cosin
scientific software

cosin scientific software AG

Hans-Stützle-Str. 20
81249 München
GERMANY

info@cosin.eu
www.cosin.eu

cosin/road

Cosin Road Modeling

Documentation and User's Guide

Contents

Document Revision: 2021-2-r24053

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.

1 Introduction

To specify and evaluate road excitations, **cosin/road** uses nothing but one central subroutine `evrmr` (**cosin** ***e****n****v****ironmental models:* ***r****oad* ***m****ain* ***r****outine*).

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

- **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,
- 3^{rd}-party generally available road models, like the CRG format,

and more.

In the following chapters, the most important road models, from the list above, 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 3rd party simulation software 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 that the road file is in *cosin/io* format, it first tries to read certain information that is common to all road types within this format. Then, depending on the respective road or obstacle type, it will search for more data that is specific to this road type.

2.1 Data Used by All cosin/io-style Road Types

If the *cosin/io* format has been recognized, **cosin/road** searches the data block `$road_type` and reads the below mentioned data. This data is then used for the road profile calculations of all road types and in all subsequent invocations:

Name of input variable	Unit	Meaning
`type`	-	a character string out of the following list: • `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. Typically, for each type additional parameters are required, described in chapter 1 or in the sequel
`offset`	mm	value to be added after calculation of the reference road height (that is, a shift of the road in vertical direction). Parameter is optional

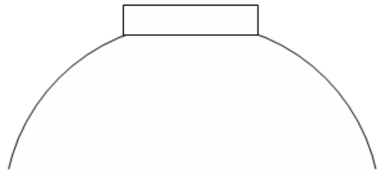
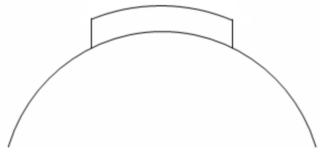
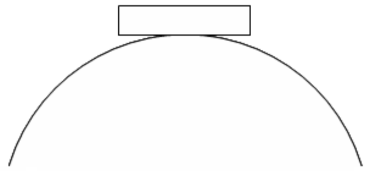
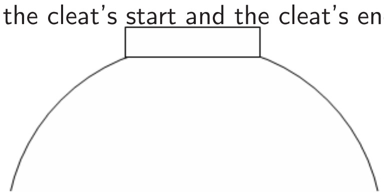
Name of input variable	Unit	Meaning
`start_at`	m	to minimize irrelevant tire accelerations when momentarily applying a tire deflection, start of the road profile as defined by the contents of the data block `$road_type` can be delayed by use of a 'ramp' at the beginning of the simulation With parameter `start_at`, the end of this ramp and start of the regular road is defined in terms of the travel distance. This delay does not apply to the road types **drum** and **hydropulse**. Additionally, several road types have a parameter **start** that marks the beginning of a single obstacle relative to a travel distance of 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 a period interval as specified by this parameter (optional)
`x_period`	m	road height will be periodic in the x-coordinate, with a period length as specified by this parameter (optional)
`y_period`	m	road height will be periodic in the y-coordinate, with a period length as specified by this parameter (optional)
`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 searched for and read, in accordance to the following tables:

2.2 cosin/io 2D Road Type `drum`

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

Name of input variable	Unit	Meaning
diameter	m	diameter of the tire test drum. **cosin/road** assumes an outer drum, if diameter $< \mathbf{0}$
wheel_placement	-	0 above drum (default) 1 before drum 2 on the left of drum 3 below drum 4 behind drum 5 on the right of drum
v	m/s	surface speed of the rotating drum surface (be sure to choose rolling_speed = 0 in data block $sources. Otherwise, the wheel will move away from the drum)
acceleration_time	s	optional time span at beginning of the simulation during which the drum is accelerated to the nominal speed
number_cleats	-	number of extra cleats on the drum (number_cleats = 0 allowed)
cleat_height	mm	height of the extra cleats
cleat_starting_angle	deg	drum angle coordinate of the 1^{st} cleat
cleat_width	mm	width of the cleat, measured in transversal direction of the cleat (formerly called cleat_length)
cleat_length_x	mm	length of the cleat, measured in circumferential direction of the drum. Specification of cleat_length_x will overwrite cleat_width. Both differ if cleat_angle is non-zero
cleat_bevel_edge_length	mm	length of the bevel edge of the cleat, measured in circumferential direction of the drum. Bevel edge has a 45deg slope
cleat_edge_rounded	0/1/-1	1: bevel edge rounded, 0: bevel edge linear, not rounded, -1: bevel edge concavely rounded
cleat_angle	deg	direction of the cleat relative to the y-axis. Direction angle is measured in counter-clockwise direction, if looking from above onto the drum surface. cleat_angle = 0 if the cleat is placed parallel to the drum rotation axis. **NOTE**: for releases prior to 2018-3 the cleat direction for diameter $< \mathbf{0}$ has been measured in clockwise direction and for diameter $> \mathbf{0}$ in counter-clockwise direction, using the deprecated item cleat_direction
mu_factor_cleat	-	friction modification factor of the cleat surface

Name of input variable	Unit	Meaning
`curved_cleat_surface`	-	0: cleat's surface is flat  1: cleat's surface is curved according to the drum surface curvature.  **Default value** is 0 (flat)
`flat_cleat_support`	-	0: cleat is mounted on the drum such that its support, on the drum surface, is curved according to overall drum curvature  1: cleat is mounted on the drum such that its support, on the drum surface, is flat between the cleat's start and the cleat's end  **Default value** is 0 (non-flat cleat support). This parameter is only used and relevant if `curved_cleat_surface = 0`
`stopper_distance`	mm	distance between front and rear stopper of the 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. The data is mirrored for the rear stopper. If specified, it is recommended to check the actual appearance of drum surface and stoppers, using **cosin/tools**.

Name of input variable	Unit	Meaning
`non_uniformity_spline`	string	name of an optional spline data-block, containing *phi/r* data pairs. They describe the angle-dependent radial non-uniformity of the drum surface. *phi* (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]
`cleat_lat_shape_fact_spline`	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
`drum_lat_shape_spline`	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
`cleat_raise_time`	s	start time at which the cleat is raised. **Default value** is 0.05s (set `cleat_raise_time = -1` to raise the hight of the cleat to `cleat_height` at the simulation start)
`cleat_raise_duration`	s	time span during which the cleat is raised to reach the defined cleat height. **Default value** is 0.02s

If one or more of the four optional spline data-blocks, as mentioned at the end of the above table, is specified, **cosin/road** looks for the spline data in the 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 a data block `$flatbelt`, and reads:

Name of input variable	Unit	Meaning
`v`	m/s	speed of the belt surface (be sure to choose `rolling_speed = 0` in data block `$sources`. Otherwise, wheel will move away from the belt)
`number_cleats`	-	number of the extra cleats on the belt (`number_cleats = 0` allowed)
`cleat_height`	mm	height of the extra cleats
`cleat_width`	mm	width of the cleats, measured in the transversal direction of the cleat (formerly called `cleat_length`)
`cleat_length_x`	mm	length of the cleats, measured in the direction direction of the belt travel. Specification of `cleat_length_x` will overwrite `cleat_width`. Both differ if `cleat_angle` is non-zero
`cleat_angle`	deg	direction of the cleat relative to the y-axis. Direction angle is measured in counter-clockwise direction, if looking from above onto the belt surface. `cleat_angle = 0` if the cleat is placed perpendicular to the belt motion
`mu_factor_cleat`	-	friction modification factor of the cleat surface
`acceleration_time`	s	optional time span at beginning of the simulation during which the belt is accelerated to the nominal speed

2.4 cosin/io 2D Road Type `file`

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

Name of input variable	Unit	Meaning
`file_name`	-	name of a file to read measured or calculated road data from. The file must be in standard or Matlab format (cf. *cosin/io* documentation). **cosin/road** expects equidistant road profile height data as columns of the matrix stored in the file, and optionally banking angles of the left and right track. Unit is [m] and [deg], respectively.
`channel_left_track`	-	column index of left track data
`channel_right_track`	-	column index of right track data. May coincide with `channel_left_track`, if only one track has been measured and/or stored
`channel_left_banking_angle`	-	column index of the banking angles of left track (given in deg). If this index is not specified, the banking angles of left track is set to 0
`channel_right_banking_angle`	-	column index of the banking angles of the right track (given in deg). May coincide with `channel_left_banking_angle`, if only one track has been measured and/or stored. If this index is not specified, the banking angles of right track is set to 0
`meas_distance`	mm	distance between two consecutive measured points in longitudinal direction. Travel distances outside of the data range, as specified in the file, are assigned a road height by means of extrapolation
`track_linked_to course`	0/1	flag whether the 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 a vehicle will stop on the road, due to numerical reasons `track_linked_to_course = 0` 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 a function of x, y, t and optional further *cosin/io* parameters p: $z = f(x, y, t, p)$. This function may be an arbitrary arithmetic expression as documented in **cosin/io**, including 1D and 2D lookup tables, random values, etc.
dx	mm	road surface displacement in x direction, specified by a function as described for z
dy	mm	road surface displacement in y direction, specified by a function as described for z
vx	m/s	road surface displacement velocity in x direction, specified by a function as described for z. If vx is specified , dx is redundant and ignored
vy	m/s	road surface displacement velocity in y direction, specified by a function as described for z. If vy is specified , dy is redundant and ignored
mu	-	road friction factor, specified by a function as described for z

2.6 cosin/io 2D Road Type `hydropulse_harmonic`

cosin/road looks for the 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 the front left / front right / rear left / rear right wheel
amplitude_fl amplitude_fr amplitude_rl amplitude_rr	mm	hydro-pulse excitation amplitudes at the 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 the 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 to the road type `stochastic_uneven`, cf. below. **cosin/road** looks for the data block `$hydropulse_noise`, and reads:

Name of input variable	Unit	Meaning

`intensity_fl`
`intensity_fr`
`intensity_rl`
`intensity_rr`
	-	factor to control the intensity of the '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 the 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 the 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 the front left / front right / rear left / rear right wheel. In addition to the special constants available in all **cosin/io**-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 the data block `$plank` or `$cleat`, and reads:

Name of input variable	Unit	Meaning
height	mm	height of the plank/cleat
start	m	start of the plank/cleat (travel distance)
pos_center	length	x-component of position of plank/cleat center. If set, this value overrides item start
width	mm	width of the plank/cleat, measured in the transversal direction of the plank/cleat (formerly called length)
length_x	mm	length of the plank/cleat, measured in the rolling direction. Specification of length_x will overwrite width. Both differ if the direction is non-zero
cleat_angle	angle	direction of the cleat relative to the y-axis. Direction angle is measured in counter-clockwise direction, if looking from above onto the road surface. **NOTE**: in releases prior to 2018-3 a data item direction has been used which measured the direction angle in clockwise direction. For compatibility reasons, this data item is still accepted but considered deprecated
lat_shift	mm	lateral shift of the cleat (formerly called lateral_shift)
bevel_edge_length	mm	length of the bevel edge of the cleat, measured in circumferential direction of the drum. Bevel edge has a 45deg slope
edge_rounded	0/1/-1	1: bevel edge rounded, 0: bevel edge linear, not rounded, -1: bevel edge concavely rounded
mu_factor_cleat	-	friction value factor of the plank/cleat

2.10 cosin/io 2D Road Type `pot_hole`

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

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

2.11 cosin/io 2D Road Type `ramp`

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

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

2.12 cosin/io 2D Road Type `roof`

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

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

2.13 cosin/io 2D Road Type `sine`

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

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

2.14 cosin/io 2D Road Type `sine_sweep`

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

linear sweep: the frequency increases linearly with respect to the travel distance. The road height value z(s) as a 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 the 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 the 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:

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

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

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

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

Name of input variable	Unit	Meaning
`start`	m	start of the swept sine wave (travel distance)
`end`	m	end of the swept sine wave (travel distance)
`amplitude_at_start`	mm	amplitude of the swept sine wave at the start
`apmlitude_at_end`	mm	amplitude of the swept sine wave at the end
`wave_length_at_start`	m	wave length of the swept sine wave at the start
`wave_length_at_end`	m	wave length of the swept sine wave at the end. Must be less or equal to `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 the data block `$road_type`, **cosin/road** reads:

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

After having read the 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 the travel distance in m, and in the second is the road height in mm. For the syntax of the spline data definitions, cf. cosin/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 the 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, that are 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 spectral density (PSD) of this road surface profile is given by:

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

where:

$$S \gg 1\,m$$

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

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 a '**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 $\Omega_0 = 1\,rad/m$, is used in ISO 8608 to classify road surface profiles. These classifications are listed in the following table:

Road class	$G_d\left(\Omega_0\right)\ \left[10^{-6}m^3\right]$		
	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 so that these two signals are completely independent, if `correlation_rl = 0.0`, and identical, if `correlation_rl = 1.0`:

$$z_l\left(s\right) = z_1\left(s\right) + \frac{corr_{rl}}{2}\left(z_2\left(s\right) - z_1\left(s\right)\right)$$

$$z_r(s) = z_2(s) + \frac{corr_{rl}}{2}\left(z_2(s) - z_1(s)\right)$$

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

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

2.17 cosin/io 2D Road Type `tilt_table`

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

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

3 Road Profiles and Obstacles in TeimOrbit Format

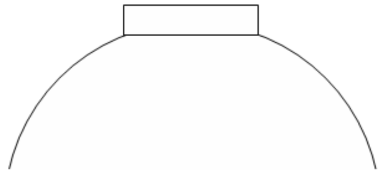
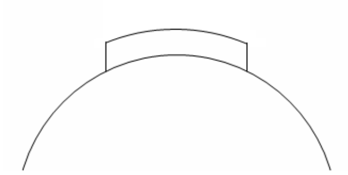
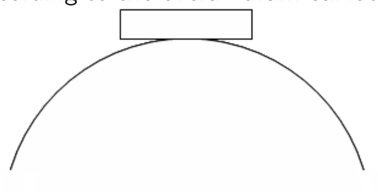
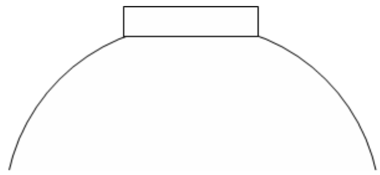
If **cosin/road** has recognized that the road data file is in TeimOrbit format (typically, the file name extension is 'rdf'), it will branch into the respective reading and evaluation routines.

All available road types in this format are described below. Please refer to the original documentation for TeimOrbit syntax rules, as well as for meaning and contents of all other data sections which are 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 the section `[PARAMETERS]`:

Name of input variable	Unit	Meaning
`diameter`	length	diameter of the tire test drum. **cosin/road** assumes a outer drum, if the `diameter < 0`
`wheel_placement`	-	0 above drum (default) 1 before drum 2 on the left of drum 3 below drum 4 behind drum 5 on the right of drum
`x_coord_drum_center`	length	x coordinate of the drum center in the global coordinate system, 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 the drum center in the global coordinate system, used for all wheels
`x_coord_drum_center_wheel_i` $i = 1, 2, 3, \ldots, 25$	length	x coordinate of the drum center in the global coordinate system, used only for wheel i $i = 1, 2, 3, \ldots, 25$
`y_coord_drum_center_wheel_i` $i = 1, 2, 3, \ldots, 25$	length	y coordinate of the drum center in the global coordinate system, used only for wheel i $i = 1, 2, 3, \ldots, 25$
`number_cleats`	-	number of extra cleats on the drum (`number_cleats = 0` is allowed)
`cleat_starting_angle`	angle	drum angle coordinate of the 1^{st} cleat, used for all wheels
`cleat_starting_angle_wheel_i` $i = 1, 2, 3, \ldots, 25$	angle	drum angle coordinate of the 1^{st} cleat, used only for wheel i $i = 1, 2, 3, \ldots, 25$
`cleat_angle`	angle	direction of the cleat relative to the drum spin axis. Direction is zero if the cleat is oriented exactly in the transversal direction (which is the default case). The direction angle is measured counter-clockwise if looking from above onto the drum surface. **Default value** is 0 (transversal cleat) **NOTE**: for releases prior to 2018-3 the cleat direction for `diameter` $<$ **0** has been measured in clockwise direction and for `diameter` $>$ **0** in counter-clockwise direction, using the deprecated item `cleat_direction`

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

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

Either:

v	length/time	constant surface speed of the drum
acceleration_time	time	time span at the beginning of the dynamic simulation, to accelerate the drum from stand-still to the 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 the speed-vs.-time relationship 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 the cleats
cleat_width	length	width of the cleats, measured in transversal direction of the cleat (formerly called cleat_length)
cleat_length_x	length	length of the cleats, measured in circumferential direction of the drum. Specification of cleat_length_x will overwrite cleat_width. Both differ if cleat_angle is non-zero

or:

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

or:

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

Name of input variable	Unit	Meaning
number_cleats	-	number of the extra cleats on the belt (number_cleats = 0 is allowed)
cleat_angle	angle	direction of the cleat relative to the belt lateral direction. Direction is zero if the cleat is orientated exactly in the transversal direction (which is the default case). Direction angle is measured in counter-clockwise direction, 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 speed of the belt surface
acceleration_time	time	time span at beginning of the dynamic simulation, to accelerate the belt from stand-still to the 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 the speed-vs.-time relationship of the belt'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 the cleats
cleat_width	length	width of the cleats, measured in transversal direction of the cleat (formerly called cleat_length)
cleat_length_x	length	length of the cleats, measured in circumferential direction of the drum. Specification of cleat_length_x will overwrite cleat_width. Both differ if cleat_angle is non-zero
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 the x/z relationship 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 the x/z relationship of the cleat's cross section, together with the rounding radii for 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 the x/z relationship 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 servo-hydraulic 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 the section [PARAMETERS]:

Name of input variable	Unit	Meaning
sampling_time	time	assumed time increment of the wheel support displacement values, if no separate time channel is provided
flat_belt_velocity	length/time	horizontal velocity of an additional, optional flat belt device **Default value** is 0 (no additional flat belt)
acceleration_time	time	time span at the beginning of the dynamic simulation, to accelerate the flat belt from stand-still to the 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 the 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 the x/y/z relationship of the time-dependent spatial wheel support displacement. Time values are assumed to be equidistant, with a increment defined by sampling_time

or:

data sub-section (TZ_DATA)	arbitrarily many lines with 2 values each of type time, length	data defining the t/z relationship 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 the t/x/y/z relationship of the time-dependent 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 the section `[PARAMETERS]`:

Name of input variable	Unit	Meaning
`height`	length	height of the plank/cleat
`start`	length	start of the plank/cleat (travel distance)
`pos_center`	length	x-component of position of plank/cleat center. If set, this value overrides item `start`
`width`	length	width of the plank/cleat, measured in transversal direction of the plank/cleat (formerly called `length`)
`length_x`	length	length of the plank/cleat, measured in the rolling direction. Specification of `length_x` will overwrite `width`. Both differ if `direction` is non-zero
`cleat_angle`	angle	direction of the cleat relative to the y-axis. Direction angle is measured in counter-clockwise direction, if looking from above onto the road surface. **NOTE**: in releases prior to 2018-3 a data item `direction` has been used which measured the direction angle in clockwise direction. For compatibility reasons, this data item is still accepted but considered deprecated
`lateral_shift`	mm	lateral shift of the cleat
`bevel_edge_length`	mm	length of the bevel edge of the cleat, measured in circumferential direction of the drum. Bevel edge has a 45deg slope
`edge_rounded`	0/1/-1	1: bevel edge rounded, 0: bevel edge linear, not rounded, -1: bevel edge concavely rounded
`mu_factor_cleat`	-	friction value factor of the 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 the section [PARAMETERS]:

Name of input variable	Unit	Meaning
depth	length	depth of the pot-hole
start	length	start of the pot-hole (travel distance)
length	length	length of the 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 the section [PARAMETERS]:

Name of input variable	Unit	Meaning
height	length	height of the ramp
start	length	start of the ramp (travel distance)
slope	length	slope of the 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 the 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 the section [PARAMETERS]:

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

3.9 TeimOrbit 2D Road Type sine_sweep

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

linear sweep: the frequency increases linearly with respect to the 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 the 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 the section `[PARAMETERS]`:

Name of input variable	Unit	Meaning
`start`	length	start of the swept sine wave (travel distance)
`end`	length	end of the swept sine wave (travel distance)
`amplitude_at_start`	length	amplitude of the swept sine wave at the start
`amplitude_at_end`	length	amplitude of the swept sine wave at the end
`wave_length_at_start`	length	wave length of the swept sine wave at the start
`wave_length_at_end`	length	wave length of the swept sine wave at the end. Must be less or equal to `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 for both 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.

These values are then 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 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 a '**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 $\Omega_0 = 1\,rad/m$, is used in ISO 8608 to classify road surface profiles. These classifications are listed in 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 so that these two signals are completely independent, if `correlation_rl = 0.0`, and identical, if `correlation_rl = 1.0`:

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

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

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 the 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 the table above. Mean $G_d(\Omega_0)$ value of the respective class will be used
`path_constant`		m	'path constant' S to control the high-pass integration filter cut-off frequency in path domain. Parameter is optional, default value is 1000 m
`correlation_rl`		-	variable to control the correlation between the 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 the section `[PARAMETERS]`:

Name of input variable	Unit	Meaning
`y_coord_rot_axis`	mm	y coordinate of the tilt-table's rotation axis (which is assumed to be parallel to the global x-axis)
`start_time`	s	start time of the tilt table's rotation
`end_time`	s	end time of the tilt table's rotation (tilt table continues rotating throughout simulation if not set)
`angular_velocity`	deg/s	tilt table's angular velocity about the x-axis; assumed to be constant during the 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 the 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 a 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 onto the 2D data

4 Regular Grid Road Data Files (RGR Files)

This chapter describes the format of the **R**egular **G**rid **R**oad Data files (RGR-files). RGR-files can be used to describe a 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 is called regular grid data.

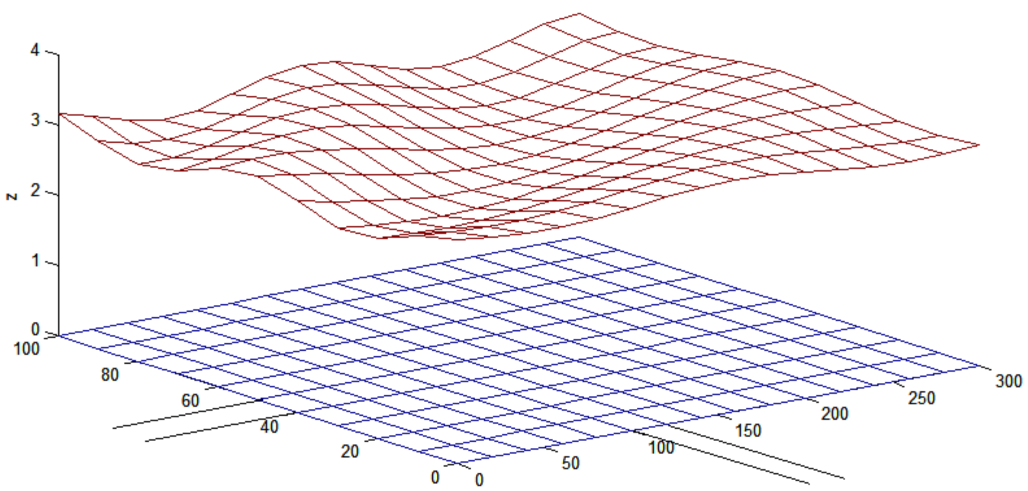


Figure 1: Regular Grid Geometry

Regular grid core data are defined by

- the grid's minimum x-value: x_{min}
- the grid's constant mesh-size in x-direction: $\triangle x$
- 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: $\triangle y$
- 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 $x_i = x_{min} + (i-1)\triangle x$ and $y_k = y_{min} + (k-1)\triangle y$.

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:

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

Default length unit of all x, y, and z-values is millimeter [mm], if not specified differently in the data file header (see below).

RGR-files may also contain curved center-line data, location-dependent variable friction values, soft soil data (see below), and more. Please refer to *cosin/tools for Roads* for more.

4.2 Regular Grid Data File Format

cosin/tools for Roads provides extensive support in generation, modification, and analysis of RGR-files. A detailed description of the losslessly compressed binary file format is available for certified development partners.

4.3 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, both a simple **place-holder model** as well as the so-called **Bekker-Wong model** are implemented. For the foundations of the Bekker-Wong model, please refer to Wong's textbook (*Terramechanics and Off-Road Vehicle Engineering, Butterworth-Heinemann, Amsterdam 2010*). For details on how to select the model, please refer to the*cosin/tools for Roads* documentation.

The **place-holder 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: $z_o(x, y)$ 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:

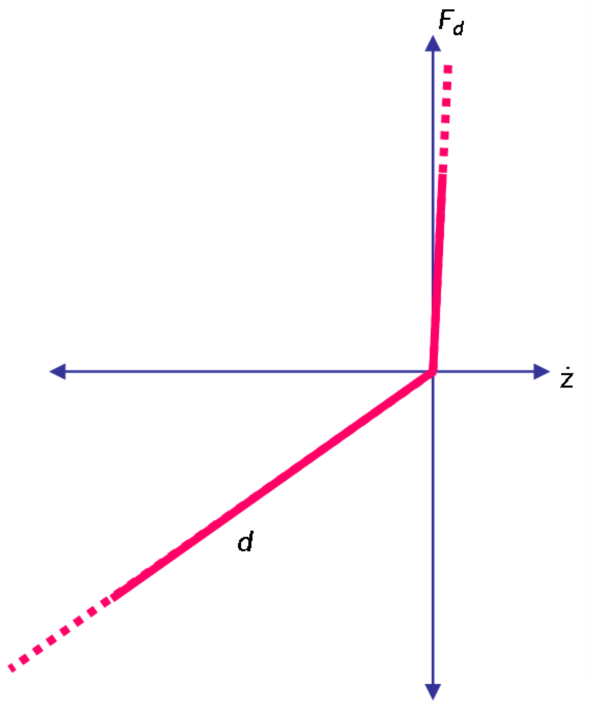


Figure 2: Non-Linear Damping Characteristic in Soft-Soil Placeholder Model

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 optional additional parameter `max_sim_vel` (maximum sinkage velocity) limits the local increase in sinkage per time unit, in order to prevent unrealistic deep sinkage due to short-term peak forces. Its effect is similar to a nonlinear sinkage damping. The physical unit is [length unit]/s, default value, and reasonable for most applications, is 2 m/s.

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 is 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 `ymax` 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.

6 User-Defined Road Models

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

- **urm.dll** in Windows, and
- **liburm.so** (or **liburm.**dylib, respectively) in Linux and OS X.

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 are as follows:

Parameter	C/C++ type	Data flow	Unit	Meaning
`ti`	`int`	in	-	wheel index. Typically: 1=fl, 2=fr, 3=rl, 4=rr, ..

Parameter	C/C++ type	Data flow	Unit	Meaning
`t`	`double`	in	s	simulation time, provided by **cosin/road**
`x`	`double`	in	m	x-comp. of location where the road height is needed; provided by **cosin/road**. If road is moved with cti function ctiSetRoadMotionData or similar, position is relative to this moving coordinate system
`y`	`double`	in	m	y-comp. of location where the road height is needed; provided by **cosin/road**. If road is moved with cti function ctiSetRoadMotionData or similar, position is relative to this moving coordinate system
`z`	`double*`	out	m	road height. If road is moved with cti function ctiSetRoadMotionData or similar, height is relative to this moving coordinate system
`vx`	`double*`	out	m/s	x-comp. of road surface velocity relative to the global coordinate system (non-zero for example in drum or flat-belt simulations). If road is moved with cti function ctiSetRoadMotionData or similar, velocity is relative to this moving coordinate system
`vy`	`double*`	out	m/s	y-comp. of road surface velocity relative to the global coordinate system. If road is moved with cti function ctiSetRoadMotionData or similar, velocity is relative to this moving coordinate system
`vz`	`double*`	out	m/s	z-comp. of road surface velocity relative to the global coordinate system (non-zero for example in hydraulic 4-poster simulations). If road is moved with cti function ctiSetRoadMotionData or similar, velocity is relative to this moving coordinate system
`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

Parameter	C/C++ type	Data flow	Unit	Meaning
`file`	`char*`	in	string	name of road data file (provided by **cosin/road**). 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 **every** call to urm. However, typically, it needs to be read only during first call of urm with the respective wheel index. urm 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 **Interfaces/user_code** of the **cosin** installation directory:

```
/* 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;
}
```

Index