optional parameter data-blocks, the sample tdd-file, like any other .tdd-file, consists of

- a data-block called `$basic_data` with
 - the tire and rim sizes,
 - certain additional basic geometry properties,

- information needed to determine the tire's mass distribution,
 - the description of stiffness, damping, and friction properties of the tread rubber, and
 - a normalized look-up table of x/y-pairs to describe the cross-section geometry; and
- several data-blocks called \$layer_list_i (i = 1, 2, 3, ..), each containing one layer list, together with the respective value of the normalized co-ordinate s. The part of the tire to which the layer list belongs, is implicitly given by the respective name of the variable defining s. This name must be one out of
 - s_belt,
 - s_sidewall (if left and right side-walls are symmetric),
 - s_left_sidewall, or
 - s_right_sidewall.

```

*#echo
*****
* FTire/calc passenger car tire tdd-template
$parameters *****
width      205      ! tire section width [mm]
asp_ratio  65       ! tire aspect ratio [%]
rim_diam   15       ! rim diameter [in]
rim_width  6.0      ! rim width [in]
load_index 94       ! load index [-]
infl_press 2.0      ! nominal inflation pressure [bar]
mass       10.0     ! tire mass [kg]
dist_belt  21       ! distance belt border to side-wall [mm]
filler     60.0     ! bead filler rel. to side-wall height [%]
ang_belt   24.0     ! cord angle in steel belt [deg]
E_carc     1500     ! Young's modulus of carcass cord [N/mm^2]
dia_belt   0.5      ! cross-section area of belt cord thread [mm]
dia_carc   0.5      ! cross-section area of carcass [mm]
pre_strain 50.0     ! bandage pre-strain [N/mm]

$more_parameters *****
E_belt     210000    ! Young's modulus of steel cord [N/mm^2]
E_il       2.0       ! Young's modulus of inner-liner rubber [N/mm^2]
E_bf       15.3     ! Young's modulus of bead-filler rubber [N/mm^2]
E_bb       8.6      ! Young's modulus of bead-base rubber [N/mm^2]
E_cp       2.4      ! Young's modulus of carcass-ply rubber [N/mm^2]
E_bp       9.1      ! Young's modulus of belt-ply rubber [N/mm^2]
E_b        6.1      ! Young's modulus of bandage rubber [N/mm^2]
E_s        2.4      ! Young's modulus of side-wall rubber [N/mm^2]
E_tr       6.1      ! Young's modulus of tread rubber [N/mm^2]
h_rs       4.0      ! height rim strip [mm]
h_bf       10.0     ! height bead filler [mm]
h_il       0.3      ! height inner liner [mm]

```

```

h_base      2.0          ! height tread base [mm]
h_side      2.3          ! height side-wall outer layer [mm]
h_carc      1.0          ! height single carcass layer [mm]
h_band      1.0          ! height single bandage layer [mm]
h_belt      1.2          ! height single belt layer [mm]
ang_car1    85.0         ! cord angle in 1. carcass ply [deg]
ang_car2    95.0         ! cord angle in 2. carcass ply [deg]

```

\$auxiliary_parameters *****

```

A_carc      .25*pi*dia_carc^2      ! cross-sect. area carcass thread [mm^2]
A_belt      .25*pi*dia_belt^2      ! cross-sect. area belt cord thr. [mm^2]
d_carc      0.7/dia_carc           ! carcass cord density [threads/mm]
d_band      0.7/dia_carc           ! bandage cord density [threads/mm]
d_belt      0.7/dia_belt           ! belt cord density [threads/mm]
rbw         100-200*dist_belt/width ! belt width rel. to tire width [%]

```

\$basic_data *****

```

tire_section_width      width      ! [mm]
tire_aspect_ratio       asp_ratio  ! [%]
rim_diameter            rim_diam   ! [inch]
rim_width               rim_width  ! [inch]
load_index              load_index ! [LI]
inflation_pressure      infl_press ! [bar]

rel_belt_width          rbw        ! [%]
bandage_pre_strain      pre_strain ! [N/mm]

tire_mass               mass       ! [kg]
mass_percentage_bead    8.0       ! [%]
mass_percentage_left_sidewall 15.0    ! [%]
mass_percentage_right_sidewall 15.0    ! [%]
mass_percentage_belt    30.0     ! [%]
mass_percentage_tread   32.0     ! [%]

stiffness_tread_rubber  64       ! [Shore A]
damping_tread_rubber    0.0002  ! [s]
damping_matrix_rubber   0.005   ! [s]
tread_depth            8         ! [mm]
tread_positive         60       ! [%]

sliding_velocity        0.1      ! [m/s]
blocking_velocity       50.0     ! [m/s]
low_ground_pressure     0.1     ! [bar]
med_ground_pressure     2.0     ! [bar]
high_ground_pressure    10.0    ! [bar]

mu_adhesion_at_low_p    1.3     ! [-]
mu_sliding_at_low_p     1.1     ! [-]

```

```

mu_blocking_at_low_p          0.8  ! [-]
mu_adhesion_at_med_p         1.3  ! [-]
mu_sliding_at_med_p          1.0  ! [-]
mu_blocking_at_med_p         0.8  ! [-]
mu_adhesion_at_high_p        1.3  ! [-]
mu_sliding_at_high_p         1.0  ! [-]
mu_blocking_at_high_p        0.8  ! [-]

number_segments =             100  ! [-]
number_struct_nodes_per_seg = 13   ! [-]
number_tread_blocks_per_seg = 8    ! [-]
number_cont_points_per_block_x = 2  ! [-]
number_cont_points_per_block_y = 2  ! [-]
maximum_time_step =           .00005 ! [s]

```

```
carcass_contour! 205/65 R 15 *****
```

```

73.49    4.24
78.58    13.91
85.02    22.15
90.21    32.04
94.95    41.22
98.16    51.70
99.65    65.40
97.85    77.51
93.18    90.01
85.46   101.48
75.09   109.95
62.90   115.73
48.17   119.46
35.43   120.88
20.10   122.35
0.00    123.25

```

```
tread_contour !*****
```

```

85.46   115.00
75.09   120.00
62.90   127.00
48.17   130.00
35.43   131.00
20.10   131.00
0.00    131.00

```

```

*-----
* cr.sect. cord dens. height  alpha      E cord E matrix  type
* mm^2      thr/mm      mm      deg      N/mm^2 N/mm^2
*-----

```

```

$layer_list_1 ! left bead *****
s_left_sidewall = 0.0

```

```

0.0      0.0      h_il      0.0      0.0      E_il      ! inner liner
A_carc   d_carc   h_carc   ang_car1  E_carc   E_cp      ! 1. body ply
A_carc   d_carc   h_carc   ang_car2  E_carc   E_cp      ! 2. body ply
0.0      0.0      h_bf      0.0      0.0      E_bf      ! bead filler
A_carc   d_carc   h_carc   ang_car2  E_carc   E_cp      ! 2. body ply
A_carc   d_carc   h_carc   ang_car1  E_carc   E_cp      ! 1. body ply
0.0      0.0      h_rs      0.0      0.0      E_bb      ! rim strip

$layer_list_2 ! left filler center *****
s_left_sidewall = 0.005*filler
0.0      0.0      h_il      0.0      0.0      E_il      ! inner liner
A_carc   d_carc   h_carc   ang_car1  E_carc   E_cp      ! 1. body ply
A_carc   d_carc   h_carc   ang_car2  E_carc   E_cp      ! 2. body ply
0.0      0.0      .5*h_bf   0.0      0.0      E_bf      ! bead filler
A_carc   d_carc   h_carc   ang_car2  E_carc   E_cp      ! 1. body ply
A_carc   d_carc   h_carc   ang_car1  E_carc   E_cp      ! 2. body ply
0.0      0.0      h_rs      0.0      0.0      E_bb      ! rim strip

$layer_list_3 ! left side-wall *****
s_left_sidewall = .01*filler
0.0      0.0      h_il      0.0      0.0      E_il      ! inner liner
A_carc   d_carc   h_carc   ang_car1  E_carc   E_cp      ! 1. body ply
A_carc   d_carc   h_carc   ang_car2  E_carc   E_cp      ! 2. body ply
0.0      0.0      h_side    0.0      0.0      E_s       ! side

$layer_list_4 ! right bead *****
s_right_sidewall = 0.0
0.0      0.0      h_il      0.0      0.0      E_il      ! inner liner
A_carc   d_carc   h_carc   ang_car1  E_carc   E_cp      ! 1. body ply
A_carc   d_carc   h_carc   ang_car2  E_carc   E_cp      ! 2. body ply
0.0      0.0      h_bf      0.0      0.0      E_bf      ! bead filler
A_carc   d_carc   h_carc   ang_car2  E_carc   E_cp      ! 2. body ply
A_carc   d_carc   h_carc   ang_car1  E_carc   E_cp      ! 1. body ply
0.0      0.0      h_rs      0.0      0.0      E_bb      ! rim strip

$layer_list_5 ! right filler center *****
s_right_sidewall = 0.005*filler
0.0      0.0      h_il      0.0      0.0      E_il      ! inner liner
A_carc   d_carc   h_carc   ang_car1  E_carc   E_cp      ! 1. body ply
A_carc   d_carc   h_carc   ang_car2  E_carc   E_cp      ! 2. body ply
0.0      0.0      .5*h_bf   0.0      0.0      E_bf      ! bead filler
A_carc   d_carc   h_carc   ang_car2  E_carc   E_cp      ! 1. body ply
A_carc   d_carc   h_carc   ang_car1  E_carc   E_cp      ! 2. body ply
0.0      0.0      h_rs      0.0      0.0      E_bb      ! rim strip

$layer_list_6 ! right side-wall *****
s_right_sidewall = .01*filler
0.0      0.0      h_il      0.0      0.0      E_il      ! inner liner

```

```

A_carc    d_carc    h_carc    ang_car1   E_carc E_cp    ! 1. body ply
A_carc    d_carc    h_carc    ang_car2   E_carc E_cp    ! 2. body ply
0.0       0.0       h_side    0.0        0.0    E_s      ! side

$layer_list_7 ! belt boundary *****
s_belt = 0.0
0.0       0.0       h_il      0.0        0.0    E_il     ! inner liner
A_carc    d_carc    h_carc    ang_car1   E_carc E_cp    ! 1. body ply
A_carc    d_carc    h_carc    ang_car2   E_carc E_cp    ! 2. body ply
A_belt    d_belt    h_belt    ang_belt   E_belt E_bp    ! 1. belt ply
A_belt    d_belt    h_belt    -ang_belt  E_belt E_bp    ! 2. belt ply
A_carc    d_band    h_band    0.0        E_carc E_b     ! 1. bandage
A_carc    d_band    h_band    0.0        E_carc E_b     ! 2. bandage
0.0       0.0       h_base    0.0        0.0    E_tr     ! read base

$layer_list_8 ! belt zenith *****
s_belt = 0.5
0.0       0.0       h_il      0.0        0.0    E_il     ! inner liner
A_carc    d_carc    h_carc    ang_car1   E_carc E_cp    ! 1. body ply
A_carc    d_carc    h_carc    ang_car2   E_carc E_cp    ! 2. body ply
A_belt    d_belt    h_belt    ang_belt   E_belt E_bp    ! 1. belt ply
A_belt    d_belt    h_belt    -ang_belt  E_belt E_bp    ! 2. belt ply
A_carc    d_band    h_band    0.0        E_carc E_b     ! 1. bandage
A_carc    d_band    h_band    0.0        E_carc E_b     ! 2. bandage
0.0       0.0       h_base    0.0        0.0    E_tr     ! read base

$layer_list_9 ! belt boundary *****
s_belt = 1.0
0.0       0.0       h_il      0.0        0.0    E_il     ! inner liner
A_carc    d_carc    h_carc    ang_car1   E_carc E_cp    ! 1. body ply
A_carc    d_carc    h_carc    ang_car2   E_carc E_cp    ! 2. body ply
A_belt    d_belt    h_belt    ang_belt   E_belt E_bp    ! 1. belt ply
A_belt    d_belt    h_belt    -ang_belt  E_belt E_bp    ! 2. belt ply
A_carc    d_band    h_band    0.0        E_carc E_b     ! 1. bandage
A_carc    d_band    h_band    0.0        E_carc E_b     ! 2. bandage
0.0       0.0       h_base    0.0        0.0    E_tr     ! read base

```

4.3 Description of Tire Design Data in .tdd-File

Here is the comprehensive list of all data in a .tdd-file, together with their physical unit:

data block \$basic_data		
tire_section_width	mm	the maximum tire width, at inflated, but unloaded operating conditions. Typically, this value is the first number in the tire dimension string

tire_aspect_ratio	%	is the percentage of tire height to tire width. Typically, this value is the second number in the tire dimension string
rim_diameter	in	is the rim diameter. Typically, this value is the third number in the tire dimension string
rim_width	in	the rim width
load_index	LI	load index, a part of the tire's operational code. LI is the entry to a standardized table giving the maximum load per single wheel
rel_belt_width	%	width of widest steel belt layer, relative to tire's section width
tire_mass	kg	the tire's total mass
bandage_pre_strain	N/mm	pre-strain in bandage layer, in circumferential direction. Pre-strain is being measured as force per length unit in lateral direction
mass_percentage_bead	%	mass percentage, relative to total mass, of bead and one half of rim strip
mass_percentage_left_sidewall	%	mass percentage, relative to total mass, of all parts of the tire on the left-hand side that do not belong to bead, nor belt, nor tread region
mass_percentage_right_sidewall	%	mass percentage, relative to total mass, of all parts of the tire on the right-hand side that do not belong to bead, nor belt, nor tread region
mass_percentage_belt	%	mass percentage, relative to total mass, of all parts in the steel belt region, apart from tread base and tread cap
mass_percentage_tread	%	mass percentage, relative to total mass, of tread base and cap
stiffness_tread_rubber	Shore A	commonly used measure for the modulus of elasticity of the tread rubber. The following approximation is used to determine Young's modulus by means of the shore-A-stiffness S: $E = 10^{0.020477 \cdot S - 0.66095} \text{N/mm}^2$
damping_tread_rubber	s	quotient of tread rubber damping modulus and tread rubber elasticity modulus (Young's modulus)
tread_depth	mm	mean groove depth in tread
tread_positive	%	percentage of that share of the footprint that acutally has road contact, relative to the overall footprint area

sliding_velocity	m/s	first sliding velocity value used to define the sliding friction characteristics (for a more detailed description, see FTire_model documentation)
blocking_velocity	m/s	second sliding velocity value used to define the sliding friction characteristics
low_ground_pressure	bar	first (low) ground pressure value used to define the sliding friction characteristics
med_ground_pressure	bar	second (medium) ground pressure value used to define the sliding friction characteristics
high_ground_pressure	bar	third (high) ground pressure value used to define the sliding friction characteristics
mu_adhesion_at_low_p	-	sticking friction value of tread rubber on reference road surface at low ground pressure
mu_sliding_at_low_p	-	sliding friction value of tread rubber on reference road surface, at first sliding velocity value and low ground pressure
mu_blocking_at_low_p	-	sliding friction value of tread rubber on reference road surface, at second sliding velocity value and low ground pressure
mu_adhesion_at_med_p	-	sticking friction value of tread rubber on reference road surface at medium ground pressure
mu_sliding_at_med_p	-	sliding friction value of tread rubber on reference road surface, at first sliding velocity value and medium ground pressure
mu_blocking_at_med_p	-	sliding friction value of tread rubber on reference road surface, at second sliding velocity value and medium ground pressure
mu_adhesion_at_high_p	-	sticking friction value of tread rubber on reference road surface at high ground pressure
mu_sliding_at_high_p	-	sliding friction value of tread rubber on reference road surface, at first sliding velocity value and high ground pressure
mu_blocking_at_high_p	-	sliding friction value of tread rubber on reference road surface, at second sliding velocity value and high ground pressure

carcass_contour	mm	<p>arbitrarily many x/y data-points, following</p> <ul style="list-style-type: none"> • the carcass line in the belt and upper side-wall region, and • the bead filler center line in the lower side-wall and bead region. <p>Data must begin with the left bead center, and reach either to the belt center, or to the right bead center. If only one half of the carcass contour is defined, the geometry will be automatically mirrored.</p> <p>x-coordinate is the axial direction, y-coordinate the radial direction.</p> <p>If file format is 'cosin/io', carcass_contour is treated as a matrix with two columns and several rows. In this case, as usual, all values can be either defined by numerical data, or by arithmetic expressions.</p> <p><i>FTire/calc</i> will automatically mirror, shift, and stretch the data points to match exactly the given tire size.</p>
tread_contour	mm	<p>arbitrarily many x/y data-points, following the tread outer contour line.</p> <p>Data must begin with the left tread boundary, and reach either to the tread center, or to the right tread boundary. If only one half of the carcass contour is defined (cf. above), the geometry will be automatically mirrored.</p> <p>x-coordinate is the axial direction, y-coordinate the radial direction.</p> <p>If file format is 'cosin/io', tread_contour is treated as a matrix with two columns and several rows. In this case, as usual, all values can be either defined by numerical data, or by arithmetic expressions.</p> <p><i>FTire/calc</i> will automatically mirror, shift, and stretch the data points of tread_contour in the same way as carcass_contour, to match exactly the given tire size.</p>
data blocks \$layer_list_i (i = 1, 2, 3, ...)		

one of	s_belt s_sidewall s_left_sidewall s_right_sidewall	-	normalized curvilinear co-ordinate, defining location of tire structure part which is described by the layer list. The region of the tire where this part belongs to (left side-wall, right side-wall, or belt), is implicitly given by the name of the variable. The layer list is used for both side-walls, if the name is s_sidewall
remainder of data-block: arbitrarily many line with 6 values each		diverse	geometry and stiffness of a single layer, as discussed in 4.1: <ol style="list-style-type: none"> 1. the cross section area of one thread of the reinforcing cord ply (carcass or belt) [mm²] 2. the cord density, given by the number of threads per length unit, in the direction perpendicular to the cord direction [1/mm] 3. the layer height [mm] 4. the cord direction (0 deg means circumferential direction, 90 deg means transversal direction) [deg] 5. Young's modulus of the cord material [N/mm²] 6. Young's modulus of the matrix (rubber) material [N/mm²].

5 Additional Tools

FTire/calc provides two additional easy-to-use tools to analyze the properties of the tire defined by the .tdd-file. These tools are not running through the complete list of all modal and static load-case calculations. This is way they reach much shorter response times.

Both tools are accessible by a button in the 'control and analysis' column. Any change in a load-case definition entry field also affects these tools.

In contrast to the calculation of an *FTire* data file, the tools provide interactive graphical output to better visualize the computation results.

5.1 Analysis of Static Load-Cases

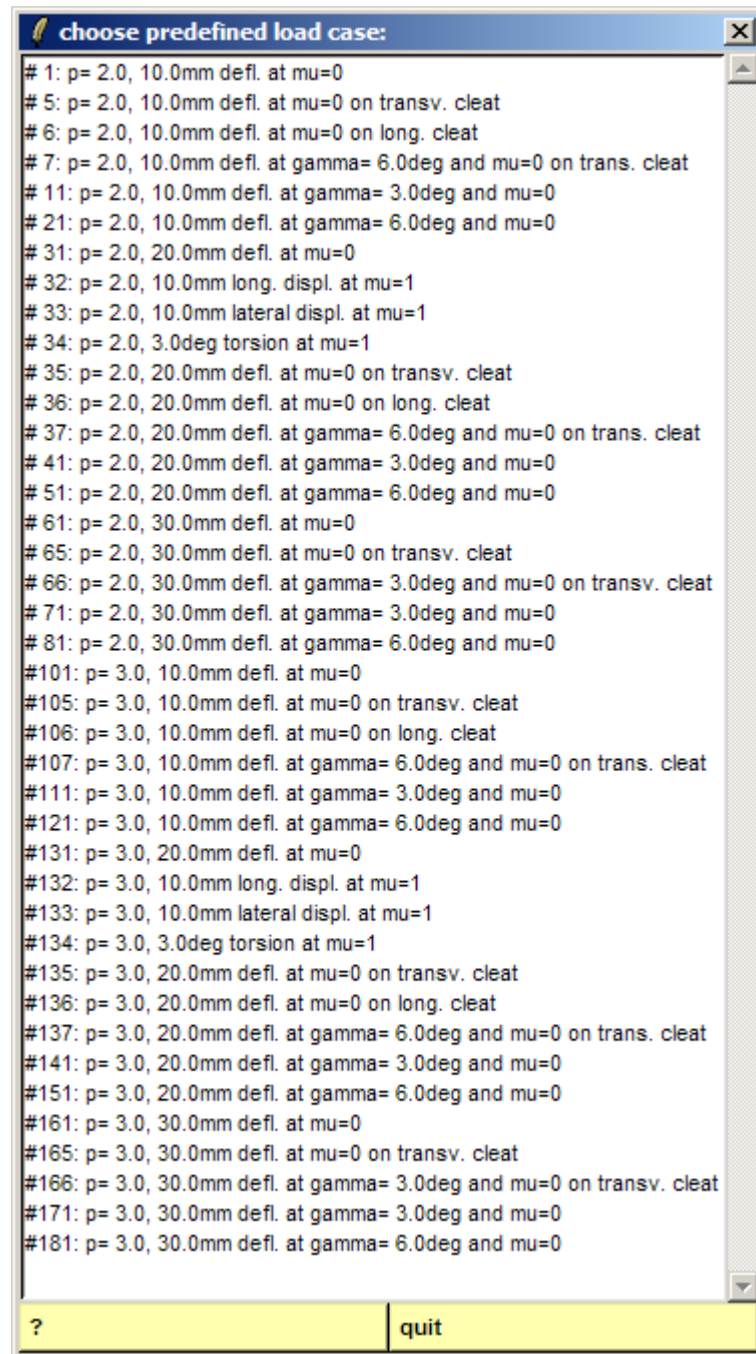


Figure 5: load-case selection window

Clicking the 'calculate single load case..' button in the 'control and analysis' column will display a window like shown in figure 5. This window lists all available predefined load-cases. Clicking an item in the list will launch the calculation of the respective load-case. The list will reflect the actual values of the load-case data, as chosen in the entry fields above.

After completion of the calculation, a graphics window will open, showing the distorted grid (blue lines) and the vector-valued contact forces (black lines).

The graphics can be transformed, zoomed, and controlled in a variety of ways. To learn more on how

to interactively control the graphics output, press the F1 key while the graphics window is active. This on-line help describes the controls that are available in **each** *cosin* application using this kind of graphics. However, there are some more, *FTire/calc*-specific interactive key-stroke controls, which are also listed in the graphics menu:

key g	toggles wire-frame representation of the FE mesh
key o	toggles display of the obstacle (if surface is not flat)
key f	toggles display of vector-valued nodal forces
key c	toggles display of contact forces
key p	toggles plot of contact pressure distribution, instead of tire structure
key b	toggles b/w and colored display (only effective for contact pressure plot)
key r	toggles additional display of un-distorted tire structure as reference (only effective for mesh plot)
key e	closes graphics window (like Esc key).

5.2 Modal Analysis

The second tool, launched by clicking the ‘**animate single mode..**’ button in the ‘control and analysis’ column, allows to animate a mode shape of the linearized *FETire/modal* structure.

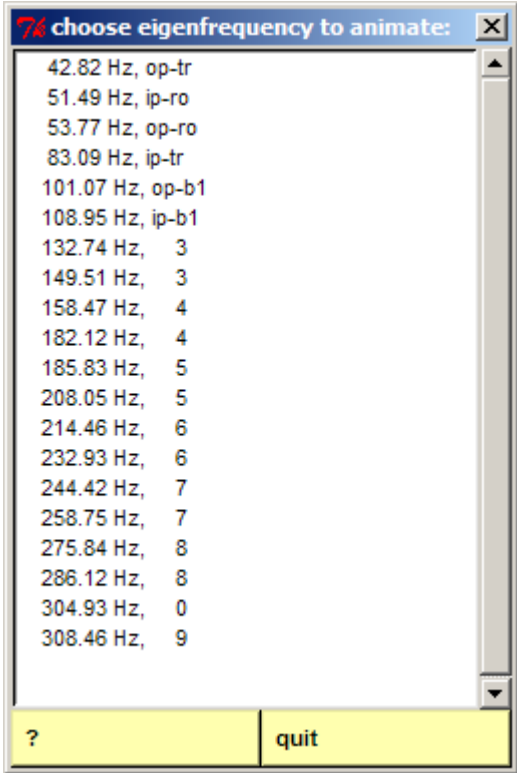


Figure 6: mode selection window

In order to enable the user to choose the mode to be animated, a list of the first 20 eigenfrequencies is calculated and displayed. Generation of the list might take some seconds, so please be patient.

The list will display the eigenfrequencies in ascending order, together with either a short string, or the mode order. The string characterizes the recognized mode type:

- **op-tr** out-of-plane translatoric 'rigid-body' mode
- **ip-rp** in-plane rotatoric 'rigid-body' mode
- **op-ro** out-of-plane rotatoric 'rigid-body' mode
- **ip-tr** in-plane translatoric 'rigid-body' mode
- **op-b1** first out-of-plane bending mode
- **ip-b1** first in-plane bending mode.

Clicking a frequency in the list will open a graphics window (again, this might take some seconds), and animate the respective mode. Animation is aborted by hitting the **Esc** or **e** key. As with the graphics of the load-case calculation, hitting space-bar will display the graphics menu.