

cosin/road

Cosin Road Modeling
Documentation and User's Guide

Contents

1	Introduction	1
2	Road Profiles and Obstacles in <i>cosin/io</i> Format	2
2.1	Data Common to all <i>cosin/io</i> -style Road Types	2
2.2	<i>cosin/io</i> 2D Road Type drum	5
2.3	<i>cosin/io</i> 2D Road Type flatbelt	7
2.4	<i>cosin/io</i> 2D Road Type file	8
2.5	<i>cosin/io</i> 2D Road Type function	9
2.6	<i>cosin/io</i> 2D Road Type hydropulse_harmonic	10
2.7	<i>cosin/io</i> 2D Road Type hydropulse_noise	10
2.8	<i>cosin/io</i> 2D Road Type hydropulse_function	11
2.9	<i>cosin/io</i> 2D Road Type plank or cleat	11
2.10	<i>cosin/io</i> 2D Road Type pot_hole	11
2.11	<i>cosin/io</i> 2D Road Type ramp	12
2.12	<i>cosin/io</i> 2D Road Type roof	12
2.13	<i>cosin/io</i> 2D Road Type sine	12
2.14	<i>cosin/io</i> 2D Road Type sine_sweep	12
2.15	<i>cosin/io</i> 2D Road Type spline	14
2.16	<i>cosin/io</i> 2D Road Type stochastic_uneven	14
2.17	<i>cosin/io</i> 2D Road Type tilt_table	16
3	Road Profiles and Obstacles in TeimOrbit™ Format	16
3.1	TeimOrbit 2D Road Type drum	16
3.2	TeimOrbit 2D Road Type flatbelt	20
3.3	TeimOrbit 2D Road Type hydraulic_test_rig	21
3.4	TeimOrbit 2D Road Type plank or cleat	23

3.5	TeimOrbit 2D Road Type pot_hole	23
3.6	TeimOrbit 2D Road Type ramp	24
3.7	TeimOrbit 2D Road Type roof	24
3.8	TeimOrbit 2D Road Type sine	24
3.9	TeimOrbit 2D Road Type sine_sweep	24
3.10	TeimOrbit 2D Road Type stochastic_uneven	25
3.11	TeimOrbit 2D Road Type tilt_table	27
3.12	Superposition of TeimOrbit 2D Roads with 3D Roads	28
4	Regular Grid Road Data Files (RGR Files)	28
4.1	Regular Grid Data Items	28
4.2	Regular Grid Data File Format	29
4.3	Combination with Curved Center-Line Data Files	30
4.4	Combination with Soft-Soil or Snow Models	33
5	cosin Road Track Files	35
6	User-Defined Road Models	35
	Index	38

Preface

This documentation describes all routines of the *cosin* simulation software which provide attributes of the road or terrain surface, being used in tire or vehicle simulations. These routines are subsumed under the product name *cosin/road*. *cosin/road* is used, for example, by the [CTI tire interface](#) to 3rd party simulation software, by [cosin/mbs](#), and by [FTire/sim](#).

Some data files use the *cosin/io* syntax, which is described in the separate documentation chapter [cosin/io User's Guide](#).

All product or brand names in this documentation are trademarks or registered trademarks of their respective holders.

1 Introduction

To specify and evaluate road excitation, *cosin/road* uses nothing but one central subroutine `evrnr` (*cosin* environmental models: *road main* routine).

This routine, for example being called by the *CTI tire interface* to 3rd party simulation software, by *cosin/mbs*, and by *FTire/sim*, receives, during initialization, the name of the **road data** file by the calling program. On basis of the file name and the file contents, `evrnr` determines the data file format. Accordingly, it branches into one of several evaluation routines. Among these are routines for

- *cosin/io-style* 2D and 3D deterministic, stochastic, or measured obstacles/profiles,
- TeimOrbit-style 2D and 3D deterministic, stochastic, or measured obstacles/profiles,
- RGR regular grid road models,
- *cosin* road track files,
- user-defined road models,
- 3rd-party generally available road models, like the CRG format,

and more.

In the following chapters, the most important ones of these road models are documented in detail.

Please note that not all of the road models might be available in all simulation environments which use *cosin/road*. On the other hand, *cosin/road* might be able to call additional, proprietary road models of the calling software.

Please refer to the respective original documentation to learn more about the availability of road models in your simulation environment.

2 Road Profiles and Obstacles in *cosin/io* Format

If *cosin/road* detects the road file being in *cosin/io* format, it first tries to read certain information common to all road types within this format. Next, depending on the respective road or obstacle type, it will search for more data, specific to this type.

2.1 Data Common to all *cosin/io*-style Road Types

If the *cosin/io* format has been recognized, *evrnr* searches data block `$road_type` to read the above mentioned data. These data are used in road profile calculation for all types and in all subsequent invocations:

Name of input variable	Unit	Meaning
type	-	a character string out of the following list: <ul style="list-style-type: none">• cleat• drum• file• function• hydropulse_harmonic• hydropulse_noise• plank• pot_hole• ramp• roof• sine• sine_sweep• spline• stochastic_uneven• tilt_table to define the type of the obstacle or irregularity. Normally, to every type additional parameters are looked for, as explained in chapter 1 and in the following

Name of input variable	Unit	Meaning
offset	mm	value to be added after calculation of reference road height (that is, a shift of the road in vertical direction). Parameter is optional
start_at	m	to minimize irrelevant tire accelerations when momentarily applying a tire deflection, start of the road profile as defined by the contents of data block \$road_type can be delayed by use of a 'ramp' at the beginning of the simulation With parameter start_at, the end of this ramp and start of the regular road is defined in terms of travel distance. This delay does not apply to road types drum and hydropulse . Additionally, several road types have a parameter start that marks the beginning of a single obstacle relative to travel distance 0. Parameter is optional
y_min	m	road height will be zero, if $y < y_{min}$. Parameter is optional
y_max	m	road height will be zero, if $y > y_{min}$. Parameter is optional
t_period	s	road height will be periodic in time, with period interval specified by parameter (optional)
x_period	m	road height will be periodic in x-coordinate, with period length specified by parameter (optional)
y_period	m	road height will be periodic in y-coordinate, with period length specified by parameter (optional)

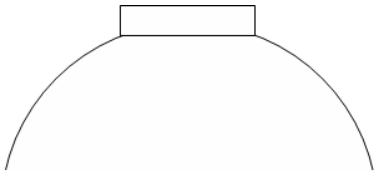
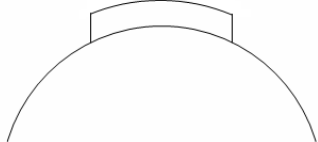
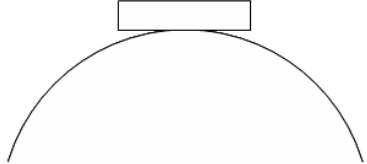
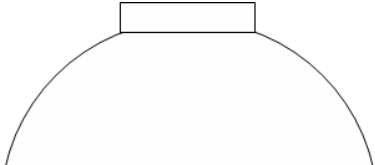
Name of input variable	Unit	Meaning
mu_factor	-	friction value factor μ to be used in tire models to modulate friction/stiction coefficients between tread rubber and road surface. If mu_factor is defined by an arithmetic expression, it may contain x, y, and t as independent variables: $\mu = f(x, y, t, p)$. By this, distributed and/or time-dependent road friction can easily be modeled. The function f may be an arbitrary arithmetic expression as documented in cosin/io , including 1D and 2D lookup tables, random values, etc.

Depending on the type of road chosen, more data or data blocks are looked for and read, according to the following tables:

2.2 *cosin/io* 2D Road Type **drum**

cosin/road looks for data block \$drum, and reads:

Name of input variable	Unit	Meaning
diameter	m	diameter of the tire test drum. <i>cosin/road</i> assumes an outer drum, if diameter < 0
v	m/s	rotation speed of drum surface (be sure to choose rolling_speed = 0 in data block \$sources. Otherwise, wheel will move away from drum)
acceleration_time	s	optional time span at beginning of simulation during that drum is accelerated to nominal speed
number_cleats	-	number of extra cleats on drum (number_cleats = 0 allowed)
cleat_height	mm	height of extra cleats
cleat_starting_angle	deg	drum angle coordinate of 1 st cleat
cleat_length	mm	length of cleat, measured in circumferential direction of drum
cleat_bevel_edge_length	mm	length of bevel edge of cleat, measured in circumferential direction of drum. Bevel edge has 45deg slope
cleat_direction	deg	direction of cleat relative to drum spin axis. Direction is zero if cleat is oriented exactly in transversal direction (which is the default case). Direction angle is measured counter-clockwise if looking from above onto the drum surface. Default value is 0 (transversal cleat)
mu_factor_cleat	-	friction modification factor on cleats

Name of input variable	Unit	Meaning
curved_cleat_surface	-	<p>0: cleat's surface is flat</p>  <p>1: cleat's surface is curved according to drum surface curvature.</p>  <p>Default value is 0 (flat)</p>
flat_cleat_support	-	<p>0: cleat is mounted on the drum such that its support on the drum surface is curved according to overall drum curvature</p>  <p>1: cleat is mounted on the drum such that its support on the drum surface is flat between cleat's start and cleat's end</p>  <p>Default value is 0 (non-flat cleat support). This parameter is only used and relevant if curved_cleat_surface = 0</p>
stopper_distance	mm	distance between front and rear stopper of drum opening. Default value is 75 % of drum diameter
stopper_geometry	string	name of an optional spline data-block, containing x/z data pairs, both in [mm]. The data pairs describe the geometry of the stoppers. Data are mirrored for the rear stopper. If specified, it is recommended to check the actual appearance of drum surface and stoppers, using <i>FTire/roadtools</i>

Name of input variable	Unit	Meaning
<code>non_uniformity_spline</code>	string	name of an optional spline data-block, containing ϕ/r data pairs. They describe the angle-dependent radial non-uniformity of the drum surface. ϕ (the circumferential coordinate) is to be specified in [deg]. r (the radius variation, superimposed to the nominal drum radius) is to be specified in [mm]
<code>cleat_lat_shape_fact_spline</code>	string	name of a spline data-block, containing y/f data pairs (y in [mm], f dimensionless). The optional data pairs describe a varying lateral geometry of the cleat(s). $f(y)$ is a factor by which the nominal cleat height is multiplied
<code>drum_lat_shape_spline</code>	string	name of a spline data-block, containing y/z data pairs (both in [mm]). The optional data pairs describe a varying lateral geometry of the drum surface. $z(y)$ is superimposed to the nominal drum radius

If one or more of the four optional spline data-blocks, mentioned at the end of the above table, is specified, *cosin/road* looks for spline data in respectively named sub-data blocks. These data blocks must contain two data columns each, for the respective independent and dependent variable. For the syntax of spline data definitions, cf. *cosin/io* documentation.

2.3 *cosin/io* 2D Road Type **flatbelt**

cosin/road looks for data block `$flatbelt`, and reads:

Name of input variable	Unit	Meaning
v	m/s	rotation speed of belt surface (be sure to choose <code>rolling_speed = 0</code> in data block <code>\$sources</code> . Otherwise, wheel will move away from belt)
number_cleats	-	number of extra cleats on belt (<code>number_cleats = 0</code> allowed)
cleat_height	mm	height of extra cleats
cleat_length	mm	length of cleat, measured in circumferential direction of drum
cleat_bevel_edge_length	mm	length of bevel edge of cleat, measured in circumferential direction of drum. Bevel edge has 45deg slope
mu_factor_cleat	-	friction modification factor on cleats
acceleration_time	s	optional time span at beginning of simulation during that belt is accelerated to nominal speed

2.4 *cosin/io* 2D Road Type **file**

cosin/road looks for data block `$file`, and reads:

Name of input variable	Unit	Meaning
file_name	-	name of a file to read measured or calculated data from. The file must be in standard or Matlab format (cf. cosin/io documentation). <i>cosin/road</i> expects equidistant road profile height data as columns of the matrix stored in the file, and optionally banking angles of left and right track. Unit is [m] and [deg], resp.
channel_left_track	-	column index of left track data
channel_right_track	-	column index of right track data. May coincide with <code>channel_left_track</code> , if only one track has being measured and/or stored
channel_left_banking_angle	-	column index of banking angle of left track (given in deg). If this index is not specified, banking angle of left track is set to 0
channel_right_banking_angle	-	column index of banking angle of right track (given in deg). May coincide with <code>channel_left_banking_angle</code> , if only one track has being measured and/or stored. If this index is not specified, banking angle of right track is set to 0
meas_distance	mm	distance between two consecutive measured points in longitudinal direction. Travel distances outside the range of data in the file get a road height by extrapolation
track_linked_to_course	0/1	flag whether track (or road) has to be 'linked' to the actual vehicle course (that is, whether the vehicle travel path is used to determine the look-up distance in the road profile). If vehicle will stop on the road, due to numerical reasons <code>track_linked_to_course = 0</code> is recommended

2.5 *cosin/io* 2D Road Type function

In data block `$road_type`, *cosin/road* reads:

Name of input variable	Unit	Meaning
z	mm	road height, as function of x, y, t and optional further <i>cosin/io</i> parameters: $z = f(x, y, t, p)$. That function may be an arbitrary arithmetic expression as documented in <i>cosin/io</i> , including 1D and 2D lookup tables, random values, etc.

2.6 *cosin/io* 2D Road Type **hydropulse_harmonic**

cosin/road looks for data block \$hydropulse_harmonic, and reads:

Name of input variable	Unit	Meaning
frequency_fl frequency_fr frequency_rl frequency_rr	Hz	hydro-pulse excitation frequencies at front left / front right / rear left / rear right wheel
amplitude_fl amplitude_fr amplitude_rl amplitude_rr	mm	hydro-pulse excitation amplitudes at front left / front right / rear left / rear right wheel. Set amplitude to zero, if the respective wheel is not to be excited
phase_fl phase_fr phase_rl phase_rr	deg	hydro-pulse phase angles at front left / front right / rear left / rear right wheel

2.7 *cosin/io* 2D Road Type **hydropulse_noise**

This road type, generating individual stochastic hydro-pulse signals for up to 4 wheels, works similar as road type stochastic_uneven, cf. below. *cosin/road* looks for data block \$hydropulse_noise, and reads:

Name of input variable	Unit	Meaning
intensity_fl intensity_fr intensity_rl intensity_rr	-	factor to control intensity of 'white velocity noise' (which approximates measured spectra of road profiles pretty good), at front left / front right / rear left / rear right wheel
time_constant_fl time_constant_fr time_constant_rl time_constant_rr	s	time-constant of high-pass integration filter, at front left / front right / rear left / rear right wheel

2.8 *cosin/io* 2D Road Type **hydropulse_function**

This road type allows to compute wheel-individual road height functions, using general arithmetic expressions, depending on x, y, and t. These arithmetic expressions can make use of all the special functions made available in *cosin/io*, including random number generators, all kinds of splines, data read from files, etc.. *cosin/road* looks for data block \$hydropulse_function, and reads:

Name of input variable	Unit	Meaning
z_fl z_fr z_rl z_rr	-	arithmetic expressions used to compute the road heights at front left / front right / rear left / rear right wheel. Additionally to the special constants available in all <i>cosin/io</i> -style arithmetic expressions, they may make arbitrary use of the independent variables x, y, and t

2.9 *cosin/io* 2D Road Type **plank** or **cleat**

cosin/road looks for data block \$plank or \$cleat, and reads:

Name of input variable	Unit	Meaning
height	mm	height of plank/cleat
start	m	start of plank/cleat (travel distance)
length	mm	length of plank/cleat, measured along x-axis
bevel_edge_length	mm	length of bevel edge, measured along x-axis. Bevel edge has 45deg slope. With edge_rounded = 1, rounded corners instead of bevel edges are used. In this case, bevel_edge_length is radius of the corner
edge_rounded	0/1	1: edge rounded, 0: edge linear, not rounded
direction	deg	direction of plank/cleat relative to y-axis. direction = 0 if plank/cleat is placed crosswise
mu_factor_cleat	-	friction value factor on plank/cleat

2.10 *cosin/io* 2D Road Type **pot_hole**

cosin/road looks for data block \$pothole, and reads:

Name of input variable	Unit	Meaning
depth	mm	depth of pot-hole
start	m	start of pot-hole (travel distance)
length	mm	length of pot-hole

2.11 *cosin/io* 2D Road Type **ramp**

cosin/road looks for data block \$ramp, and reads:

Name of input variable	Unit	Meaning
heigth	mm	height of ramp
start	m	start of ramp (travel distance)
slope	-	slope of ramp; 1 means 45deg

2.12 *cosin/io* 2D Road Type **roof**

cosin/road looks for data block \$roof, and reads:

Name of input variable	Unit	Meaning
heigth	mm	height of 'roof' (= triangle-shaped obstacle)
start	m	start of 'roof' (travel distance)
length	mm	length of 'roof', measured along x-axis

2.13 *cosin/io* 2D Road Type **sine**

cosin/road looks for data block \$sine, and reads:

Name of input variable	Unit	Meaning
amplitude	mm	amplitude of sine wave
wave_length	m	wave length of sine wave
start	m	start of sine wave (travel distance)
length	mm	length of wave section (travel distance, optional)

2.14 *cosin/io* 2D Road Type **sine_sweep**

A sine-shaped road profile with slowly varying frequency and amplitude is calculated in two different ways:

linear sweep: the frequency increases linearly with respect to travel distance. The road height value $z(s)$ as function of travel distance s is calculated as follows:

$$z(s) = \left(a_s + \frac{a_e - a_s}{s_e - s_s} (s - s_s) \right) \cdot \sin \left(2\pi \cdot \left(f_s + \frac{f_e - f_s}{2(s_e - s_s)} (s - s_s) \right) \cdot (s - s_s) \right)$$

(note the factor '2' in denominator, which is not an error!). The actual frequency (= derivative of the sine function argument with respect to travel path, divided by 2π ; this is not equal to that factor that is multiplied by $2\pi(s - s_s)$ in the sine function!) is given by

$$f(s) = f_s + \frac{f_e - f_s}{s_e - s_s} (s - s_s)$$

logarithmic sweep: with every cycle, the wave length decreases by a constant factor. The road height value is calculated as follows:

$$z(s) = \left(a_s + \frac{a_e - a_s}{s_e - s_s} (s - s_s) \right) \cdot \sin \left(2\pi f_s s_\infty \ln \frac{s_\infty}{s_\infty + s_s - s} \right)$$

where

$$s_\infty = \frac{f_e}{f_e - f_s} (s_e - s_s)$$

s_∞ is that travel path where theoretically an infinitely high frequency was reached, measured relative to sweep start s_s . The actual frequency is given by

$$f(s) = \frac{s_\infty}{s_\infty + s_s - s} f_s$$

For data supply, *cosin/road* looks for data block \$sine_sweep, and reads:

Name of input variable	Unit	Meaning
start	m	start of swept sine wave (travel distance)
end	m	end of swept sine wave (travel distance)
amplitude_at_start	mm	amplitude of swept sine wave at start
apmlitude_at_end	mm	amplitude of swept sine wave at end
wave_length_at_start	m	wave length of swept sine wave at start
wave_length_at_end	m	wave length of swept sine wave at end. Must be less or equal wave_length_at_start
sweep_type	0/1	0: frequency changes linearly with respect to travel distance 1: wave length changes each cycle by a constant factor

2.15 *cosin/io* 2D Road Type **spline**

In data block \$road_type, *cosin/road* reads:

Name of input variable	Unit	Meaning
periodic	0/1	0 (default): spline data are to be evaluated in a non periodic manner 1: spline data are to be evaluated periodically, as 'repeating sequence'

After having read data block \$road_type, *cosin/road* looks for the spline data in sub-data block \$spline_data. This data block contains two data columns: the first is travel distance in m, the second road height in mm. For the syntax of spline data definitions, cf. [cosin/iocosin/io](#) documentation.

2.16 *cosin/io* 2D Road Type **stochastic_uneven**

A stochastic uneven road profile both for left and right wheels is generated, that has properties very close to measured road profiles.

To this end, in a first step discrete white noise signals are formed on the basis of nearly **uniformly** distributed random numbers, two of these assigned to every 10 mm of travel path. The distribution of these random numbers is approximated by summing up several **equally** distributed random numbers, taking advantage of the 'law of large numbers' of mathematical statistics.

Next, these values are integrated with respect to travel distance, using a simple first order timediscrete integration filter. The reason for not just using a pure integrator is to cut off extremely low frequencies, which would result in large road elevation values, not at all affecting vehicle dynamics.

The independent variable of that filter is not time, but travel path. That is why the filter cut-off frequency Ω is controlled by a '**path** constant' $S = \Omega_c^{-1}$ instead of a **time** constant. Approximate power cc spectral density (PSD) of this road surface profile is given by

$$G_d(\Omega) = \frac{K^2}{\Omega^2 + S^2} \approx G_d(\Omega_0) \cdot \left(\frac{\Omega_0}{\Omega}\right)^2$$

where

$$S \gg 1 m$$

$$G_d(\Omega_0) = \left(\frac{K}{\Omega_0}\right)^2$$

The filter process results in two realizations of this approximate '**white** velocity noise'; that means, two signals, the derivatives of which are close to **white noise**. Signals with that property are known as road

profiles with 'waviness' $w = 2$ (cf. ISO 8608). Several investigations show that the waviness derived from measured road displacement PSDs ranges from about 1.8 to 2.2.

The reference spectral density value $G_d(\Omega_0)$, with $\Omega_0 = 1 \text{ rad/m}$, is used in ISO 8608 to classify road surface profiles, according to the following table:

Road class	$G_d(\Omega_0) [10^{-6}m^3]$		
	min. value	mean value	max. value
A	-	1	2
B	2	4	8
C	8	16	32
D	32	64	128
E	128	256	512
F	512	1024	2048
G	2048	4096	8192
H	8192	16384	-

The last step in the generation of the stochastic uneven road profiles is to linearly combine the two realizations $z_1(s)$, $z_2(s)$ of the above mentioned process, resulting in the left and right profile $z_l(s)$, $z_r(s)$. This is done such that these two signals are completely independent, if `correlation_rl = 0.0`, and identical, if `correlation_rl = 1.0`:

$$z_l(s) = z_1(s) + \frac{\text{corr}_{rl}}{2} (z_2(s) - z_1(s))$$

$$z_r(s) = z_2(s) + \frac{\text{corr}_{rl}}{2} (z_2(s) - z_1(s))$$

For data supply, `cosin/road` looks for data block `$stochastic_uneven`, and reads:

Name of input variable		Unit	Meaning
one of	<code>reference_spectral_density</code>	m^3	$G_d(\Omega_0)$
	<code>ISO_8608_road_class</code>	A, B, ...	ISO 8608 road class, according to table above. Mean $G_d(\Omega_0)$ value of the respective class will be used
<code>path_constant</code>		m	'path constant' S to control high-pass integration filter cut-off frequency in path domain. Parameter is optional, default value is 1000 m
<code>correlation_rl</code>		-	variable to control correlation between left and right track: no correlation, if zero; complete correlation (that is, left track = right track), if one. Any value between 0 and 1 is allowed

2.17 *cosin/io* 2D Road Type **tilt_table**

cosin/road looks for data block \$tilt_table, and reads:

Name of input variable	Unit	Meaning
y_coord_rot_axis	mm	y coordinate of tilt-table's rotation axis (which is assumed to be parallel to global x-axis)
start_time	s	start time of tilt table's rotation
angular_velocity	deg/s	tilt table's angular velocity about x-axis; assumed to be constant during operation

3 Road Profiles and Obstacles in TeimOrbitTM Format

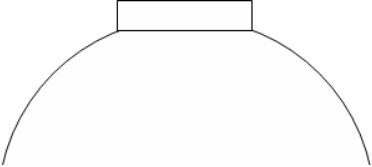
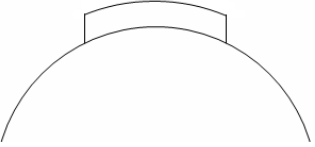
If *cosin/road* has recognized that the road data file is in [TeimOrbit format](#) (typically, the file name extension is 'rdf'), it will branch into the respective reading and evaluation routines.

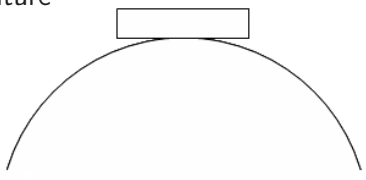
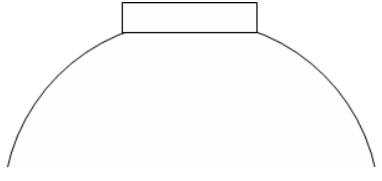
Below, all available road types in this format are described. Please refer to the original documentation for TeimOrbitTM syntax rules, as well as for meaning and contents of all other data sections not documented here.

3.1 TeimOrbit 2D Road Type **drum**

The rotating drum is implemented as a 2D type road. If ROAD_TYPE is set to drum, *cosin/road* searches for and reads the following data items in section [PARAMETERS]:

Name of input variable	Unit	Meaning
diameter	length	diameter of the tire test drum. <i>cosin/road</i> assumes outer drum, if diameter < 0
x_coord_drum_center	length	x coordinate of drum center in global coordinates, used for all wheels. Remark: make sure to place the drum near the rim center, and to keep the wheel's translational speed small enough not to leave the drum's zenith
y_coord_drum_center	length	y coordinate of drum center in global coordinates, used for all wheels
x_coord_drum_center_wheel_i <i>i</i> = 1, 2, 3, ..., 25	length	x coordinate of drum center in global coordinates, used only for wheel <i>i</i> <i>i</i> = 1, 2, 3, ..., 25

Name of input variable	Unit	Meaning
y_coord_drum_center_wheel_i <i>i</i> = 1, 2, 3, ..., 25	length	y coordinate of drum center in global coordinates, used only for wheel <i>i</i> <i>i</i> = 1, 2, 3, ..., 25
number_cleats	-	number of extra cleats on drum (number_cleats = 0 allowed)
cleat_starting_angle	angle	drum angle coordinate of 1 st cleat, used for all wheels
cleat_steering_angle_wheel_i <i>i</i> = 1, 2, 3, ..., 25	angle	drum angle coordinate of 1 st cleat, used only for wheel <i>i</i> <i>i</i> = 1, 2, 3, ..., 25
cleat_direction	angle	direction of cleat relative to drum spin axis. Direction is zero if cleat is oriented exactly in transversal direction (which is the default case). Direction angle is measured counter-clockwise if looking from above onto the drum surface. Default value is 0 (transversal cleat)
curved_cleat_surface	-	<p>0: cleat's surface is flat</p>  <p>1: cleat's surface is curved according to drum surface curvature.</p>  <p>Default value is 0 (flat)</p>

Name of input variable	Unit	Meaning
flat_cleat_support	-	<p>0: cleat is mounted on the drum such that its support on the drum surface is curved according to overall drum curvature</p>  <p>1: cleat is mounted on the drum such that its support on the drum surface is flat between cleat's start and cleat's end</p>  <p>Default value is 0 (non-flat cleat support). This parameter is only used and relevant if curved_cleat_surface = 0</p>
mu_factor_cleat	-	friction modification factor on cleats
stopper_distance	length	distance between front and read stopper of drum opening. Default value is 75 % of drum diameter

The drum speed is defined by one of the following alternatives:

Either:

v	length/time	constant rotation speed of drum surface
acceleration_time	time	time span at beginning of dynamic simulation, to accelerate drum from stand-still to nominal velocity Default value is 0

or:

data sub-section (TV_DATA)	arbitrarily many lines with 2 values of type time, length/time	data pairs defining speed-vs.-time data of the drum's time-dependent surface speed profile. t-values (first value in each line) should be in ascending order
-------------------------------	---	--

The cleat's exact geometry is defined by one of the three following alternatives:

Either:

cleat_height	length	height of cleats
cleat_length	length	length of cleat, measured in circumferential direction of drum
cleat_bevel_edge_length	length	length of bevel edge of cleat, measured in circumferential direction of drum. Bevel edge has 45deg slope
cleat_edge_rounded	-	0: cleat bevel edge flat 1: cleat bevel edge rounded by a quarter circle

or:

data sub-section (XZ_DATA)	arbitrarily many lines with data of type 2 x length	data pairs defining x/z data of the cleat's cross section. No rounding of edges is assumed. x-values should be in ascending order. First and last y-value should be zero
-------------------------------	---	--

or:

data sub-section (XZR_DATA)	arbitrarily many lines with data of type 3 x length	data triples defining x/z data of the cleat's cross section, together with rounding radii in these data points. Rounding radii may be zero. x-values should be in ascending order. First and last y-value should be zero
--------------------------------	---	--

Optionally, the exact height geometry of the stoppers can be specified by spline data:

data sub-section (stopper_geometry)	arbitrarily many lines with data of type 2 x length	data pairs defining x/z data of the stoppers. x-values must be non-negative and ascending: $x = 0$ refers to the stopper's start in longitudinal direction
--	---	--

3.2 TeimOrbit 2D Road Type **flatbelt**

The rotating flat-belt testrig is implemented as a 2D type road. If ROAD_TYPE is set to flatbelt, *cosin/road* searches for and reads the following data items in section [PARAMETERS]:

Name of input variable	Unit	Meaning
number_cleats	-	number of extra cleats on belt (number_cleats = 0 allowed)
cleat_direction	angle	direction of cleat relative to belt lateral direction. Direction is zero if cleat is oriented exactly in transversal direction (which is the default case). Direction angle is measured counter-clockwise if looking from above onto the belt surface. Default value is 0 (transversal cleat)
mu_factor_cleat	-	friction modification factor on cleats

The flat-belt speed is defined by one of the following alternatives:

Either:

v	length/time	constant rotation speed of drum surface
acceleration_time	time	time span at beginning of dynamic simulation, to accelerate drum from stand-still to nominal velocity Default value is 0

or:

data sub-section (TV_DATA)	arbitrarily many lines with 2 values of type time, length/time	data pairs defining speed-vs.-time data of the drum's time-dependent surface speed profile. t-values (first value in each line) should be in ascending order
-------------------------------	--	--

The cleat's exact geometry is defined by one of the three following alternatives:

Either:

cleat_height	length	height of cleats
cleat_length	length	length of cleat, measured in circumferential direction of drum
cleat_bevel_edge_length	length	length of bevel edge of cleat, measured in circumferential direction of drum. Bevel edge has 45deg slope
cleat_edge_rounded	-	0: cleat bevel edge flat 1: cleat bevel edge rounded by a quarter circle

or:

data sub-section (XZ_DATA)	arbitrarily many lines with data of type 2 x length	data pairs defining x/z data of the cleat's cross section. No rounding of edges is assumed. x-values should be in ascending order. First and last y-value should be zero
-------------------------------	---	--

or:

data sub-section (XZR_DATA)	arbitrarily many lines with data of type 3 x length	data triples defining x/z data of the cleat's cross section, together with rounding radii in these data points. Rounding radii may be zero. x-values should be in ascending order. First and last y-value should be zero
--------------------------------	---	--

Optionally, the exact height geometry of the stoppers can be specified by spline data:

data sub-section (stopper_geometry)	arbitrarily many lines with data of type 2 x length	data pairs defining x/z data of the stoppers. x-values must be non-negative and ascending: $x = 0$ refers to the stopper's start in longitudinal direction
--	---	--

3.3 TeimOrbit 2D Road Type **hydraulic_test_rig**

A general, time history-driven servohydraulic test rig excitation (eventually combined with a flat belt test rig) is implemented as another 2D type road. If ROAD_TYPE is set to hydraulic_test_rig, *cosin/road* searches for and reads the following data items in section [PARAMETERS]:

Name of input variable	Unit	Meaning
sampling_time	time	assumed time increment of wheel support displacement values, if no separate time channel is provided
flat_belt_velocity	length/time	optional time span at beginning of simulation, during which an optional, additional flat belt device is accelerated from stand-still to nominal speed Default value is 0 (no additional flat belt)
acceleration_time	time	time span at beginning of dynamic simulation, to accelerate the flat belt device from stand-still to nominal velocity Default value is 0

The wheel support displacement (either purely in vertical (z) direction, or fully spatial in x/y/z direction) is defined by one of the following alternatives:

Either:

data sub-section (Z_DATA)	arbitrarily many lines with a single value of type length	values defining time-dependent vertical wheel support displacement values. Time values are assumed to be equidistant, with increment defined by sampling_time
------------------------------	---	---

or:

data sub-section (XYZ_DATA)	arbitrarily many lines with 3 values each of type 3 x length	data triples defining x/y/z data of the time-dependent spatial wheel support displacement. Time values are assumed to be equidistant, with increment defined by sampling_time
--------------------------------	---	---

or:

data sub-section (TZ_DATA)	arbitrarily many lines with 2 values each of type time,length	data defining t/z data of the time-dependent spatial wheel support displacement. Time values must be in ascending order. x/y displacements are set to zero
-------------------------------	--	--

or:

data sub-section (TXYZV_DATA)	arbitrarily many lines with 5 values each of type time, 3 x length, length/time	data defining t/x/y/z data of the timedependent spatial wheel support displacement, and in addition velocity of the flat belt device. Time values must be in ascending order
----------------------------------	--	--

3.4 TeimOrbit 2D Road Type **plank** or **cleat**

If ROAD_TYPE is set to cleat, *cosin/road* searches for and reads the following data items in section [PARAMETERS]:

Name of input variable	Unit	Meaning
height	length	height of plank/cleat
start	length	start of plank/cleat (travel distance)
length	length	length of plank/cleat, measured along x-axis
bevel_edge_length	length	length of bevel edge, measured along x-axis. Bevel edge has 45deg slope. With edge_rounded = 1, rounded corners instead of bevel edges are used. In this case, bevel_edge_length is radius of the corner
edge_rounded	0/1	1: edge rounded, 0: edge linear, not rounded
direction	angle	direction of plank/cleat relative to y-axis. direction = 0 if plank/cleat is placed crosswise
mu_factor_cleat	-	friction value factor on plank/cleat

3.5 TeimOrbit 2D Road Type **pot_hole**

If ROAD_TYPE is set to pot_hole, *cosin/road* searches for and reads the following data items in section [PARAMETERS]:

Name of input variable	Unit	Meaning
depth	length	depth of pot-hole
start	length	start of pot-hole (travel distance)
length	length	length of pot-hole

3.6 TeimOrbit 2D Road Type **ramp**

If ROAD_TYPE is set to ramp, *cosin/road* searches for and reads the following data items in section [PARAMETERS]:

Name of input variable	Unit	Meaning
height	length	height of ramp
start	length	start of ramp (travel distance)
slope	length	slope of ramp; 1 means 45deg

3.7 TeimOrbit 2D Road Type **roof**

If ROAD_TYPE is set to roof, *cosin/road* searches for and reads the following data items in section [PARAMETERS]:

Name of input variable	Unit	Meaning
height	length	height of 'roof' (= triangle-shaped obstacle)
start	length	start of 'roof' (travel distance)
length	length	length of 'roof', measured along x-axis

3.8 TeimOrbit 2D Road Type **sine**

If ROAD_TYPE is set to sine, *cosin/road* searches for and reads the following data items in section [PARAMETERS]:

Name of input variable	Unit	Meaning
amplitude	length	amplitude of sine wave
wave_length	length	wave length of sine wave
start	length	start of sine wave (travel distance)
length	length	length of wave section (travel distance, optional)

3.9 TeimOrbit 2D Road Type **sine_sweep**

A sine-shaped road profile with slowly varying frequency and amplitude is calculated in two different ways:

linear sweep: the frequency increases linearly with respect to travel distance. The road height value $z(s)$ as function of travel distance s is calculated as follows:

$$z(s) = \left(a_s + \frac{a_e - a_s}{s_e - s_s} (s - s_s) \right) \cdot \sin \left(2\pi \cdot \left(f_s + \frac{f_e - f_s}{2(s_e - s_s)} (s - s_s) \right) \cdot (s - s_s) \right)$$

(note the factor '2' in denominator, which is not an error!). The actual frequency (= derivative of the sine function argument with respect to travel path, divided by 2π ; this is not equal to that factor that is multiplied by $2\pi (s - s_s)$ in the sine function!) is given by

$$f(s) = f_s + \frac{f_e - f_s}{s_e - s_s} (s - s_s)$$

logarithmic sweep: with every cycle, the wave length decreases by a constant factor. The road height value is calculated as follows:

$$z(s) = \left(a_s + \frac{a_e - a_s}{s_e - s_s} (s - s_s) \right) \cdot \sin \left(2\pi f_s s_\infty \ln \frac{s_\infty}{s_\infty + s_s - s} \right)$$

where

$$s_\infty = \frac{f_e}{f_e - f_s} (s_e - s_s).$$

s_∞ is that travel path where theoretically an infinitely high frequency was reached, measured relative to sweep start s_s . The actual frequency is given by

$$f(s) = \frac{s_\infty}{s_\infty + s_s - s} f_s$$

The swept sine road is implemented as a 2D type road. If ROAD_TYPE is set to `sine_sweep`, `cosin/road` searches for and reads the following data items in section [PARAMETERS]:

Name of input variable	Unit	Meaning
start	length	start of swept sine wave (travel distance)
end	length	end of swept sine wave (travel distance)
amplitude_at_start	length	amplitude of swept sine wave at start
amplitude_at_end	length	amplitude of swept sine wave at end
wave_length_at_start	length	wave length of swept sine wave at start
wave_length_at_end	length	wave length of swept sine wave at end. Must be less or equal wave_length_at_start
sweep_type	0/1	0: frequency changes linearly with respect to travel distance 1: wave length changes each cycle by a constant factor

3.10 TeimOrbit 2D Road Type **stochastic_uneven**

A stochastic uneven road profile both for left and right wheels is generated, that has properties very close to measured road profiles.

To this end, in a first step discrete white noise signals are formed on the basis of nearly **uniformly** distributed random numbers, two of these assigned to every 10 mm of travel path. The distribution of these random numbers is approximated by summing up several **equally** distributed random numbers, taking advantage of the 'law of large numbers' of mathematical statistics.

Next, these values are integrated with respect to travel distance, using a simple first order time discrete integration filter. The reason for not just using a pure integrator is to cut off extremely low frequencies, which would result in large road elevation values, not at all affecting vehicle dynamics.

The independent variable of that filter is not time, but travel path. That is why the filter cut-off frequency Ω is controlled by a '**path** constant' $S = \Omega_c^{-1}$ instead of a **time** constant. Approximate power cc spectral density (PSD) of this road surface profile is given by

$$G_d(\Omega) = \frac{K^2}{\Omega^2 + S^2} \approx G_d(\Omega_0) \cdot \left(\frac{\Omega_0}{\Omega}\right)^2$$

where

$$S \gg 1 \text{ m}$$

$$G_d(\Omega_0) = \left(\frac{K}{\Omega_0}\right)^2$$

The filter process results in two realizations of this approximate '**white** velocity noise'; that means, two signals, the derivatives of which are close to **white noise**. Signals with that property are known as road profiles with '**waviness**' $w = 2$ (cf. ISO 8608). Several investigations show that the waviness derived from measured road displacement PSDs ranges from about 1.8 to 2.2.

The reference spectral density value $G_d(\Omega_0)$, with $\Omega_0 = 1 \text{ rad/m}$, is used in ISO 8608 to classify road surface profiles, according to the following table:

Road class	$G_d(\Omega_0) [10^{-6} m^3]$		
	min. value	mean value	max. value
A	-	1	2
B	2	4	8
C	8	16	32
D	32	64	128
E	128	256	512
F	512	1024	2048
G	2048	4096	8192
H	8192	16384	-

The last step in the generation of the stochastic uneven road profiles is to linearly combine the two realizations $z_1(s)$, $z_2(s)$ of the above mentioned process, resulting in the left and right profile $z_l(s)$,

$z_r(s)$. This is done such that these two signals are completely independent, if $\text{correlation_rl} = 0.0$, and identical, if $\text{correlation_rl} = 1.0$:

$$z_l(s) = z_1(s) + \frac{\text{corr}_{rl}}{2} (z_2(s) - z_1(s))$$

$$z_r(s) = z_2(s) + \frac{\text{corr}_{rl}}{2} (z_2(s) - z_1(s))$$

The stochastic uneven road is implemented as a 2D type road. If `ROAD_TYPE` is set to `stochastic_uneven`, `cosin/road` searches for and reads the following data items in section [PARAMETERS]:

Name of input variable		Unit	Meaning
one of	reference_spectral_density	m ³	$G_d(\Omega_0)$
	ISO_8608_road_class	A, B, ...	ISO 8608 road class, according to table above. Mean $G_d(\Omega_0)$ value of the respective class will be used
path_constant		m	'path constant' S to control high-pass integration filter cut-off frequency in path domain. Parameter is optional, default value is 1000 m
correlation_rl		-	variable to control correlation between left and right track: no correlation, if zero; complete correlation (that is, left track = right track), if one. Any value between 0 and 1 is allowed

3.11 TeimOrbit 2D Road Type **tilt_table**

If `ROAD_TYPE` is set to `tilt_table`, `cosin/road` searches for and reads the following data items in section [PARAMETERS]:

Name of input variable	Unit	Meaning
y_coord_rot_axis	mm	y coordinate of tilt-table's rotation axis (which is assumed to be parallel to global x-axis)
start_time	s	start time of tilt table's rotation
angular_velocity	deg/s	tilt table's angular velocity about x-axis; assumed to be constant during operation

3.12 Superposition of TeimOrbit 2D Roads with 3D Roads

All TeimOrbit 2D roads can be superimposed by a triangulated or a RGR-type 3D road. This is achieved by specifying the name of the respective 3D road data file in data section [MODEL] of the 2D road data file. This technique, for example, can be used to define a non-flat hydraulic test-rig cylinder surface by means of RGR data:

Name of input variable	Unit	Meaning
superimpose_data_file	-	name of the 3D road data file (triangular or RGR type) to be superimposed to the 2D data

4 Regular Grid Road Data Files (RGR Files)

This chapter describes the format of Regular Grid Road Data files (RGR-files). RGR-files can be used to describe high resolution road surface measurements for use with *FTire* or other high-end tire models in road load simulations. The data format is designed such that both memory amount and evaluation effort is as small as possible.

4.1 Regular Grid Data Items

In order to make the evaluation of road surface data as efficient as possible, it is advantageous to use data points that are equally spaced in x- and y-direction. Such data are called regular grid data.

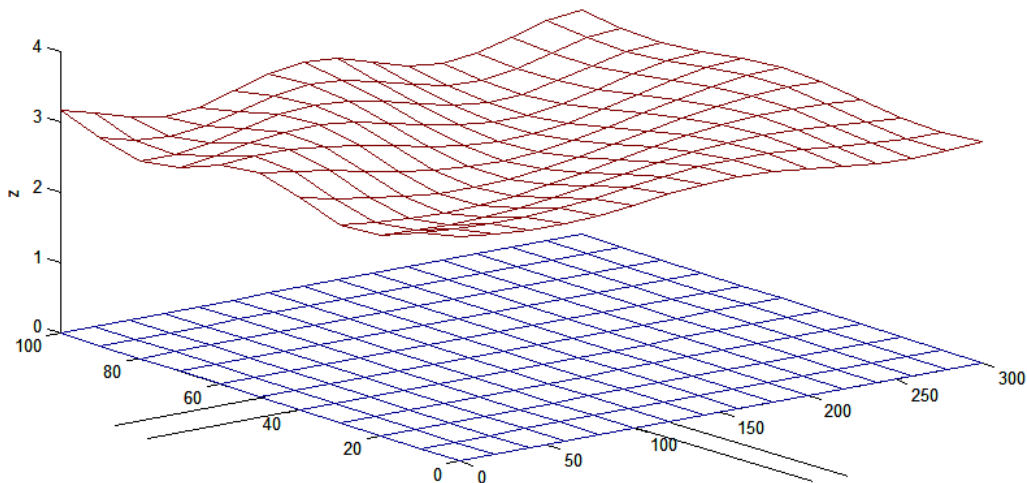


Figure 1: Regular Grid Geometry

Regular grid data are defined by

- the grid's minimum x-value: x_{min}
- the grid's constant mesh-size in x-direction: Δx

- the number of grid nodes in x-direction: n_x
- the grid's minimum y-value: y_{min}
- the grid's constant mesh-size in y-direction: Δy
- the number of grid nodes in y-direction: n_y
- the surface's z-values in the grid points: $z_{ik} = z(x_i, y_k)$, where $x_i = x_{min} + (i-1)\Delta x$ and $y_k = y_{min} + (k-1)\Delta y$.

Typically (but not necessarily), the x-direction is close to the vehicle's main driving direction. The z_{ik} -values are stored 'x-strip-wise', in the following sequence:

$$z_{11}, z_{12}, \dots, z_{1n_y}, z_{21}, z_{22}, \dots, z_{2n_y}, \dots, z_{n_x 1}, z_{n_x 2}, \dots, z_{n_x n_y}$$

Length unit of all x, y, and z-values is millimeter [mm], if not specified otherwise in an accompanying RDF-file (see chapter 4.3).

4.2 Regular Grid Data File Format

Regular grid data files may be specified inside an accompanying RDF-file. This RDF-file is required by some 3rd-party simulation environments as road interface.

Because, typically, RGR-files are extensions to the calling simulation program, it is not possible to visualize them directly within the animation window in this environment. Rather, such a road surface will be displayed in an FTire animation window. An alternative method is to create and use an equivalent SHL-file, by means of *FTire/roadtools*.

The accompanying RDF-file must be one out of the following types:

- method = '2d' and road_type = 'flat'
- method = '3d_spline' (discussed in chapter 4.3)

In either case, the regular grid data file is specified by an optional data item

- RGR_DATA_FILE = 'file-name'

in section [MODEL] of the RDF-file, observing the respective operating system's file naming conventions.

In older versions, the regular grid data file had to be specified in a comment line of the RDF-file's header section. This way of specification, though still valid and recognized, is obsolete and might be removed in coming versions.

The regular grid data file, if in ASCII format, see below, might coincide with the RDF-file. This coincidence is to be specified by the file-name '*'. In this case, the regular grid data are expected in the section [RGR_DATA] of the RDF-file.

The regular grid data file (\roads\rgr.dat in the example above) may be either in ASCII or in binary format. In the case of ASCII format, it will contain

- x_{min} , Δx , and n_x in the first record,

- y_{min} , Δy , and n_y in the second record,
- z_{ik} , one per line, in the subsequent records, in the order indicated above.

Instead of the first two lines mentioned above, an alternative, formatted header line can be specified as first line. The single header-line has the following syntax:

- `$RGR_data xmin=0 dx=5 nx=5000 ymin=-1000 dy=5 ny=401 mu=1.0 ! comment`

This alternative allows the optional specification of a friction coefficient correction factor μ , as well as data defining an additional soft soil/snow model, see chapter 4.4.

In either case, all data items use the same length unit as specified in the accompanying RDF-file. They can be written in any valid number format, with or without decimal point and/or exponent. To make the loading of such a (usually huge) data file as fast as possible, no blank or comment lines are allowed. In any line however, after the mandatory number of data items, any arbitrary comment string is admissible, as long as it is separated by at least one blank space or tab from the last data item. Numbers, if more than one are allowed in a single line, must not be separated by commas. Use one or more blank spaces instead.

If the data file is given in binary format, *cosin/roads* it will automatically recognize and manage the correct byte ordering ('little endian' or 'big endian'). As a consequence, all binary RGR files can be read without any adaptation on all operating systems, no matter on what other system they had been created.

All floating point values are saved in single precision, using 4 byte each, all integer data with 4 bytes either. No line separation is provided. The sequence of data items is

- x_{min} , Δx , n_x , y_{min} , Δy , n_y , z_{ik} (in the order indicated above).

Instead of the binary values for x_{min} , Δx , n_x , y_{min} , Δy , n_y , a formatted ASCII header string can be specified as the file's leading bytes. This header-string is essentially the same as in completely ASCII-formatted files, described above. It has the syntax:

- `$RGR_data xmin=0 dx=5 nx=5000 ymin=-1000 dy=5 ny=401 mu=1.0 !`

Binary z-value data are assumed to start immediately after the trailing exclamation mark, which is mandatory; a line separator or comment string is neither necessary nor admissible after the exclamation mark.

Note that in either case the number of z_{ik} -values must be either zero (flat surface at $z = 0$ is assumed in this case), or at least $n_x \cdot n_y$. Otherwise, the simulation will stop with an error message.

4.3 Combination with Curved Center-Line Data Files

Regular grid data files can also be combined with curvilinear road center-lines, as they are defined in TeimOrbit data files of type '3D-Spline'.

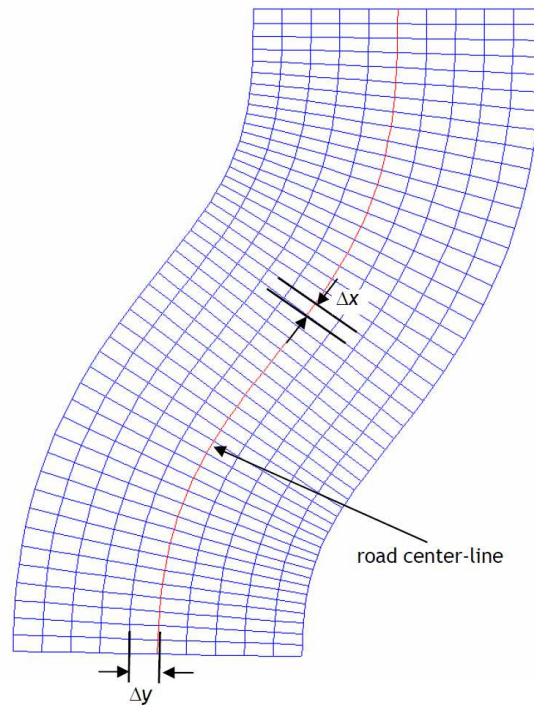


Figure 2: Curved Center-Line in RGR Model

An additional optional data item `RGR_DATA_FILE = 'file-name'`, or several items of the form `RGR_DATA_FILE_i='i file-name'` in section [MODEL] of such a RDF-file will cause *FTire*'s road evaluation routine to combine the RDF-file with the specified regular grid data file(s). The required format of these data files must be exactly as described in chapter 4.2 above.

As with the 2d RDF-files mentioned above, the regular grid data file might coincide with the RDF-file, being specified by `RGR_DATA_FILE = '*'`. In this case, the regular grid data are expected in the section [RGR_DATA] of the RDF-file. Below, a sample 3D-Spline RDF-file is partially listed, containing the reference to a regular grid data file.

```

$-----COSIN_HEADER
[COSIN_HEADER]
FILE_TYPE='rdf'
FILE_VERSION=5.00
FILE_FORMAT='ASCII'
(COMMENTS)
{comment_string)
'3d Spline Road'
$-----UNITS
[UNITS]
LENGTH           = 'meter'
FORCE             = 'newton'
ANGLE             = 'radians'
MASS              = 'kg'
TIME              = 'sec'

```

```

$-----DEFINITION
[MODEL]
METHOD          = '3D_SPLINE'
VERSION         = 1.00
RGR_LENGTH_UNIT = 'mm'
RGR_DATA_FILE   = '/road/rgr.dat'
$-----ROAD_PARAMETERS
[GLOBAL_PARAMETERS]
CLOSED_ROAD     = 'YES'
SEARCH_ALGORITHM = 'fast'
ROAD_VERTICAL   = '0.0 0.0 1.0'
FORWARD_DIR     = 'NORMAL'
MU_LEFT        = 1.0
MU_RIGHT       = 1.0
WIDTH          = 15.0
$-----DATA_POINTS
[DATA_POINTS]
{
  X      Y      Z      WIDTH  BANK  MU_LEFT  MU_RIGHT }
 10.0000 0.0000 0.0000 12.0000 0.0000 0.900  0.900
  8.0000 0.0000 0.0000 12.0000 0.0000 0.900  0.900
  6.0000 0.0000 0.0000 12.0000 0.0000 0.900  0.900
  4.0000 0.0000 0.0000 12.0000 0.0000 0.900  0.900
  2.0000 0.0000 0.0000 12.0000 0.0000 0.900  0.900
  0.0000 0.0000 0.0000 12.0000 0.0000 0.900  0.900
 -2.0000 0.0000 0.0000 12.0000 0.0000 0.900  0.900
 -4.0000 0.0000 0.0000 12.0000 0.0000 0.900  0.900
 -6.0000 0.0000 0.0000 12.0000 0.0000 0.900  0.900

```

The combination is done as follows: the road center line, as defined in the RDF-file, is used as curvilinear x-axis of the grid. x data points are equidistant along the center line with a spacing of Δx . The x -coordinate starts with value 0 at the beginning of the road, which is defined by the first data point in the RDF-file. Note that the spacing of the data points in the RDF-file is completely independent on the spacing of the curvilinear regular grid.

The grid's y -coordinate is chosen to be perpendicular to the center-line. That is, it coincides with the center-line's normal in the road surface plane. Positive y -values define grid points that are located on the left of the center-line, if viewed along the direction of increasing x -values.

An RGR-file, defined inside an RDF-file, will assume the same length unit as the RDF-file, if not a data item `RGR_LENGTH_UNIT = 'unit'` is specified in section [MODEL]. 'unit' can be any one of the length units supported by the TeimOrbit file, and will be used by the RGR-file instead of the RDF-file's length unit.

In order to make the regular grid data evaluation well-defined and unique, the center line's minimal curvature radius is required to be greater than half the grid width.

If the 3D-Spline data define nonzero z-values at the center-line and/or a non-zero banking angle, the regular grid's z-values are superimposed to the resulting RDF-file's z-values. More than one regular grid data files may be specified, like in the example snippet below.

```

$----- DEFINITION
[MODEL]
METHOD          = '3D_SPLINE'
VERSION         = 1.00
RGR_LENGTH_UNIT = 'mm'
RGR_DATA_FILE_1 = '\roads\patch1.dat'
RGR_DATA_FILE_2 = '\roads\patch2.dat'
RGR_DATA_FILE_3 = '\roads\patch3.dat'
RGR_DATA_FILE_4 = '\roads\patch4.dat'
RGR_DATA_FILE_5 = '\roads\patch5.dat'
RGR_DATA_FILE_6 = '\roads\patch6.dat'
$----- ROAD_PARAMETERS

```

In this case, *cosin/road* will dynamically load/unload the respective regular grid data file. Loading/unloading will depend on whether or not the actual position is inside the extent of the actually loaded regular grid data file (called a patch). If not, *cosin/road* will search the patch that contains the respective position. In order to prevent time-consuming frequent loading/unloading, the patches should overlap such that the full vehicle completely fits into the overlapping regions.

cosin/road's evaluation of 3d-Spline data files accepts the following extra options in section [GLOBAL_PARAMETERS]:

- LINEAR_INTERPOL = 'YES' will interpolate the center-line data points piecewise linearly, rather than by a smooth spline;
- LINEAR_INTERPOL_MU = 'YES' will interpolate the friction correction factors $\mu(s)$, defined together with the center-line data points, piecewise linearly rather than by a smooth spline.

4.4 Combination with Soft-Soil or Snow Models

If using the formatted ASCII header line, beginning with the key-word \$RGR_data, an additional soft-soil model can be activated and parameterized within this line. This soil-model will take *FTire*'s contact forces to modify the actual road surface geometry.

At present, only a simple place-holder model is implemented. This model is described by the following local dependency between contact pressure and road surface sinking:

$$F_d(\dot{z}) + k(z_0 - z) + p_{cont} = 0$$

with: $z_0(x, y)$ undeformed surface height, $z(x, y)$ deformed surface height, and $p_{cont}(x, y)$ local contact pressure.

The damping term F_d is strongly nonlinear. The large slope for positive height changes ensures a persistent soil compaction:

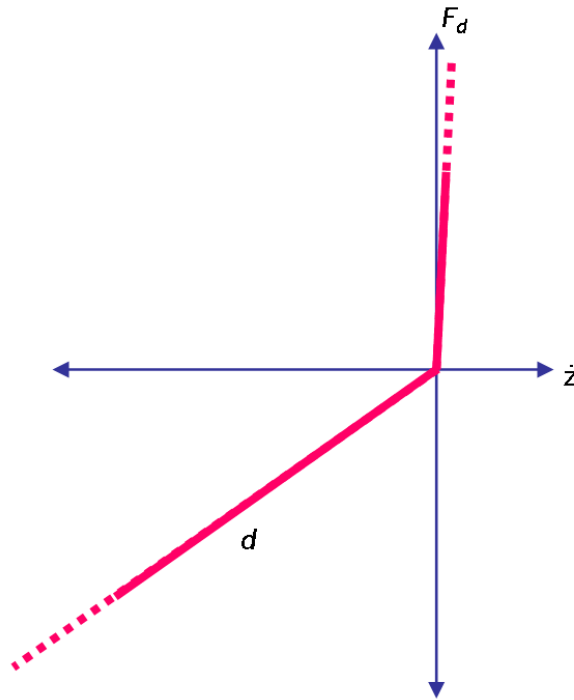


Figure 3: Non-Linear Damping Characteristic in Soft-Soil Placeholder Model

The model is quantified by two parameters (together with typical values):

$$k = 0.025 \frac{MPa}{mm}$$

$$d = 0.1 k$$

(optionally, the model can be activated only inside a rectangle $[x_{soil,min}, x_{soil,max}] \times [y_{soil,min}, y_{soil,max}]$). This simple model is activated with the following entries in the header line:

- `$RGR_data xmin=0 dx=5 nx=1000 ymin=-200 dy=5 ny=80 mu=1.0`
`soil_model=1000 smp1=0.025 smp2=0.1 !`

(the optional rectangle bounds $x_{soil,min}$, $x_{soil,max}$, $y_{soil,min}$, $y_{soil,max}$ are specified by smp3, smp4, smp5, and smp6).

The model type 1000, selected by the entry `soil_model=1000`, activates the simple model as described above (more model types might follow later). The entries `smp1=0.025` and `smp2=0.1` specify the numerical values of the two respective model parameters.

In case of a soft soil model, the grid data section may be omitted completely. In this case, a surface which is originally flat is assumed. Note that for a rigid surface, this is formally allowed as well, but would not be meaningful. There are simpler ways to describe a perfectly flat and rigid surface.

If no grid data are given, the grid resolution can be adapted automatically to the tire's tread resolution. This is done if `dx` and `dy` are not set. Moreover, rather than specifying `nx` and `ny` (which would lead to a grid of unknown extent), `xmax` and `yymax` may be defined in this case:

- `$RGR_data xmin=0 xmax=5000 ymin=-200 ymax=200 mu=1.0 soil_model=1000 smp1=0.025`
`smp2=0.1`

5 *cosin* Road Track Files

If the road data file is in *cosin/io* format, but does not contain the data block `$road_type`, it is interpreted as ***cosin* road track file**. *cosin* road track files, among others, give a graphical representation of

- road tracks,
- road shoulders,
- center lines,

and more. Road track files also provide an easy-to-apply definition of nominal tracks to follow with driver models, and full 3D road profiles including road transversal inclination, banked curves, etc. Please contact info@ftire.com for more details on *cosin* road track files.

6 User-Defined Road Models

If the road data file has file extension **urm**, *cosin/road* assumes that the user provided an own road evaluation routine, compiled into a dynamically loadable library named

- **urm.dll** in Windows, and
- **liburm.so** (or **liburm.sl**, respectively) in Linux and Unix.

This library is searched for, according to the rules set by the respective operating system. Safest way, in either case, to make it available is to put it into the working directory from which *cosin/road* and its calling solver is invoked. The library must contain a C or C++ function with the following prototype:

```
extern void urm (int ti, double t, double x, double y, double*z,  
                double*vx, double*vy, double*vz, double*mu,  
                int*ier, char*file);
```

The sole task of this routine is to provide

- road height z ,
- road surface velocities v_x , v_y , v_z , and
- friction modification factor μ ,

all being functions of

- time t , and
- location x , y .

cosin/road passes to this routine the name of the *data file*, with extension *urm* as mentioned above, which is to be read and interpreted under the sole responsibility of the user-defined road model.

The meaning of the invocation parameters is as follows:

Parameter	C/C++ type	Data flow	Unit	Meaning
ti	int	in	-	wheel index. Typically: 1=fl, 2=fr, 3=rl, 4=rr, ..
t	double	in	s	simulation time, provided by <i>cosin/road</i>
x	double	in	m	x-comp. of location where road height is needed; provided by <i>cosin/road</i>
y	double	in	m	y-comp. of location where road height is needed; provided by <i>cosin/road</i>
z	double*	out	m	road height
vx	double*	out	m/s	x-comp. of road surface velocity relative to global coordinate system (non-zero for example in drum or flat-belt simulations)
vy	double*	out	m/s	y-comp. of road surface velocity relative to global coordinate system
vz	double*	out	m/s	z-comp. of road surface velocity relative to global coordinate system (non-zero for example in hydraulic 4-poster simulations)
mu	double*	out	-	friction characteristic modification factor (value is typically 1.0, which means no modification of the friction characteristic as defined by the tire model)
ier	int*	out	-	error code, must be 0 if road evaluation (or file opening and reading) was successful, any value other than 0 else
file	char*	in	string	name of road data file (provided by <i>cosin/road</i>). May or may not contain the path; the exact interpretation of the string is according to the rules of the respective operating system. It is up to the user routine to decide when this file is to be opened and read. The name will be provided in each call to <i>urm</i> . However, typically, it needs to be read only during first call of <i>urm</i> with the respective wheel index. <i>urm</i> might have to save the information in the file in local but static variables.

Below, you find a listing of a most simple example of such a C function. The code (*urm.c*) is contained in sub-folder *sdk* of the *FTire/lib* download:

```

/* place holder for user defined road model (URM) */
#include <stdio.h>

extern void urm (int ti, double t, double x, double y, double*z,
                double*vx, double*vy, double*vz, double*mu, int*ier, char*file) {

    static int first=1;

    if (first) {
        printf("\nthis is the demo user road model, using data file %s..\n",file);
        first=0;
    }

    /* terminate road model */
    if (t>=0.9e60) {
        return;
    }

    *z=0.0;
    if (x>1.0 && x<1.2) *z=0.02;
    *vx=0.0;
    *vy=0.0;
    *vz=0.0;
    *mu=1.0;
    *ier=0;
}

```

In Windows, provided you have installed any of the Microsoft C/C++ compilers, this function can be compiled and linked into a dynamical link library with the batch file **makeum.bat**, likewise contained in sub-folder **sdk** of the *FTire/lib* download. The file assumes that the installation location of your compiler is contained in the respective search paths. This holds for regular installations.

Index

C

[cleat](#), 11, 23

D

[drum](#), 5, 16

F

[file](#), 8

[flatbelt](#), 7, 20

[function](#), 9

H

[hydraulic_test_rig](#), 21

[hydropulse_function](#), 11

[hydropulse_harmonic](#), 10

[hydropulse_noise](#), 10

P

[plank](#), 11, 23

[pot_hole](#), 11, 23

R

[ramp](#), 12, 24

[roof](#), 12, 24

S

[sine](#), 12, 24

[sine_sweep](#), 12, 24

[spline](#), 14

[stochastic_uneven](#), 14, 25

T

[tilt_table](#), 16, 27