Motion Simulation Manual

Table of Contents

[1. Requirements\Installation 4](#_Toc335646034)

[1.1. Requirements 4](#_Toc335646035)

[1.2. Installation 4](#_Toc335646036)

[2. MotionSim Program 5](#_Toc335646037)

[2.1. Layout 5](#_Toc335646039)

[2.2. Starting MotionSim 5](#_Toc335646040)

[3. Graphical User Interfaces (GUI) 6](#_Toc335646041)

[3.1. Main Menu Overview 6](#_Toc335646043)

[3.1.1. Loading Models 7](#_Toc335646048)

[3.1.2. Creating Models 7](#_Toc335646049)

[3.1.3. Simulation and Analysis 8](#_Toc335646050)

[3.2. Preprocess Menus 8](#_Toc335646051)

[3.2.1. Simulation Parameters 9](#_Toc335646053)

[3.2.2. Body Data 10](#_Toc335646054)

[3.2.3. Constraint Data 11](#_Toc335646055)

[3.2.4. Driving Constraints 12](#_Toc335646056)

[3.2.5. Spring/Dampers/Actuators 13](#_Toc335646057)

[3.2.6. Solver Options 14](#_Toc335646058)

[3.2.7. Simulate Button 15](#_Toc335646059)

[3.3. Analysis Menus 15](#_Toc335646060)

[3.3.1. Loading\Saving 15](#_Toc335646062)

[3.3.2. Plotting Analysis 16](#_Toc335646063)

[3.3.3. Animating Models. 17](#_Toc335646064)

[3.3.4. Frequency Analysis 19](#_Toc335646065)

[3.3.5. External Points Analysis 20](#_Toc335646066)

[4. Running Simulations 21](#_Toc335646067)

[5. Creating Models Through Excel 22](#_Toc335646068)

[5.1. Introduction 22](#_Toc335646071)

[5.2. Color Coding 22](#_Toc335646072)

[5.3. Simulation Info Sheet 23](#_Toc335646073)

[3.3.6. Simulation Type 23](#_Toc335646074)

[5.4. Body Sheet 24](#_Toc335646075)

[5.5. Constraint Sheet 25](#_Toc335646076)

[5.5.1. 2d CONSTRAINTS 26](#_Toc335646084)

[5.5.2. 3d CONSTRAINTS 27](#_Toc335646085)

[5.6. Driving/Simple Constraint Sheet 29](#_Toc335646086)

[5.6.1. Driving Constraints 29](#_Toc335646088)

[5.6.2. Simple Constraints 30](#_Toc335646089)

[5.7. SDA Sheet 30](#_Toc335646090)

[5.8. Post 31](#_Toc335646091)

[6. Simulink Integration 32](#_Toc335646092)

[6.1. MotionSim Libraries in Simulink 32](#_Toc335646094)

[6.2. Creating a Simulink Model with a Dynamic System 33](#_Toc335646095)

[6.3. Simulink Examples 34](#_Toc335646096)

[7. Program Detail 35](#_Toc335646097)

[7.1. How Models are solved 35](#_Toc335646099)

[7.1.1. KINEMATIC 36](#_Toc335646103)

[7.1.2. DYNAMIC 37](#_Toc335646104)

[7.2. File List 37](#_Toc335646105)

[7.3. Variable Structure 37](#_Toc335646106)

# Requirements\Installation

## Requirements

Software required:

* Matlab 2011 or newer (older version may work)
* Excel 2007 or newer (2003 and older not compatible)

The MotionSim program has been tested and verified to work on windows platforms only. While Matlab does exist for the Linux and Macintosh platforms, full functionality cannot be certain. Like with all numerical simulations, usability and speed will depend on the target hardware. Small simulations should be fine on most hardware; larger more accurate simulations will depend largely on memory size and processer speed.

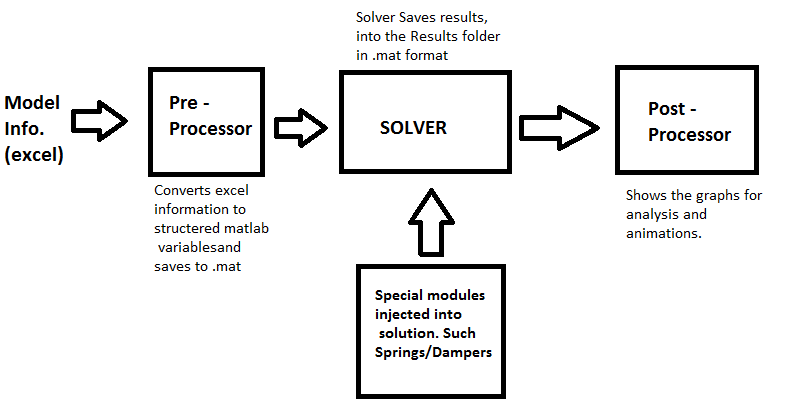
## Installation

To Install, simply unzip the MotionSim file into an directory with read and write permissions. All folders and files needed to run the MotionSim program should be extracted and ready to use. Simply start Matlab and follow the instructions on using the MotionSim program.

# MotionSim Program



## Layout



The above figure represents the typical process for simulation of multibody systems in MotionSim. The figure represents the 3 major phases in the program; Loading system parameters, solving for the system and processing the data for analysis.

## Starting MotionSim

To start the MotionSim program, follow the directions below in order:

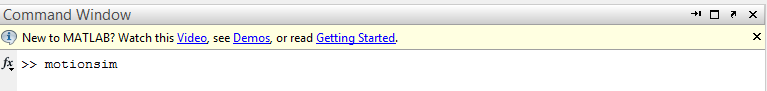
* Start Matlab



* Change the current folder to the MotionSim Main directory



* Type “motionsim” in the command window and hit enter

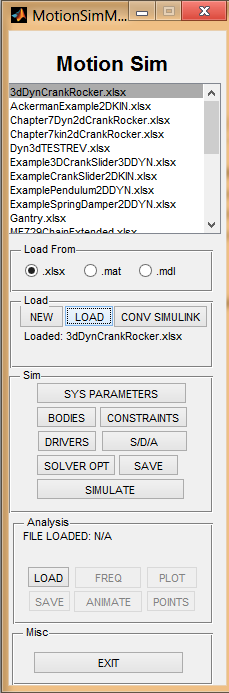


# Graphical User Interfaces (GUI)



## Main Menu Overview

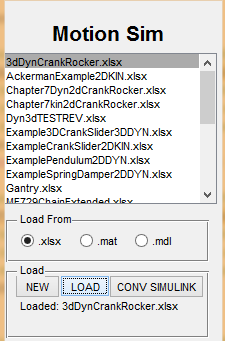
The MainMenu is the menu that appears after typing “motionsim” into the command window as detailed in the previous section. The MainMenu appears as below and contains all the panels to create, modify and analyze a model.



MotionSim’s Main Menu



### Loading Models



Main menu showing the controls to load a model

A preconfigured model’s initial coordinates, constraints and dynamic data can be loaded into memory from two different file formats.

* From an excel file (.xlsx)
* From a Matlab MAT File (.mat)

This is accomplished through the main menu GUI, where the user can select between .xlsx and .mat files. The program scans the “Models\” directory and will display the files in the list box with the corresponding file types.

The user then selects the “**Load**” button to load the model data into memory. Depending on the file type the program will decide how to load the data into memory. In the case of a MAT file, the model data will be loaded directly. For model data created in Excel format, the program will scan the excel file and arrange the data into the standard variables used for computation.

After the model information is loaded into the memory the user is free to modify, simulate and analyze the model itself.

### Creating Models

This section outlines the two ways to input data and create a file that represents a system of interest. The previous section on loading models details the methods to load the models into the program. For those interesting in Simulink models please see the Simulink Integration section in later chapters.

**MAT Models** – The MAT file is a native Matlab format used to store matrices, vectors and scalars. To create a new model the user can select the “**NEW**” button on the “**Load**” section of the main menu. The program will ask for a name and the type of simulation. The rest of the data such as initial conditions, constraints and other data can be entered through the corresponding menus in the “**Sim**” section. Using the menus the model data will be automatically created and upon simulation saved to the default “Models\” directory in a MAT file.

**EXCEL Models** – The model data can also be stored into an Excel file and automatically imported into the MotionSim program. Through the excel file the user can view, make changes and easily store model data. To create a model, there are two templates available (TEMPLATE2d.xlsx and TEMPLATE3d.xlsx) which the user should duplicate and rename to create a new model. The name of the excel file should represent the model in some form, and due to Matlab limitations should not start with a number. The user can select between the different sheets and enter in the model information to be simulated.

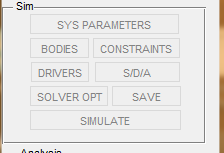
### Simulation and Analysis

After the model is loaded into memory and ready for simulation, the user then hits the “**Simulation**” button located in the “**Sim**” panel. The program will then run the simulation based of the model type and selected solvers.

When the simulation is complete the data is process and saved in the “Results\” folder with the same filename. The user can then analyze the calculated data using the “**Analysis**” panel on the Main Menu or loading the generated MAT file to perform their own analysis.

The preferred method of entering in data is through Excel based models. In the creating an Excel model section, how to create constraints are more detailed in the section but share the same data entry regardless of the method used. If the user imports an excel model, the program will automatically convert them to a MAT file and the below menus will display that data.

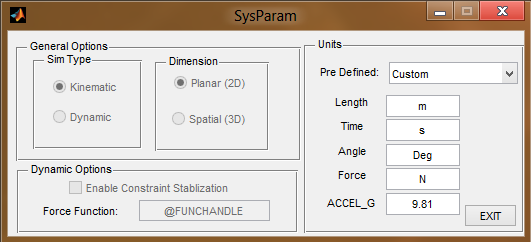
## Preprocess Menus



The preprocess menus contains all the buttons located in the “Sim” Panel of the MainMenu. These allow the system parameters such as bodies, constraints and solver configuration to be modified directly before simulation. A screen shot of the menus attached to each button and a general description of the menu is described below.

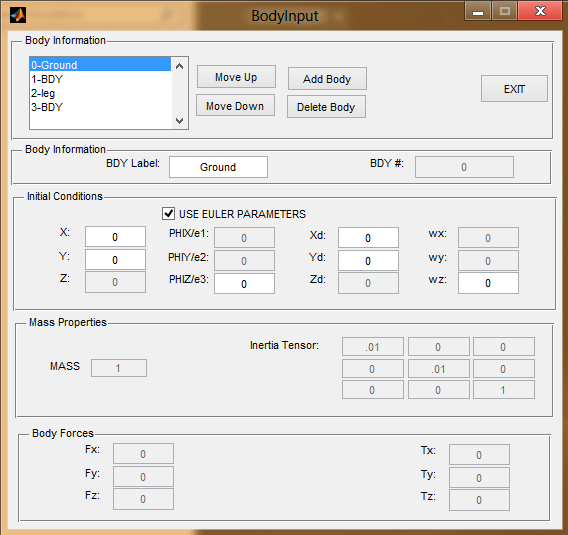


### Simulation Parameters



The “sys parameters” menu allows the selection of the simulation type and units. Once a model has been created the “Sim Type” and “Dimension” will be greyed out, as they are no longer changeable after model creation. If the model is a Kinematic model then constraint stabilization and force function will not be selectable. The units can be defined by the user or selected from a pre-defined list from the drop down menu.

### Body Data

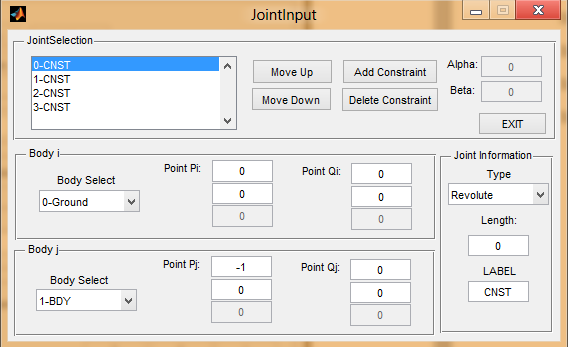


The “BODIES” button brings up the Body Input menu, which will allow the user to create, delete, renumber and enter in all the relevant data needed for simulation. Depending on the simulation type, different parameters will be greyed out indicating a not needed parameter. The above figure shows a planar kinematic model, with very few coordinates needed.

It is recommended to setup the body information first, as the other menus depend on the body information to define constraints, drivers etc.

See setting up models in excel files for a full detail of each parameter.

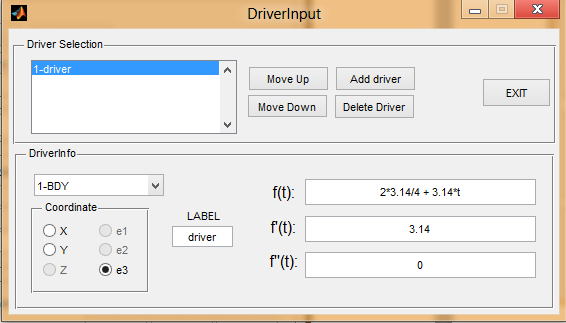
### Constraint Data



The “CONSTRAINTS” button will allow the user to create, remove or modify constraints in the model. Constraints options will only present the available constraints allowed for the simulation type.

All constraints share the same definition in Lankarani’s class and notes and are discussed in more detail in building an Excel model in later sections.

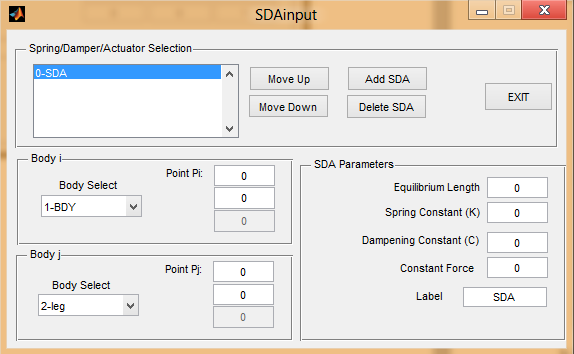
### Driving Constraints



The “DRIVERS” button on the main menu will bring up a screen that allows the user to modify driving constraints in the current model. Depending on the simulation type, unused parameters will be greyed out and not available.

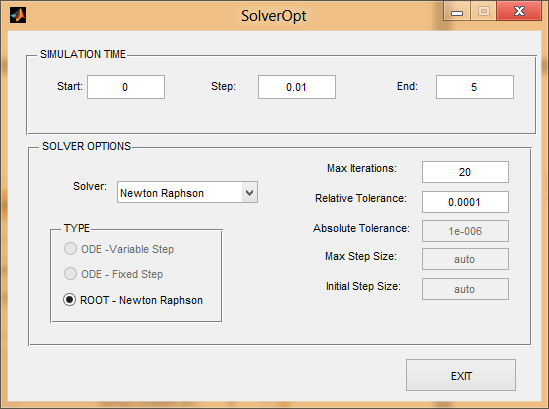
The user can add, delete or modify driving constraints from this menu. The f(t) and its derivatives used for driving constraints are Matlab evaluated functions at runtime with the variable t representing the current time. Meaning any valid Matlab expression can be entered into the f(t) definitions.

### Spring/Dampers/Actuators



The “SDA” button will bring up a menu that allows the user to input any springs, dampers or actuators in the simulation. The user selects the bodies and the coordinates where these elements are located with respect to the bodies. Then fill in the coefficients to the element. If no dampening is desired, the constant C should be left 0.

### Solver Options



The “SOLVER OPT” button will allow configuration of the solver to be used on to solve the system. Depending on the system type different selections will be available. The time is now selectable and the Step will be the step sized used and reported. The selections are detailed below and user definable.

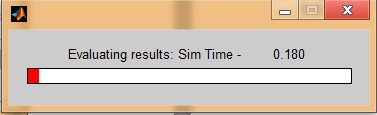
**Root Finding**: *Newton Raphson* – This algorithm is used for both planar and spatial kinematic simulations only. The step size and simulation time is directly set in the “SIMULATION TIME” panel. The Max iterations parameter sets the maximum number of NR iterations allowed. The Relative Tolerance sets the maximum allowed tolerance of any constraint equation during that timestep.

**ODE**: *Variable* – The variable ODE solvers uses Matlab’s standard ODE solutions. The default is the ODE45 algorithm based on the 4th order Runga Kutta algorithm with 5th order error correction. Six of Matlabs other solvers are available for use and are described in more detail in Matlab’s help file. These additional solvers are mainly used for special situations.

The step size is used for the reported steps that will be recorded for analysis, the keyword “auto” can be used instead of a number to report all steps used for computation. The tolerances and max/min stepsize can also be specified, and “auto” can be used to automatically generate them. More information on defining these options can be located in the Matlab’s help file.

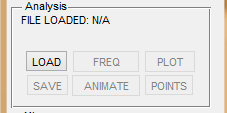
**ODE**: *Fixed Step* – Six fixed step solvers have been added to the program. The common fixed step solvers are Eulers, Heuns and Runga Kutta. There are few solver options to set for fixed step solvers, accuracy is maintained by specifying a proper time step.

### Simulate Button



To run the simulation simply pressing the “**SIMULATE**” button will begin the simulation of the pre-defined model. A status bar will appear showing the progress of the simulation and once again as it processes the results of the simulation. When a simulation is complete stats will appear in the command window for variable step solvers.

## Analysis Menus



The “Analysis” panel of the MainMenu allows the user to plot, animate and discover points of interest. After a simulation results from the simulation is automatically loaded and buttons enabled. The user can also load results directly from previous simulation via the “**LOAD**” button.

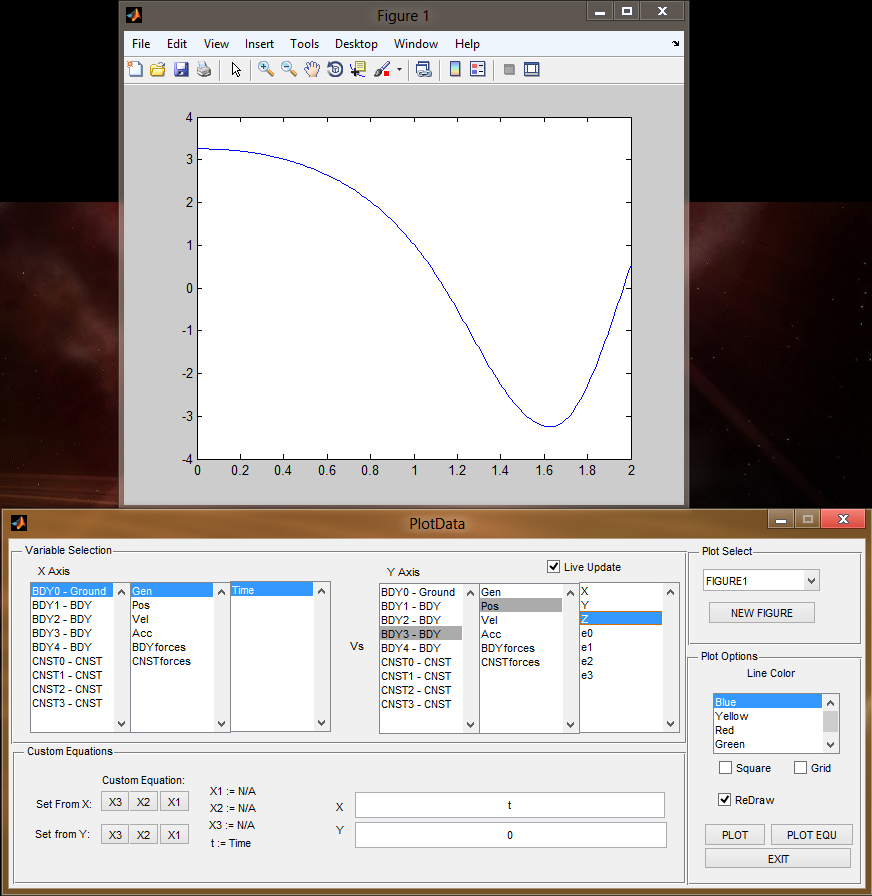


### Loading\Saving

**Loading** – When a simulation is complete the results will be automatically loaded. When the results are automatically loaded, any previous results will be overwritten. The user can also manually load previously simulated results by clicking the LOAD button, and opening a results file in the “Resutls\” folder.

**Saving** – When a simulation is completed the results are automatically saved. Although if the user has made modifications to the results, such as reprocessing point data, the results will need to be saved once again. When saving, the results are saved with the same filename in the “Results\” folder.

### Plotting Analysis

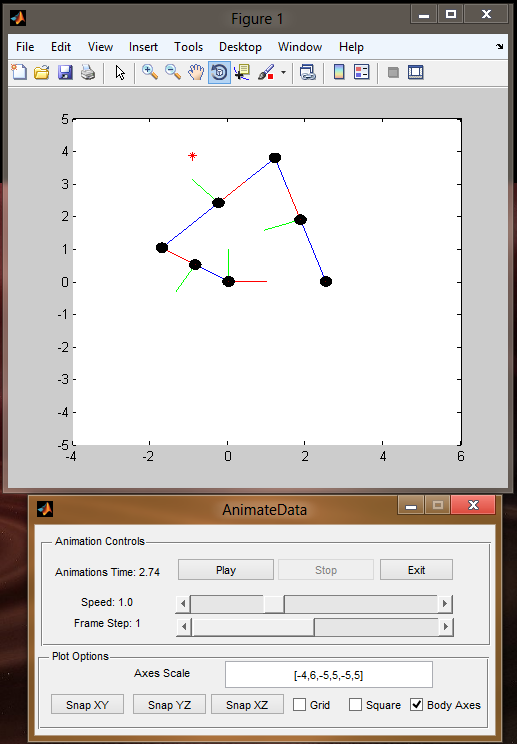


This “PLOT” menu allows the user to plot body, constraint and point data that has been simulated. Depending on the simulation type, the set of available coordinates will change.

All data is plotted in the standard Matlab plotting area, which allows the user to better analyze and customize the plot to their liking. The standard Matlab plotter gives powerful features located in the tools and view pull down menus. The user can also save the plot in any format they wish, by using the standard file-save action.

Custom Equations Panel allows the user to capture a coordinate into variables. X1, X2 and X3 are variables that can capture the coordinate of interest and be used in expressions. The X, Y text fields allows the user to input Matlab expressions based off the X1, X2 and X3 variable. To plot the custom equation the user presses the PLOT EQU button. This allows the user to accomplish advanced analysis, such as plotting motion ratios etc.

### Animating Models.

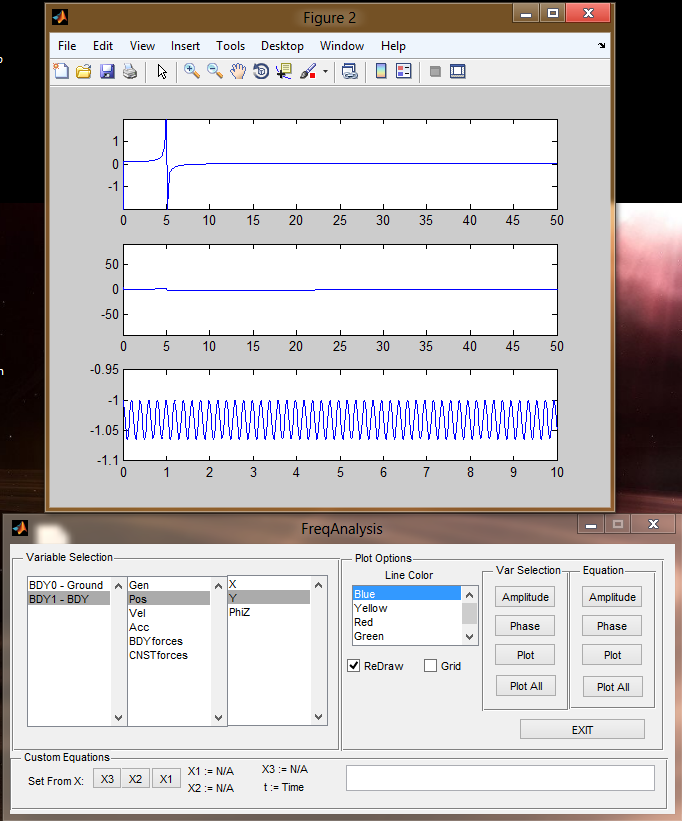


The animation menu allows the user to run simple animation of the defined systems. Lines are drawn from constraints to help visualize the system, but may not visually represent what the user had in mind.

Animation controls are available as well as quick plot options to help visualization.

Spatial simulations allows the user to rotate the plotting area to grant a full 360 view. Again the standard Matlab plot area is used, to allow as much customization by the user possible.

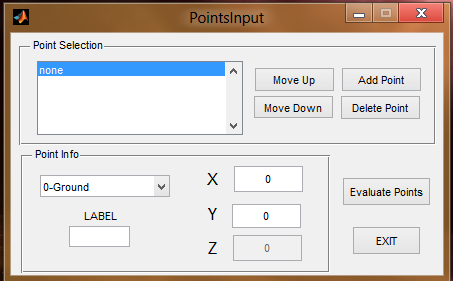
### Frequency Analysis



Frequency analysis can be performed on anybody, constraint or point coordinates. A FFT is performed on the signal and an attempt to properly scale the graph is made. The user may need to adjust the zoom level on the plot to get the data of interest into scale.

The Phase, Amplitude and plot graphs are all available either individually or in 3 subplots on the same figure.

### External Points Analysis



Points are locations attached to a specific body used for analysis. This allows the user to identify positions, velocities and accelerations away from the body origin.

Addition of points and any changes to points will require the results to be re-evaluated by clicking the “Evaluate Points” button.

Point will show in the animation and can be plotted in the plot and animation menus.

# Running Simulations

To run a basic simulation the user must do the following.

* Create an excel file that categorizes the system based off the template (MUST USE TEMPLATE)
* Place the excel file in the models directory
* Open matlab, change directory to main directory, and type MotionSim in the command window.
* Select the excel file and click load from Excel File.
* If no changes to the solver etc are needed, the user can hit the simulate button to simulate the system.

# Creating Models Through Excel



## Introduction

Excel is used as an easy data entry and customization of the system data. Due to the number of parameters to keep track of, organization has been formatted to be as easy as possible.

There are two different templates to be used for the generic simulations. A 2d template and 3d Template are provided, copy and paste these files and rename them to the name of your system. **THE 2D OR 3D TEMPLATES MUST BE USED.**

## Color Coding

Light blue parameters - are for necessary for both kinematic and dynamic simulations

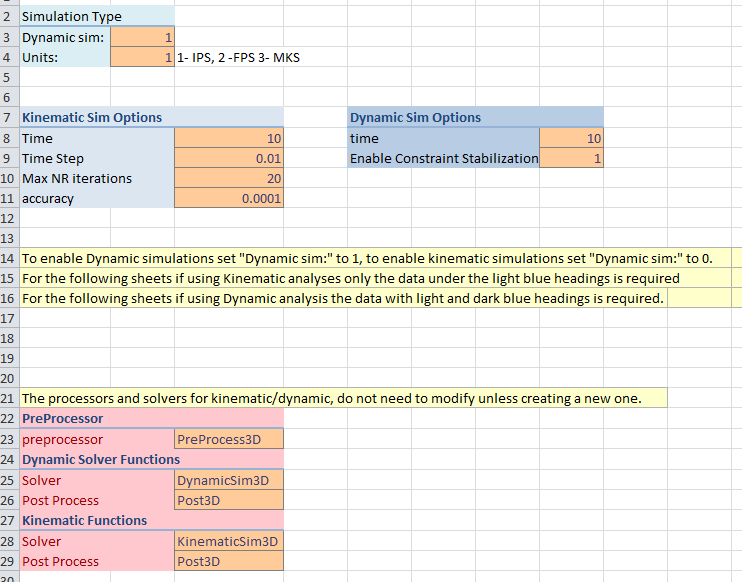
Dark Blue Parameters - are for Dynamic simulations only. If running kinematic simulations, it is better to place zeros in the parameters that aren’t used.

Light beige boxes - are for user input only, no other boxes but these are needed for input and will be ignored.

Light Tan boxes - are for notes on for your information of certain selections.

Dark pink – Found on the INFO sheet, are not to be changed unless custom processors has been developed.

## Simulation Info Sheet



The simulation information sheet looks like the above, the sections will be described below and how to fill this out. This sheet sets up the type and generic information needed to simulate the system.

### Simulation Type

Dynamic Sim: put a 1 for a dynamic simulation or a 0 for a kinematic simulation

Units: Sets default units (all parameters entered must follow consistent units) 1 for IPS, 2 for FPS, 3 for MKS systems

**Kinematic Sim Options** (Kinematic Sim’s Only)

Time: Simulation Duration in Seconds

Time Step: The step size used for the solver, if .01 will solve the system from 0 – 10s in .01s time steps.

Max NR iterations: Sets a limit on how many NR iterations are allowed, if system is ill defined may cause infinite loop, this parameter prevents that.

Accuracy: Sets the desired iteration accuracy of the system. More accuracy requires more iterations of the NR algorithm

**Dynamic Sim Options** (for Dynamic Sims only)

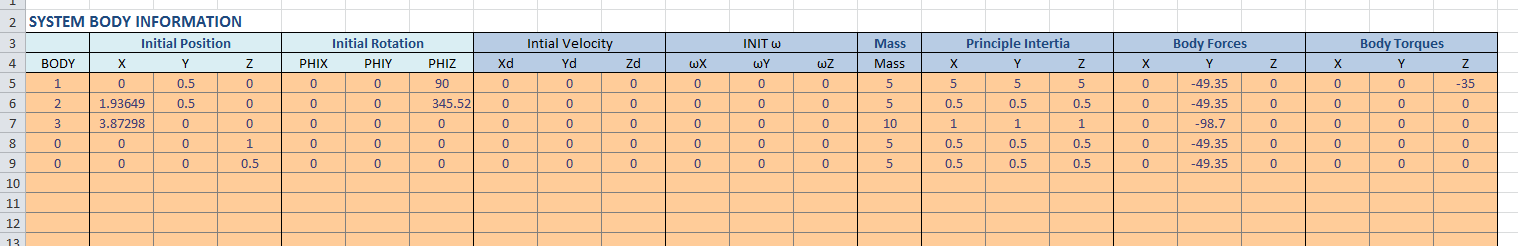
Time: Simulation Duration in Seconds,

Enable Constraint Stabilization: Enables error correcting of the constraints, in order to prevent error accumulation as time progresses (Baumgarte’s Constraint Stabilization Method)

**Preprocessor – Dynamic Solver Functions – Kinematic Functions** (Not usually modified)

NO LONGER USED

## Body Sheet



This sheet stores all the body information and initial conditions. Tabulation of the data is essential in order to construct a proper system. For Kinematic simulations, the light blue boxes are used only. The dark blue parameters are additionally required for dynamic simulations

**System Body Information**

Body: This sets the body number to use, good practice to number 1,2,3 consecutively down the column. To disable bodies, but keep the information handy for later use body number 0 to disable the row.

Initial Position: Set the X,Y,Z initial location of the body, kinematic template will only have X,Y.

Initial Rotation: sets the Phix,Phiy,Phiz rotations of the body. Only phiz is considered in the 2d case, and is automatically converted to Euler parameters in 3D simulations.

Initial Velocity: Sets the initial velocity of the system

Initial w: set the initial angular velocities of the system

Mass: sets the mass of the body

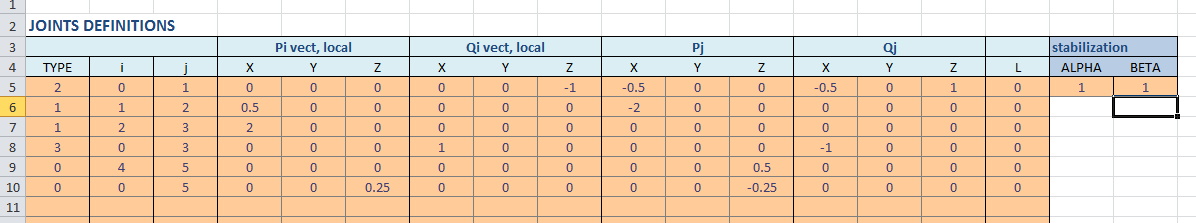
Principle inertia: for dynamic sims. sets the Ixx, Iyy,Izz portion of the inertia tensor. It is assumed the initial rotation is such that the local axes are in the principle direction.

Body Forces: set the body forces in effect throughout the simulation, usually forces caused by gravity are entered here. (g\*mass to get equivalent force produced by gravity)

Body Torques: set the body torques that is in effect through the entire simulation. Good for cranks etc.

NOTE: Euler parameters can be entered in, only when putting a 1 in the use euler paramaters option at the top of the excel sheet. Then entering in e1, e2, e3 in the initial rotation fields, The solver will automatically find e0 through the relation e0^2 + e1^2 + e2^2 + e3^2 = 1.

## Constraint Sheet



This sheet is to enter the constraint information for the system. Not P’s and Q’s points are needed for each constraint. When not needed the unused points should have zeros placed instead.

**Joint Definitions – General Information**

Type: The joints are broken into different types, such as 1 for spherical, 2 for translation etc. Use 0 to disable a constraint but keep the data. More information on the constraints and how to fill out each one is giving below.

I,j: Each constraints correspond with a body I, or a body j. Use 0 for a grounded body, and 1,2,3 corresponding to the proper body number entered into the body sheet.

Pi,Pj: enter in the X,Y,Z local coordinates for different constraints, These points are usually used for all constraints. I and j correspond to the constraint I, j defined on the left.

Qi,Qj: These points are used for certain constraints that require more than just 2 points on a body. Place zero’s in their fields if not used.

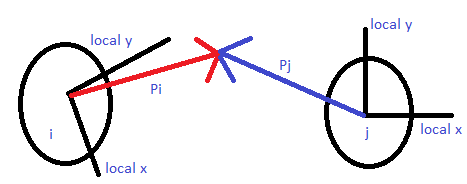
L: used for the S-S constraints, to define the length of the “ghost” link.

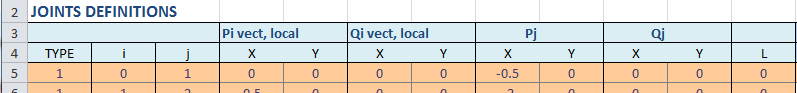
Stabilization: Enter the alpha,beta values to be used for constraint stabilization. This is for dynamic sims with constraint stabilization enabled.



### 2d CONSTRAINTS

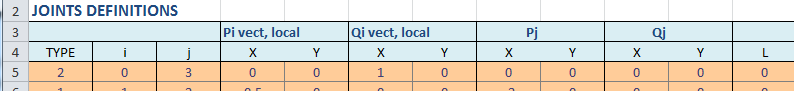
Revolute: constraint type 1 – two point’s pi,pj need to be filled. Pi and pj are local vectors of their respective bodies. These two points need to coincide together in the general coordinate system.





Example of a revolute joint, between body 0(ground) and 1, where the joint is at the 0,0 location of the ground, pj points to the coordinate.(this case the body is .5 units above the ground, so it is -.5 on its coordinate system)

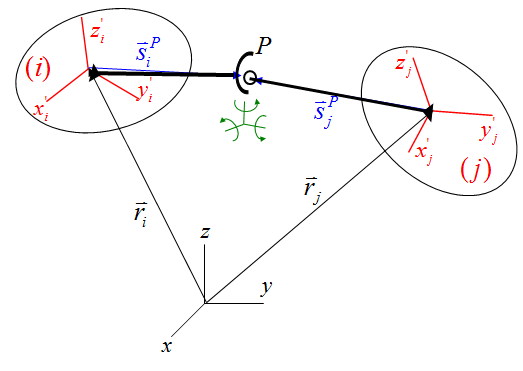
Translation: constraint type 2 – 3 points need to be defined pi,qi,pj all in local coordinates. They must all fall on the same line of action. The pi, qi points need to be two different points that fall on the line of action and are defined off the local coordinate system on body i. pj is the local coordinate of body j, that also falls on the line of translation



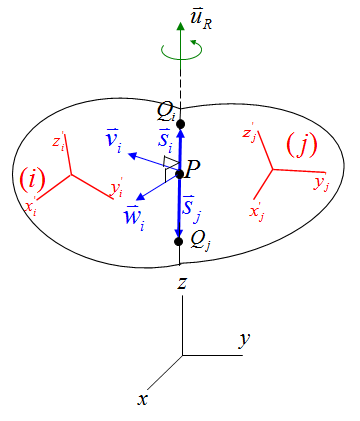
Example of a translation joint between body 0(ground) and 3. Pi and qi correspond to body 0 local coordinates system and is on the line of action. Pj is also a point on the line of action, where the body just happens to be on that line.

### 3d CONSTRAINTS

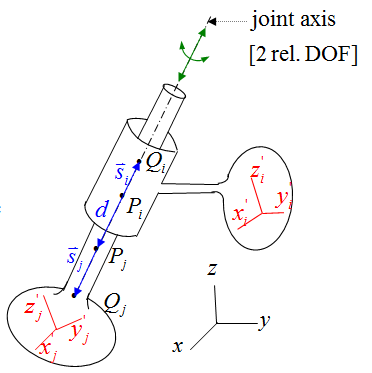
Spherical Joint: constraint type 1 - Required local points pi and pj (fill in qi and qj as zeros). Define to vector to point p that is common between the 2 bodies.



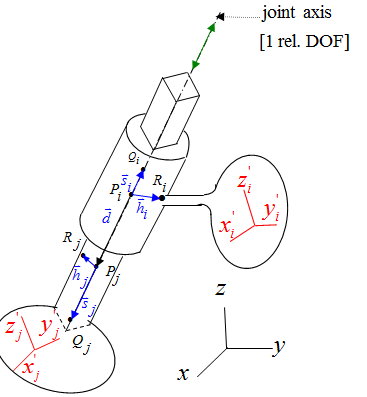
Revolute joint: constraint type 2 - fill in pi,qi,pj,qj. The pi and pj are the local coordinates of the common point, just like the spherical joint. The qi and qj are defined on the axis of rotation. Qi and qj being defined by their respective body local coordinates.



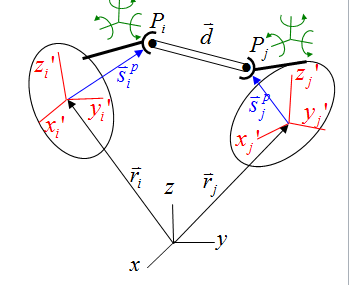
Cylindrical joint: constraint type 3 – This constraint requires 4 points (pi,pj,qj,qi) along the line of translation. Each with their respective body local coordinate system. As shown in the figure below.



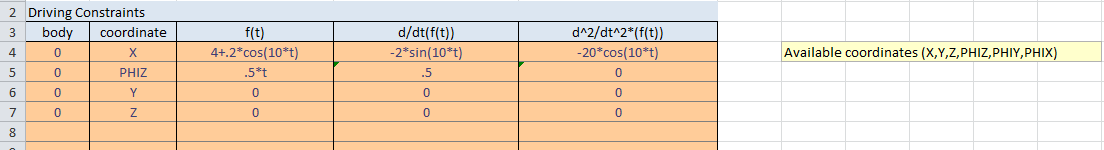
Prismatic joint: constraint type 5 – This constraint requires 4 points (pi,pj,qj,qi) along the line of translation. Each with their respective body local coordinate system. As shown in the figure below and is defined the same as cylindrical joint.



Spherical-Spherical joint: Constraint type 4, This constraint requires pi,pj and L to be entered in. L corresponds to the length of the “ghost” link. While the pi and pj correspond to the ends of the link, and must be filled out with respect to their local axis on body I and body j.



## Driving/Simple Constraint Sheet



This sheet is where the driving constraints are to be entered. It is required for all dynamic and kinematic simulations to put in the function and its derivatives. This is filled in by first defining the body, the coordinate of the body that is being driving. And the function/values of that coordinate over time. The variable t is used to represent time.



### Driving Constraints

Body: body number of the driving constraint to effect. Use 0 to disable the constraint but keep the information.

Coordinate: use X,Y,Z,PHIX,PHIY,PHIZ for 3d or X,Y,PHIZ for 2d. These are the available coordinates that must be Capitalized just as shown. The functions will affect these coordinates attatched to the particular body. For 3D Kinematic Simulations, use only phix,phiy,phiz once per body, using phix and phiy will cause errors most likely. If need to drive multiple rotations use euler parameters e1,e2,e3. Derivatives for using e1, e2,e3 are in w and wd though.

F(t): the function based of the position, if X is selected as the coordinate, then X position of the body will be drivin to this value. Use t as the variable for time. And regular math cos,sin,\*,/ is allowable just as you would enter in matlab normally. If not entered with normal matlab syntax, an error will occur, such as 20sin(t) needs to be 20\*sin(t). (NOTE: matlab evaluates cos,sin in radians)

d/d(t) (f(t)): derivative of the function f(t) , same rules apply as in defining f(t)

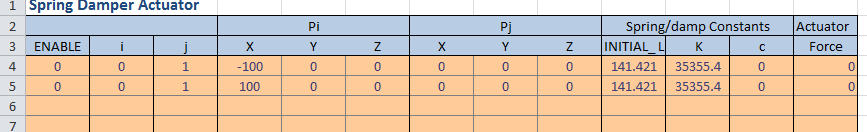
d^2/d(t)^2 (f(t)): 2nd derivative of the function f(t) , same rules apply as in defining f(t)

### Simple Constraints

To configure a simple constraint simply enter in the number in the f(t) field without the variable, Then put zero for its subsequent derivatives. This is exactly identical to simple constraint formulations, as it is processed identical to driving constraints but do not rely on time. An example is given below that sets body 2’s Y coordinate to 5.

**Body**: 2 **Coordinate**: Y **f(t):** 5 **d/dt(f(t)):** 0 **d^2/d(t)^2 f(t):** 0

## SDA Sheet



This is the SDA (Spring Damper Actuator) sheet, this controls all the information to add springs, dampers, actuators either all in effect or individually. This works by defining a point on each body where the elements are to be between. To disable certain elements like dampers, just leave c as 0 and the element will be disabled.

**SDA Definition**

Enable: This enables or disabled a spring/damper/actuator. Use 1 to enable, 0 to disable. Usefull to disable but still keep the information. Will be ignored by solver.

I,j: the I and j boxes correspond to 2 different bodies the springs act upon. Use 0 for ground.

Pi,Pj: points on the I and j bodies, enter the LOCAL X,Y,Z locations of the SDA’s on each body. All zeros for the body origins.

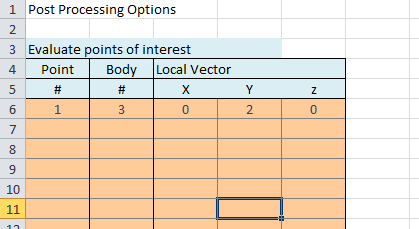
INITIAL\_L: This is the initial length of the spring, used to calculate spring forces.

K: Spring constant, set to 0 to disable

C: damping constant, set to 0 to disable

Force: actuator force, will constantly apply force between the two points. ,set to 0 to disable

## Post



The post sheet is where the post processor information is entered. Sometimes it is useful to see how different points on a body at different locations behave. This section allows you to define a point on a body that will be processed, and can be analyzed later in the post processor.

You can also preset your axis and body coordinates sizes in this sheet, just as you would in matlab. The axis has a format [xmin xmax ymin ymax zmin zmax].

Definitions:

Point #: Enter a unique number for the new point. This number will be displayed in the body’s box of the graphing GUI in the post.

Body #: This is the body number that the point will be attached too.

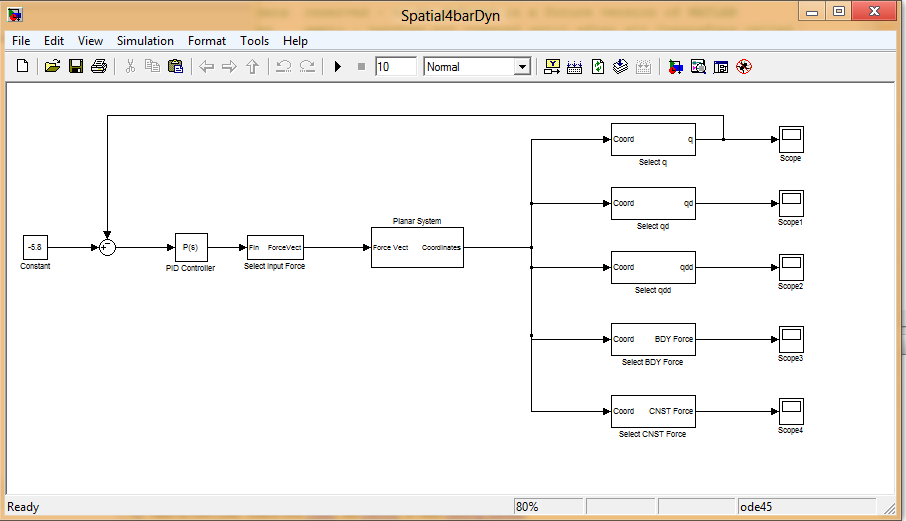
Local Vector: This is the x,y,z (x,y for 2d) local coordinates of the point on the body.

# Simulink Integration

MotionSim solver can now be used in the Simulink Environment. This is considered to be a more advanced feature, to simulate specific problems. Simulink integration will require good knowledge of Matlab and Simulink environment to implement successfully.

The same excel or MAT models are used to load in the system information. The user should create a model just as they would use the normal program.

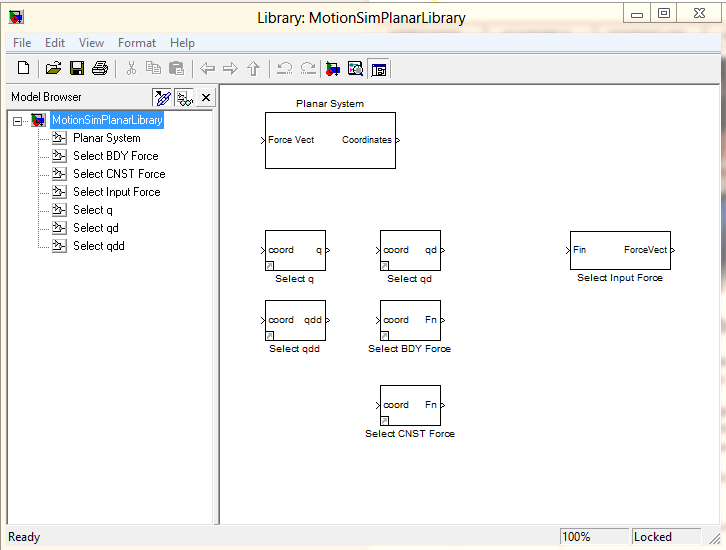
Currently only planar and spatial dynamic simulations are available.





## MotionSim Libraries in Simulink

Two libraries exist a Spatial Dynamic and Planar Dynamic library exist. The user should use these blocks in order to create a system in their Simulink environment.



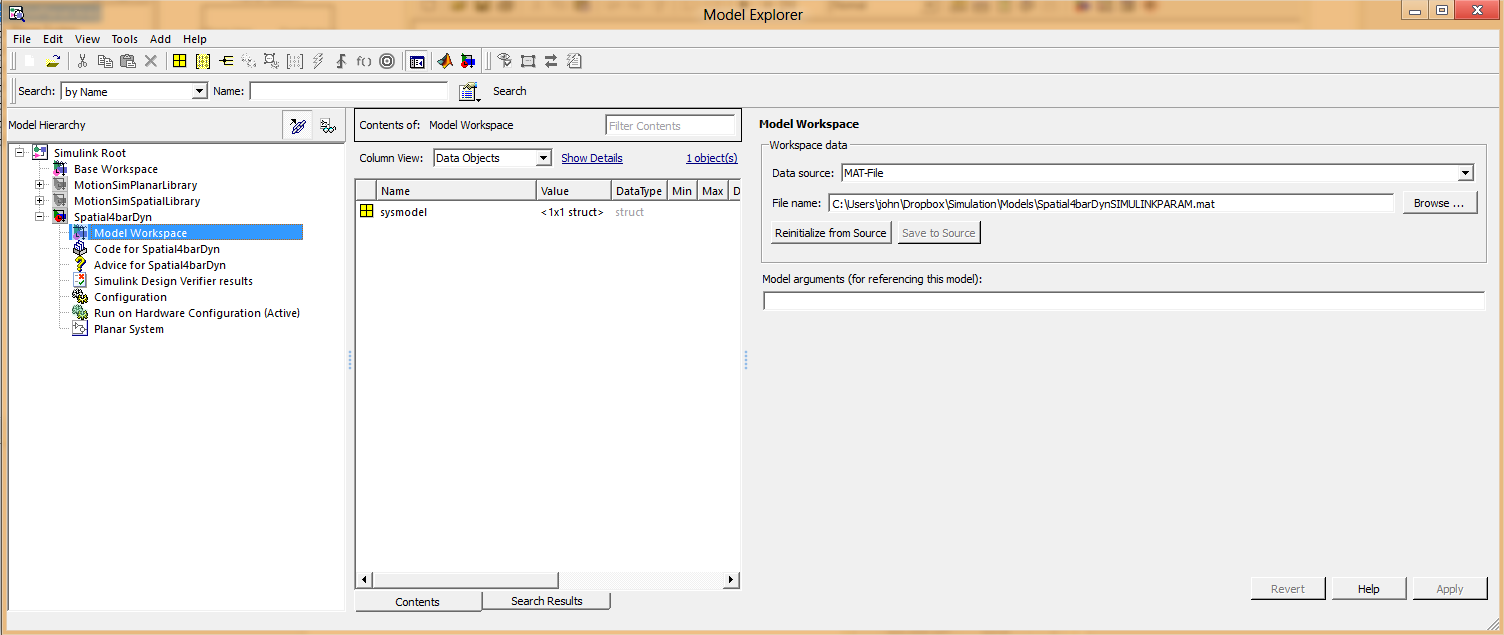
The planar library is shown above, after the system parameters are loaded into a new Simulink model. The user can place the “Planar System” block that will invoke the solver to solve the system. The other blocks are used to intercept signals from the solver output to manipulate and graph in the Simulink environment.

Input forces are available to change dynamically while the simulation is run. This allows any Simulink input to the solver to adjust for forces etc. Some examples are from actuators or servos.

## Creating a Simulink Model with a Dynamic System

To create a new Simulink model follow the instruction below, which is an ordered list describing the steps to create a Simulink model.

* The user should first convert an excel or MAT model to a Simulink compatible structure. Simply highlight the preconfigured .xlsx or MAT model and click the “CONV SIMULINK” button. A .MAT file will be create in the Simulink folder with the model name.
* Open Simulink by typing “Simulink” in the command window
* File-New Simulink model to create a new Simulink model and save the file
* Type CTRL-H to open the model explorer, and navigate to your model’s workspace
* In the model workspace options configure the following: (only done once)
  + Data-Source: MAT-File
  + File Name: Browse to the Models Mat file that was generated in the Simulink folder (done in the CONV SIMULINK step)
  + Finally click “Reinitialize from Source” to load data right away



* Next open the appropriate library: MotionSimPlanarLibrary for Planar system or MotionSimSpatialLibrary for Spatial Systems.
* Drag and drop the “Planar System” or “Spatial System” block into the models workspace.
* Additional use any of the signal selectors to grab the solvers input and output.
* Integration of the MotionSim model is complete, Simulink can now be leveraged onto the model

## Simulink Examples

Two examples files are located in the Simulink folder that shows off the spatial and planar solvers and their use.

# Program Detail



## How Models are solved

The solvers between 2d and 3d are extremely similar; the differences primarily come from change in dimensions and not the way it is solved. There are two main solution methods, one for the kinematic and the other for dynamic. The Dynamic there are some differences in the way the angles are treated between the 3d and 2d case, where the 3d uses Euler angles.

For all solutions require the same matrices in order to solve, these matrices are pre computed before each iteration. These functions are functionalized and used between the kinematic and dynamic solvers. Below is the list of matrices used.

C - Constraint Matrix

Cq – Jacobian of constraint matrix

Ct – Time derivative of the constraint matrix

Ctt – second time derivative of the constraint matrix

Qd – The (–Ctt – (Cq\*qd)q\*qd – 2\*Cqt\*qd) term the develops on the 2nd derivative of the coordinates



### KINEMATIC

With C for the constraint matrix and jacobian Cq for the derivative of the matrix takes place of the f and f’. The variable x represents the vector of coordinate q, can be written as follows below.

Which then can be re arranged and iterated to find a vector of coordinate’s q that satisfies the accuracy wanted. The final form and matlab code is show below.

**Rearranged Form**:

**Matlab Code**: q = Cq\(-C)+q;

With the vector of coordinates known for the specific time t, we can then take the derivatives of the constraint matrix (C = 0) to find the velocities and accelerations. Below is the general derivative of the constraints.

Re arranging this equation and solving below

**Rearranged Form**:

**Matlab Code**: qd = Cq\(-Ct);

For the accelerations the 2nd derivative is taken and put in the form below. Qd is as described above.

Re arranging the equations we can solve for qdd directly as shown below.

**Rearranged Form**:

**Matlab Code**: qdd = Cq\Qd;

For 3D kinematic, euler parameters are used to solve the positions, giving each body 7 DOF, with one dof taken automatically by the equation e1^2 + e2^2 + e3^2 +e0 ^2= 1

### DYNAMIC

The dynamic solver uses the 4th order Runga Kutta numerical integration scheme to integrate the positions and velocities. This method requires the accelerations in order to integrate to velocities, and then it uses those velocities to integrate to positions. Inside the RK algorithm the accelerations need to be solved. Using equations such as F=ma and T = I\*alpha, a mass matrix can be used to solve for their accelerations. For constraint systems an augmented matrix form is used, using the 2nd derivative of the constraints. Shown below is the method to solve for the accelerations.

\*q’’ = Qd

We can solve for the accelerations and input back into the RK algorithm to approximate the velocities and positions.

**3D Dynamic system angles representations**

For 3d Dynamic system, Euler angles are used instead of regular angles.

## File List

To organize files the code uses different directories to store the many files. The directories are explained here

* “Main directory” is the directory where all the folders are located, and contains the main program motionsim.m and mostionsimgui.m
* “Models” directory is the directory all the excel files should be placed.
* “PostProcess” directory contains the GUI and .m files for the post processor
* “PreProcess” directory contains the folder for the preprocessors, that process the excel file
* “Results” directory contains the .m files created by the solver, that store all the data at each time step
* “Solver” directory contains the generic functions shared by the 2d and 3d programs. The sub directories 2d and 3d contain the appropriate solvers for each.

## Variable Structure

TBD