UM Configuration and Structure

2020

Contents

1. Universal Mechanism configuration and structure

1.1. Configuration of program package UM

Universal Mechanism software consists of a standard kernel and UM object. The kernel includes executable programs and auxiliary files. UM object is the data of the multibody system computer models. UM object can be located independently from the UM kernel. The kernel may be installed on a server whereas objects – on the user's computers. Working with UM kernel involves using **UM Input** and **UM Simulation**.

UM Input:

- Description of objects;
- Generation of equations of motion;
- Compilation of equations by an external compiler;

UM Simulation:

- Simulation of the object dynamics.

1.1.1. UM location

The UM kernel is located in several standard directories. Executable files by default are located in *C:\Program Files\UM Software Lab\Universal Mechanism\9.0* directory. For 64-bit version of OS Windows default directory is *C:\Program Files (x86)\UM Software Lab\Universal Mechanism\9.0*. At program installation it is possible to specify independently the location of directory for placing executable files. Further a path to this directory we will designate in abbreviated form as **{UM}**.

Auxiliary files will be always placed in the directory for public documents in OS Windows. In Windows 7 and later this directory are located in *C:\Users\Public\Documents\UM Software Lab\Universal Mechanism\9.0*, in Windows Vista / XP this directory are located in *C:\Documents and Settings\All Users\Documents\UM Software Lab\Universal Mechanism\9.0*. Further a path to this directory we will designate in abbreviated form as **{UM Data}**.

Standard directories, sub-directories and files from them cannot be renamed or (re-)moved.

The short description of contents of subdirectories of the program package:

- {UM}\bin includes executable files of UM;
- {UM}\bin\drivers is the folder with electronic key drivers (for protected versions);
- {UM Data}\com includes files that are necessary for compilation of equations and programming in the UM environment;
- {UM Data}\components is the folder with the set of components;
- {UM Data}\MANUAL is the folder with the UM user's manual;
- {UM Data}\plugins if the folder with dynamic libraries with functionals;
- {UM Data}\SAMPLES is the folder with sample models;
- {UM Data}\SAMPLES\TUTORIAL includes tasks for manual parts "Getting started";

- {UM Data}\SAMPLES\LIBRARY includes examples of simulation illustrating features of application of various elements of a program package;
- {UM Data}\simulink is the folder with files necessary for integration of Matlab/Simulink models into UM;
- {UM Data}\templates is the folder with templates for import of data from UM into MS Excel.

1.1.2. UM Object data organization

Every object (model) created by UM is situated in a separate directory. *The name of the directory is used as the object name*. There exist syntactical restrictions on the name: it can contain Latin letters, digits and a '_' character. The first symbol of the name must be a letter. If the object directory is renamed manually, a user must once again generate and compile equations.

1.1.3. System parameters

Universal Mechanism uses a point as a decimal separator and a space as a group separator. If current settings are different then it will be fixed automatically. Changes to default settings can be done through Windows Control Panel.

1.1.4. Portability of models

Three files are of importance in this context, *input.dat*, *control file* and *user generated UM program file*. These contain the description of object parameters. If a user does not use a Control File as well as user created files for programming within the UM environment, *the input.dat file* is the only file that contains information about the model as a mechanical system. If a user creates a program file within UM environment, then it is necessary to keep the *input.dat*, *control file* and *user's files used in the control file* (refer Chapter 6).

Thus, depending on which of the above two contexts applies, these are the only files essential to transfer a model to another computer, to send the model by e-mail and so on. All other files, which are necessary for simulation of a model, can be generated by UM.

To transfer a model to another computer, the steps are

- create a directory with the new name of the model;
- copy the *input.dat* and (if necessary) the control file and user's source files;
- generate and compile equations of motion for this new object;
- execute simulation of the object .

Note. Some identifiers and units in the equations of motion of the object include the name of the object. If the object directory is renamed, it is necessary to either rewrite the old control file and (if necessary) transfer user's procedures into the new version of this file, or correct the corresponding identifiers and unit names in the old control file and rename it.

1.1.5. UM structure

UM consists of two main programs located in the *{UM}\bin* directory.

- **Data input program (UM Input, uminput.exe)**

Generally it is a pre-processor that is used for description and modification of UM models and (optionally) to generate equations of motion. It has a multi-project environment: the user can open and modify several models simultaneously.

- **Simulation program (UM Simulation, umsimul.exe)**

It includes both solver and post-processor – two last elements of the typical CAE triad (1) pre-processor – (2) solver – (3) post-processor. This program is intended for the following issues: numeric integration of the equations of motion with simultaneous animation of the object motion and plotting of graphs; linear analysis of the object equations; multi-variant projects and parametric optimization and other functions. The program imports equations of motion created by the input program in the *umtask.dll* file.

The simulation program can be used separately from the input program for simulation of previously prepared models. The simulation program has a single project environment.

1.1.6. List of UM modules

Universal Mechanism includes the following modules.

- **UM Base** is the minimal version, includes the pre- and postprocessors. UM Base includes the following additional tools.

  UM Base / Control panel is an additional tool for interactive model control.
  - **UM Base / Control panel** is an additional tool for interactive model control.
  - **UM Base / Training ground** is an additional tool for creating virtual training grounds that is used for tracked vehicles and robots.
  - **UM Base / Ride comfort** is an additional functionals for ride comfort evaluating according to the UIC 513, Sperling index and the Russian (exUSSR) OST 24.050.16-85 and GOST 55513 – 2013.
- **UM Subsystems** is an additional module which allows describing large objects as subsystems.
- **UM Automotive** is an additional module to simulate vehicle road dynamics.
- **UM Tracked Vehicle** is an additional module for simulation of tracked vehicle dynamics.
- **UM Loco** is an additional module for simulation of a railway vehicle dynamics.
  - **UM Loco / Multipoint Contact Model** is the accurate multi-point non-elliptical wheel-rail contact model. Rail is considered an inertial element.
  - **UM Loco / CONTACT add-on interface** is the interface for CONTACT program. The CONTACT program is provided and supported by Vtech CMCC (https://www.cmcc.nl/software/).
  - **UM Loco / Wheel Profile Wear Evolution** is an additional tool for prediction of railway wheel profile evolution due to wear.

    - **UM Loco / Rail Profile Wear Evolution** is an additional tool for prediction of railway rail profile evolution due to wear.
- **UM Monorail Train** is the module for simulation of monorail train dynamics taking into account the track flexibility.
- **UM MagLev** is an additional module for simulation of EMS and EDS maglev vehicles taking into account the track flexibility.
- **UM Experiments** is an additional module intended for parametrical scanning of dynamical behavior and optimization.
- **UM Cluster** is the distributed calculations service. This service extends UM Experiments module to be used in a local or global computer network cluster for parallel numerical experiments.
- **UM FEM** is an additional module for including elastic bodies into models.
- **UM Control** is an additional module that provides Matlab/Simulink and user-defined routines interface for including control, electrodynamics, hydraulics etc. into UM models. The module includes the following additional tools.
    - **User-defined routines** provides import into UM model dynamic-loaded libraries (DLL) compiled with the help of any compiler. DLLs describe mathematical models of forces or control.
    - **Matlab Import** provides export Matlab/Simulink models to DLL and import them into UM models. Host application is UM.
    - **Matlab CoSimulation** provides import UM models as S-Functions into Matlab/Simulink models. Host application is Matlab/Simulink.
    - **Block Editor** is a separate additional tool, which serves for the description of structure diagrams with the help of basic functional elements. In fact, the Block Editor is very somehow similar to the Simulink tool from the Matlab/Simulink software package.
- **UM CAD Interfaces** provides import graphical and inertia data from SolidWorks, Autodesk Inventor, Unigraphics NX, Kompas-3D, Pro/E.
- **UM Train** is an additional module for simulation of a train longitudinal dynamics.
- **UM Train 3D** is an additional module to include a train 3D model into railway vehicle models from UM Loco.
- **UM Ballast** is an module for simulation of dynamics of granular media systems in 2D.
- **UM Durability** is an additional module for S-N fatigue analysis.
- **UM 3D Contact** is an additional module for simulation of contact interaction in 3D.
- **UM COM Server** provides simulation of UM models. It is used, for example, for supporting simulation of models under Matlab/Simulink environment.
- **UM Pneumatic Systems** allows simulation of multibody systems with pneumatic elements such as isolated and interconnected air springs.
- **UM Driveline** is a collection of force elements for modeling transmissions.
- **UM Flexible Railway Track** is an additional module for simulation of interaction of railway vehicles and flexible track.
- **UM RCF** is an additional module for simulation of the process of accumulation of rolling contact fatigue (RCF) damage in railway wheels.

- **UM Scene** is an additional module for creating advanced photorealistic scenes and environment for UM models.
- **UM Quick Track** is an additional module for automatic, quick and simple creating good-looking tracks for railway vehicles and trains. It is going to be extended for road and monorail vehicles as well.
- **UM Video Flow** is an additional module for simulating camera sensors and exporting the video flow into Matlab for using as video input for computer vision algorithms.
- **UM Sensors** is an additional module for simulation typical sensors used in automotive industry (including ADAS systems) and robotics: GPS sensors, ray sensors, beacon/receiver sensors.

1.1.7. Deprecated UM modules

The following UM modules and tools are removed and no longer available, distributed and supported in UM 9.

- **UM Wheel / Rail Wear** module. Starting with UM 9 it is replaced with the pair of **UM Loco / Wheel Profile Wear Evolution** and **UM Loco / Rail Profile Wear Evolution** tools.
- **UM Loco / Non-elliptical wheel/rail contact model** tool. It is totally replaced by the **Multipoint contact model** (or **Kik-Piotrowski model** as it is written in UM. The **Multipoint contact model** is more stable and accurate.
- **UM Loco / External DLLs for creep force calculation** tool. It was based on the **UM Loco / Non-elliptical wheel/rail contact model** techniques and is no longer supported neither.

1.1.8. Hardware requirements

UM runs on computers with Windows XP / Vista / 7 / 10. Recommended system requirements are following:

- Pentium PC, 4 GHz (i7 core / multi-core processors are strongly recommended);
- 4 Gb RAM memory;
- Solid State Drive;
- up-to-date/professional video card is strongly recommended in the case of working with bodies imported from CAD or FEM software, NVidia video cards are preferable vs. ATI Radeon and Intel for UM graphics.

1.2. Command line parameters

UM Input and **UM Simulation** programs can be executed through command lines. This provides the opportunity to automatically run the programs from within an external program (e.g. Maple or a user's program).

1.2.1. Command line parameters for UM Input program

The user may generate the object data file *"input.dat"* with the help of an external program (for example, MAPLE, MATLAB, MATEMATICA). In this case the equations of motion can be generated and/or compiled by **UM Input** program in an automatic command line mode.

Command line syntax and options:

"[path to uminput.exe]\uminput.exe" [path to object] [options]

Options	Comments
/g	Generate equations of motion. If the object contains external subsystems, completeness of description and generation of equations for each the subsystem is verified. If equations are not prepared, generation is fulfilled automatically.
/b	Forced generation of equation
/c	Compile equations of motion
/n	Rewrite old control file. This option is valid if one of the options /c/b/g is present.
/i	Display generation messages in a separate window. If the option is not present the UMINPUT is executed in full console mode without visualization of messages. This option is valid if one of the options /c/b/g is present.
/f[path to file]	All messages are directed into the file. This option is valid if one of the options /c/b/g is present.

Remark. Any one of the options /c/b/g switches on the console mode.

Examples

"D:\um\bin\uminput.exe" *"d:\My Objects\UMObject"*

Run the **UM Input** program in the usual mode and open the object UM Object located in the d:\My Objects directory.

"D:\um\bin\uminput.exe" "d:\Objects\UMObject" /g /c /i "/f d:\Objects\UMObject\outinfo.txt"

Run the **UM Input** program in console mode. Generate and compile equations of motion. Generation messages are displayed in the window. Program messages are stored in the file outinfo.txt.

1.2.2. Command line parameters for UM Simulation program

The option allows running **UM Simulation** program in an automatic mode to solve object equations. The command line points to files, containing initial conditions, identifier values, configurations and so on. Unlike the usual mode, the simulation process starts immediately after program execution. Simulation results can be stored in user's defined files. The simulation program closes right after the simulation is finished or on receiving an interruption command.

The automatic mode allows the user to organize the simulation and results processing with the help of the external program.

Command line syntax:

"[path to UmSimul.Exe]\UmSimul.exe" [path to object] [options]

Options	Comments
/s	Sets the automatic mode
/c[Fileame].icf	Name of configuration file (program desktop, parameters of solver)
/p[Filename].par	File with values of identifiers
/r[Filename].rwc	Rail vehicle configuration file (wheel/rail profiles, track profiles etc.)
/t[Filename].train	Train configuration file
/a[Filename].car	Road vehicle configuration file
/o[Filename].mrt	Monorail vehicle configuration file
/u[Filename].tvc	Tracked vehicle configuration file
/i[Filename].xv	File with initial conditions (values of coordinates and their time derivatives at t=0)
/j[Filename]	File with RCF parameters
/v[Filename].var	Name of file with automatically calculated variables
/x[Filename].xva	Switches on the XVA mode. Data will be saved in the specified file.
/e[Filename].fin	Finish conditions configuration file. The specified file of finish conditions will be load from the model directory and applied for the current numerical experiment.
/b	This flag turns on so called *Batch* mode of **UM Simulation** program. No main window or any other windows appear in this mode. *Batch* mode suits for hidden running umsimul.exe as an external solver under within problem-specific pre- and post-processors.
/f[Filename.ext]	Redirect all error and warning messages to the given text file. Errors are marked with the '[Error]' sentences. Warnings are marked with '[Warning]' sentences. Example: */fumoutput.txt* means that all error and warning messag-

	es will be redirected to *umoutput.txt* file that is located in the model directory.
/h[HWND]	Handle (HWND) of a window to send back the current progress of the numerical experiment from umsimul.exe to the calling application. The handle should be given in decimal, not HEX format. This command can be used when umsimul.exe is called from third party pre- or post-processors as an external solver where HWND is the handle of the main window of the third party application that runs umsimul.exe and is received windows messages from umsimul.exe with the current progress. Umsimul.exe sends the Windows message to the HWND specified in the command line parameter every time the current progress is changed. Message number is **WM_USER+1.** Message parameters are as follows. *Message.WParam* is *ProcessID* of run umsimul.exe. If you run several umsimul.exe simultaneously this ProcessID will help you to distinguish one from another. *Message.LParam* is the current progress of the numerical experiment in percent. *Message.LParam* is integer value and is in between the following range: [0..100].

Remark. Files *.icf, *.xv, *.rwc, *.car, *.train, *.mrt, *.tvc , *.par, *.var should be created beforehand and saved in the directory of the object. If one of the files is not present in the command line, the corresponding last.* file is used (if it exists).

Examples

"D:\um\bin\umsimul.exe" "d:\My Objects\UMObject"

Run the **UM Simulation** program in a usual mode and load the model *UM Object*, located in the "d:\My Objects" directory.

"D:\um\bin\umsimul.exe" "d:\My Objects\UMObject" /s /csome.icf /vtest.var /xtest.xva

Run the **UM Simulation** program in automatic mode. Read: the configuration from the file *some.icf*; the list of automatic calculated variables – from the file *test.var*. Switch on the XVA mode and save data in the file *test.xva* (the file is stored in the object directory). The files *test.var*, *some.icf* should be created beforehand and located in the object directory *"d:\My Objects\UMObject"*.