OneCylindeEnginer Manual

Concurrent Dynamics International

April 2014

Objectives

- Part I:
  - Build a model file (OneCylinder.txt) to simulate an engine with a crank, a rod, a piston and a load
  - Build Simplot1.m to view sim results

- Part II:
  - Build a Simulink model (OneCylinder.mdl) that runs according to OneCylinder.txt in either two stroke or four stroke mode
    - Example1: Two stroke mode
    - Example2: Four stroke mode

License Restrictions

License type	Buildx.exe	Xsv01.dll
Enterprise	none	none
Project	Must stay with the object count specified by license	Runs with model_files with license specified object count

- Project license permits simulations of mechanisms with a specified object count in {bodies, wheels, forces} and in a unique configuration. No restrictions are placed on the mass property of bodies and wheels, and force placement and parameters or initial conditions.

How to Use This Manual

- OneCylindeEnginer manual is written for Enterprise license users where no restrictions are placed on the object counts {body, wheels, forces} in creating models.

- OneCylinder.txt is a seed model_file for Enterprise license users to create other models such as: multiple cylinder engine. I.e. by adding another rod and piston bodies with proper constraints yields a twoCylinderEngine

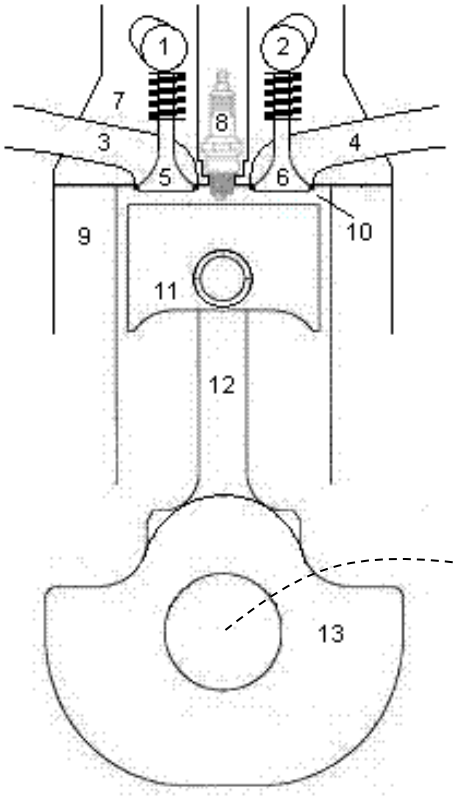
- CDI has many seed models to expedite the development of more complex mechanisms

- This manual is applicable to Project license users whose model object count is {5, 0, 1} and has a crank-rod-piston configuration as OneCylindeEnginer

Part I Topics

- OneCylinder model
- Buildx tasks
- Key files
- Main menu
- Model Menus
- Body Menu
- Body Actuation Signals
- Gravity
- Markers
- Constraint Menu
- Dynamics Input
- Dynamics Output
- Plot data
- Save model
- Simplot
- Exit buildx
- Q & A

OneCylinder Engine Model



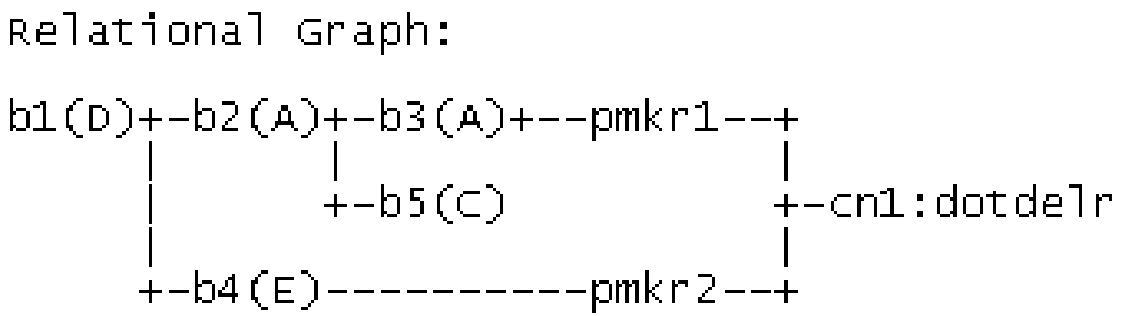
1 - Intake Cam
2 - Exhaust Cam
3 - Intake Port
4 - Exhaust Port
5 - Intake Valve
6 - Exhaust Valve
7 - Cylinder Head
8 - Spark Plug
9 - Engine Block
10 - Combustion Chamber
11 - Piston
12 - Rod
13 - Crank

b1=engine block (ground)
b2=crank
b3=rod
b4=piston
b5=load (connected to b2)

f1=combustion force on b4
tq1= crank shaft torque

y
z
x= shaft axis= y x z

pics credit: sites.psu.edu



- pmkr1, pmkr2: position markers defined on
  b3(end of rod) and b4(piston cm)
- cn1:dotdelr= constraint that enforces
  pmkr1.pos-pmkr2.pos=0
- A= 1 dof rotation, D=grounded, C=1 dof prescribed rotation
- E= 1 dof translational

- b4(piston) is given 1 dof translational motion (E) along b1.y-axis
- b4.cm (p1) motion is constrained to follow the connection point (p2) exactly at upper end of b3(rod), thus the two pmkr's (p1,p2)
- cn1=constraint(dotdelr) enforces axis'*(p1.pos-p2.pos)=0 with axis={ b1.y-axis,b1.z-axis}
- Hinge(5) motion is prescribed (C) relative to b2=crank and is set to zero, meaning that it rotates at the same rate as b2

On the Simulation

- We are building a model file in Part I to support a one cylinder engine simulation. Setup includes definition of mass property of bodies, constraints and forces. It also include the dynamics input and output signals definition required by the two-stroke or four-stroke operation of the engine.

- Part II will show how to build an mdl file to simulate the one cylinder engine running in either of the two modes.

- To expedite the simulation we prescribed a combustion force profile that starts at tdc, peaks shortly afterward and drops to zero after the piston move past the ½ point of down stroke. peak combustion force, piston friction force and the crank friction are adjustable to reflect target engines

- Combustion force profile should be based on either measured data or should be computed based on the thermal-chemical dynamics given the volume and pressure of the combustion chamber

- Users are encouraged to apply the latter effort in defining piston force and close the OneCylindeEnginer control loop with it for a more accurate engine dynamics representation

Buildx Tasks

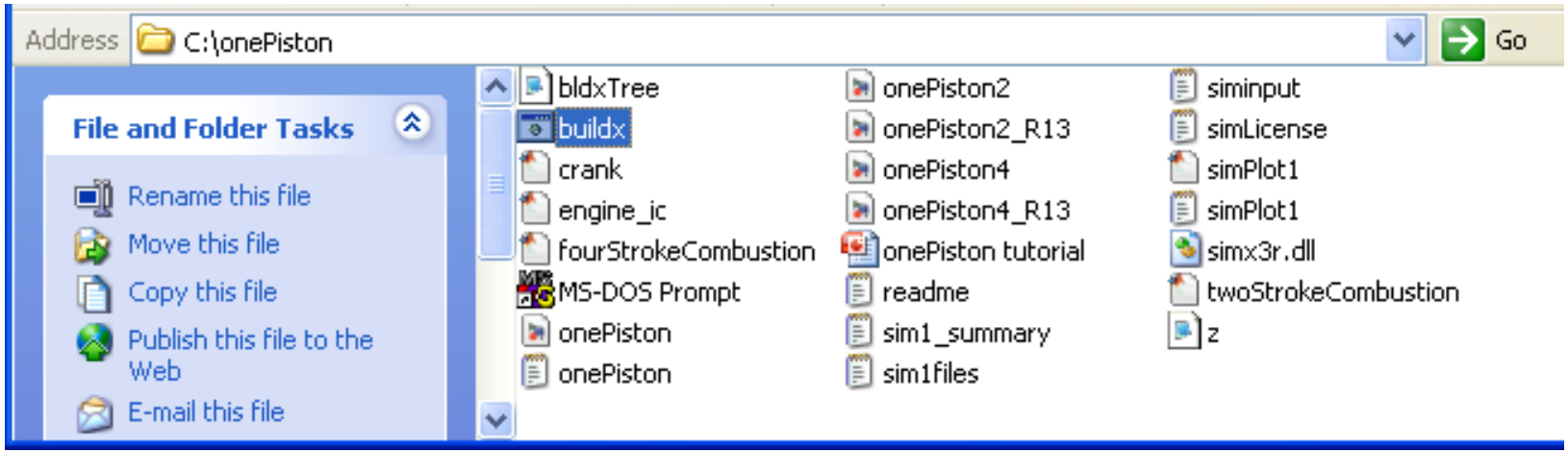
- Define 5 bodies
  - b1= ground, b2= crank, b3=rod, b4=piston, b5=load
- Define pmkr1, pmkr2, dmkr1,dmkr2, cn1:dotdelr
- Define f1 for piston forces
- Define gravity acceleration vector
- Define simx401.dll input/output/plot data
- Define Simplot1.m

Key Files

- Buildx.bat = '..\buildx'
- Input files pointer: siminput.txt
- Working files: sim1files.txt
- Model_file : OneCylinder.txt
- Simplot file: Simplot1.txt
- License: simlicense.txt

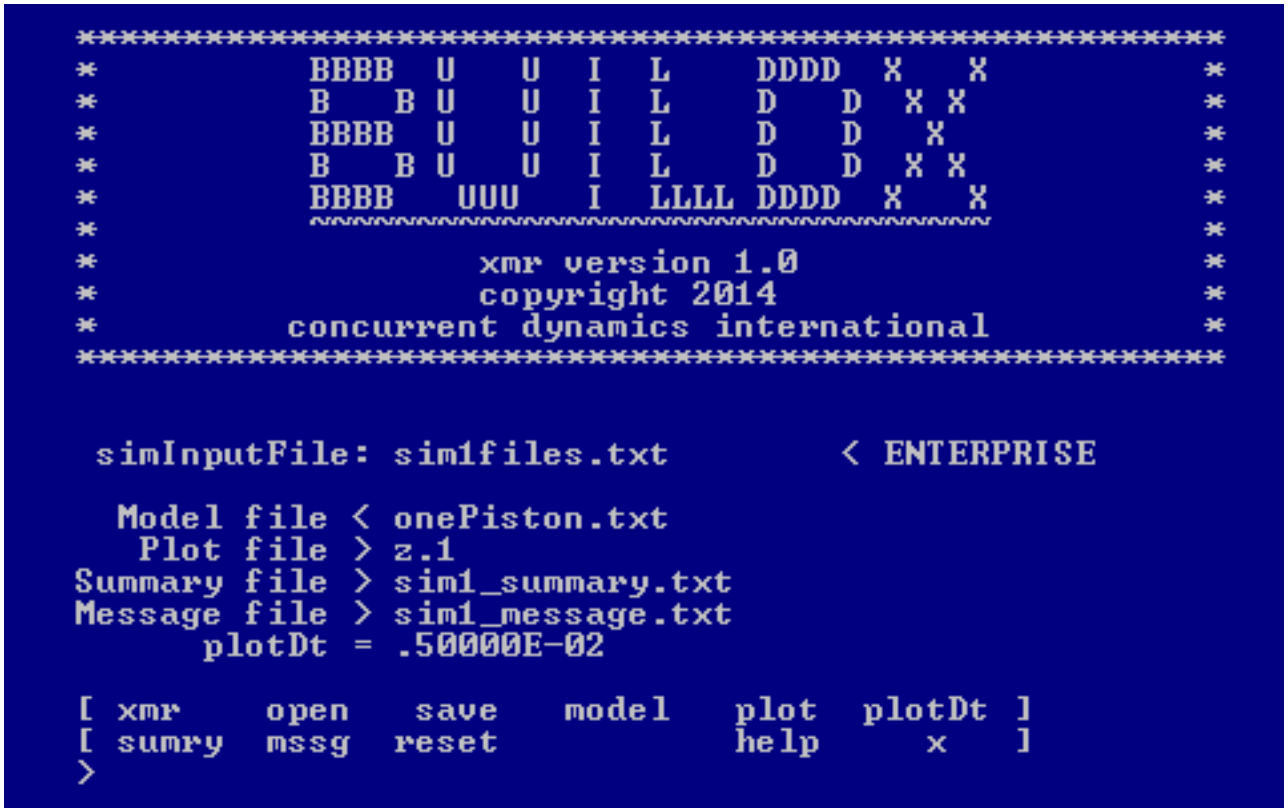
Start Buildx.exe

- Click c:\OneCylinder\buildx.bat to start buildx.exe



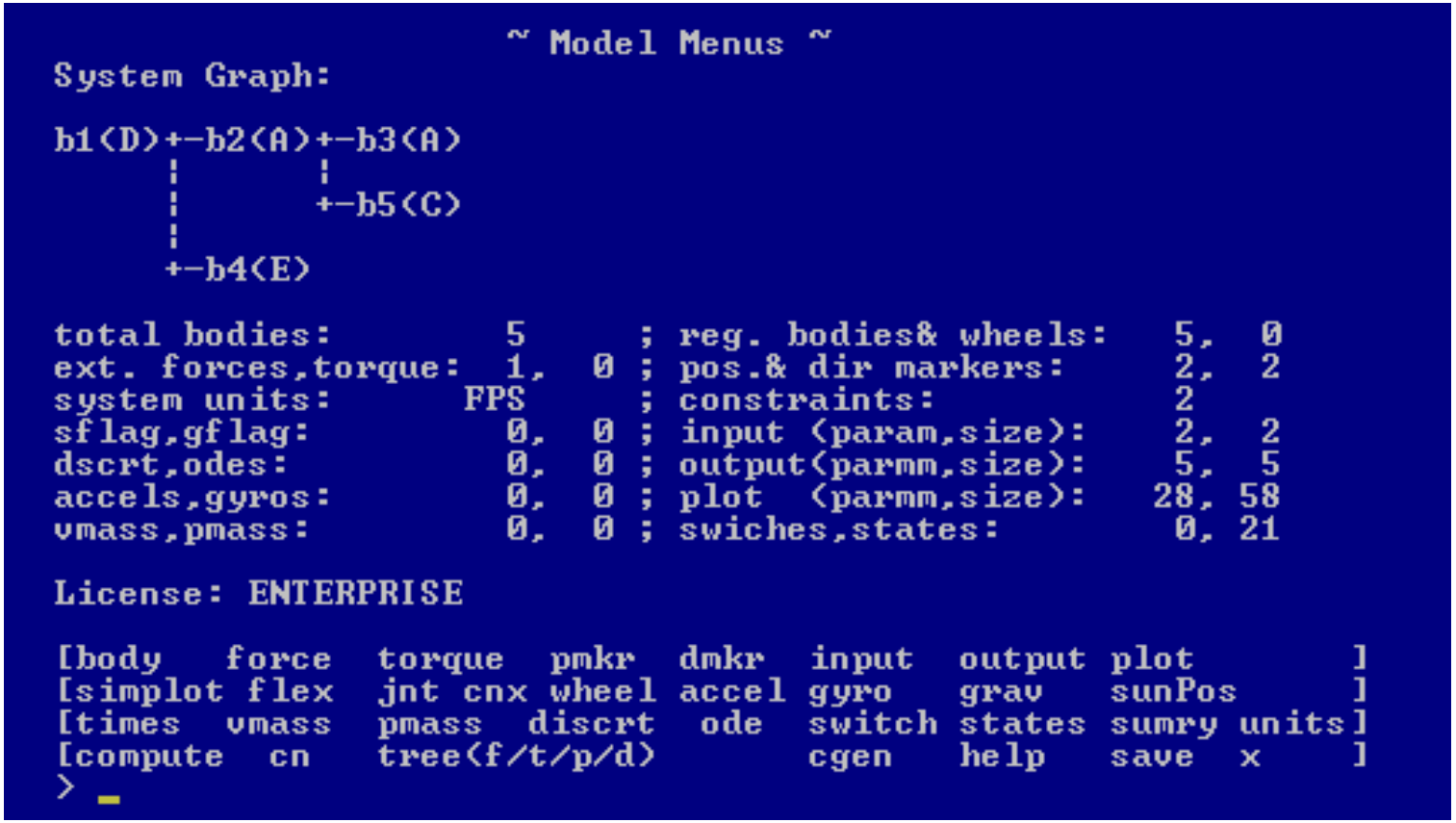
Main Menu

- See content of sim1files.txt displayed (model,plot,summary,message)
- Type 'xmr' to go to Model Menus



Model Menus

- See model objects size and to select menus for editing/browsing



Model Menus Commands

- Type 'body’ to go to xmr.Body Menu
- Type 'force’ to go to xmr.force menu
- Type 'grav’ to go to xmr.gravity menu
- Type 'pmkr’ to go to xmr.position marker menu
- Type 'dmkr’ to go to xmr.direction marker menu
- Type 'input’ to go to xmr.input menu
- Type 'output’ to go to xmr.output menu
- Type 'plot’ to go to xmr.plot menu
- Type 'Simplot’ to go to xmr.Simplot menu
- Type 'time’ to go to xmr.timing menu (edit plotdt only)

- Type 'help’ to get definition on data and commands
- Type 'x’ to exit menu

Body Menu

- See body summary, type ‘body’ from last menu
- OneCylindeEnginer has five bodies

```
                          ~ Body Menu ~
   no. of bodies :  5
   sysh_eci      :        .000          .000          .000
   sysh_b1       :        .000          .000          .000
   syscm         :        .000          .002          .000

   dvec =          .000          .000          .000
   svec =          .000          .000          .000
   hpos =          .000          .000          .000
   rpos =          .000          .000          .000
   wrel =          .000          .000          .000
   inr  =          .000          .000          .000
                  .0000         .0000         .0000

   idx name     pa  u   fl vm tp ax -- angle - -- mass --
=>   1 ground    0 FPS   0  -  D  x        .000         .000
     2 crank     1 FPS   0  -  A  x        .000         .010
     3 rod       2 FPS   0  -  A  x        .000         .010
     4 piston    1 FPS   0  -  E  y        .000         .006
     5 body5     2 FPS   0  -  C  x        .000        1.000

   [ sel  edit  idx   name  par  dvec  svec  type  whl  units]
   [ axis  ang  dpos  wrel dvel  inr   mass  hpos  hvel rpos ]
   [ add  addF  cnx   jnt  rem   dmkr  pmkr  gvec  rvel  w   ]
   [ brch  cn   copy  up   down  dolc  save  help  zero  x   ]
   > _
```

Body Menu Commands

- Type 'add<j>' to add bodies to bj : i.e. 'add5' to add child-bodies to b5
- Type 'name<j>' to edit name of bj.name
- Type 'par<j>' to edit bj.parent
- Type 'type<j>' to edit motion type of bj.hinge: {a…h}
- Type 'axis<j>' to edit bj.axis: {x, y or z}
- Type 'ang<j>' to edit inboard bj.ang for 1 dof rotational joints (type=a, c)
- Type 'mass<j>' to edit bj.mass
- Type 'edit<j>' to see all bj data

- Type 'help' to get definition on data and commands
- Type 'x' to exit menu

Inertia Menu

- Need define bj.inr for all j
- Type 'inr’ from Body Menu page to see inertia summary

```
   idx name    -     ixx    - -     iyy    - -     izz    -
               ---- ixy --- ---- ixz --- ---- iyz ---
=>   1 ground  .000000E+00  .000000E+00  .000000E+00
               .000000E+00  .000000E+00  .000000E+00
     2 crank   .500000E-01  .500000E-01  .500000E-01
               .000000E+00  .000000E+00  .000000E+00
     3 rod     .100000E-01  .500000E-02  .100000E-01
               .000000E+00  .000000E+00  .000000E+00
     4 piston  .500000E-02  .500000E-02  .500000E-02
               .000000E+00  .000000E+00  .000000E+00
     5 body5   .500000E-01  .500000E-01  .500000E-01
               .000000E+00  .000000E+00  .000000E+00
```

- Type 'inr<j>’ to edit bj.inr
- Type 'x’ to exit menu

Dvec Menu

- Need define bj.dvec, position of b.hinge in bj.parent frame
- Type 'dvec' from the Body Menu to see a dvec summary

```
   idx name         u ---------- ---dvec--- ----------
=>   1 ground     FPS         .000         .000         .000
     2 crank      FPS         .000         .000         .000
     3 rod        FPS         .000         .050         .000
     4 piston     FPS         .000         .150         .000
     5 body5      FPS         .000         .000         .000
```

- dvec(3).y= 0.05 ft means that over 1 rev of b2 (crank), the rod.hinge would have traveled 0.1 ft, therefore this engine has a stroke length of 0.1 ft

- dvec(4).y= 0.15 ft when all angles are zeros means that the piston hinge.y at tdc is 0.2 ft (.05ft+0.15ft) in b1 coordinates

- Type 'dvec<j>' to edit bj.dvec
- Type 'x' to exit menu

Svec Menu

- Need define bj.svec, cm position of bj in bj coordinate frame centered at bj.hinge
- Type 'svec' from the Body Menu to see a svec summary

	idx	name	u	---svec---		
=>	1	ground	FPS	.000	.000	.000
	2	crank	FPS	.000	.000	.000
	3	rod	FPS	.000	.050	.000
	4	piston	FPS	.000	.000	.000
	5	body5	FPS	.000	.000	.000

- svec(4)= zeros(3,1) means that the piston cm is co-located with its hinge
- Type 'svec<j>' to edit bj.svec
- Type 'x' to exit menu

Body Actuation Signals

- Bj Inboard force or torque actuates that body and impacts the motion of the rest of the system. Accelerations can be specified for joints with prescribed motion
- The Dynamics Input signals for bj are processed based on bj.type as follows.

type	Size	Input	processing
A	1	Htqax,j	Bj.torque(axis)=Htqaxj
B	3	Htq,j	Bj.torque=Htqj
C	1	Wraccax,j	Bj.wracc(axis)=Wraccaxj
D	3	Wracc,j	Bj.wracc=wraccj
E	1	Frcax,j	Bj.force(axis)=frcaxj
F	3	Frc,j	Bj.force=frcj
G	1	Hraccax,j	Bj.hraccax=hraccaxj
H	3	Hracc,j	Bj.hracc=hraccj

Position Markers

- Need define pmkr(j).pos for a constraint in this example
- Type 'pmkr' from the Body Menu to see 2 for this example

```
                        ~ pmkr Menu ~

  no. of pos markers:   2
  Pos change option:  POINT

                          -----x------ -----y------ -----z------
       refpos              .000000E+00  .000000E+00  .000000E+00
       posrb1              .000000E+00  .150000E+00  .000000E+00
       posri               .000000E+00  .150000E+00  .000000E+00

       refvel              .000000E+00  .000000E+00  .000000E+00
       velrb1              .000000E+00  .000000E+00  .000000E+00
       velri               .000000E+00  .000000E+00  .000000E+00

    idx name       pa    u ----posx---- ----posy---- ----posz----
 =>   1 pmkr1       3  FPS  .612303E-17  .100000E+00  .000000E+00
      2 pmkr2       4  FPS  .000000E+00  .000000E+00  .000000E+00
```

- pmkr1 is attached to rod (pa=3),located at [0 0.1 0] in rod coord
  this coincides with the b4(piston).cm position
- pmkr2 is attached to piston (pa=4) and located at [0 0 0] in piston coord

PMKR Menu Commands

- Type 'add<j>' to add pmkr's to bj : i.e. type 'add5' to add markers to b5
- Type 'pos<j>' to edit pmkr(j).pos
- Type 'rem<j>' to remove pmkr(j)
- Type 'help' to get definition of data and commands
- Type 'x' to exit menu

Directional Markers

- Need dmkr's for constraint definition for this model
- Type 'dmkr' from the Body Menu to see two directional markers

```
                        ~ dMkr Menu ~

  no. of directional markers:    2
  uVec( 1) in b1:       .000000     1.000000       .000000
  uVec( 1) in Pa:       .000000     1.000000       .000000
  az-axis          :    2

  idx name      pa ---------- ---uvec--- ----------
=>  1 yvec       1         .000       1.000         .000
    2 zvec       1         .000         .000       1.000
```

- dmkr1 is yaxis on b1 (pa=1), dmkr2 is zaxis on b1 (pa=1)
- Type 'add<j>' to add direction markers to bj : i.e. 'add5' to add dir markers to b5
- Type 'uvec<j>' to edit dmkr(j).uvec
- Type 'rem<j>' to remove dmkr(j)
- Type 'help' to get definitions of data and commands
- Type 'x' to exit menu

DMKR Menu Commands

- Type 'add<j>' to add dmkr's to bj : i.e. type 'add1' to add dmkr's to b1
- Type 'vec<j>' to edit dmkr(j).vec
- Type 'rem<j>' to remove dmkr(j)
- Type 'help' to get definition of data and commands
- Type 'x' to exit menu

Constraint Menu

- See constraint summary, type ‘cn’ from Model Menus or Body Menu

```
                        ~ Constraint Menu ~

    no. of constraints :  1
    cn rowSize         :  2
    cn index           :  1
    cn gama            : .00000E+00   .00000E+00   .00000E+00
    cn err             : .00000E+00   .00000E+00   .00000E+00

    cn k1, k2          : .00000E+00   .00000E+00
    cn pgain,vgain     : .40000E+06   .40000E+03

    idx type    ic  ln  ov   b1  b2 f1  f2 d1  d2  d3  p1  p2
    --- ----    --  --  --   --  --+--  --+--  --  --+--  --
 =>   1 DOTDEL   1   2   0    0   0: 0   0: 1   2   0: 1   2
```

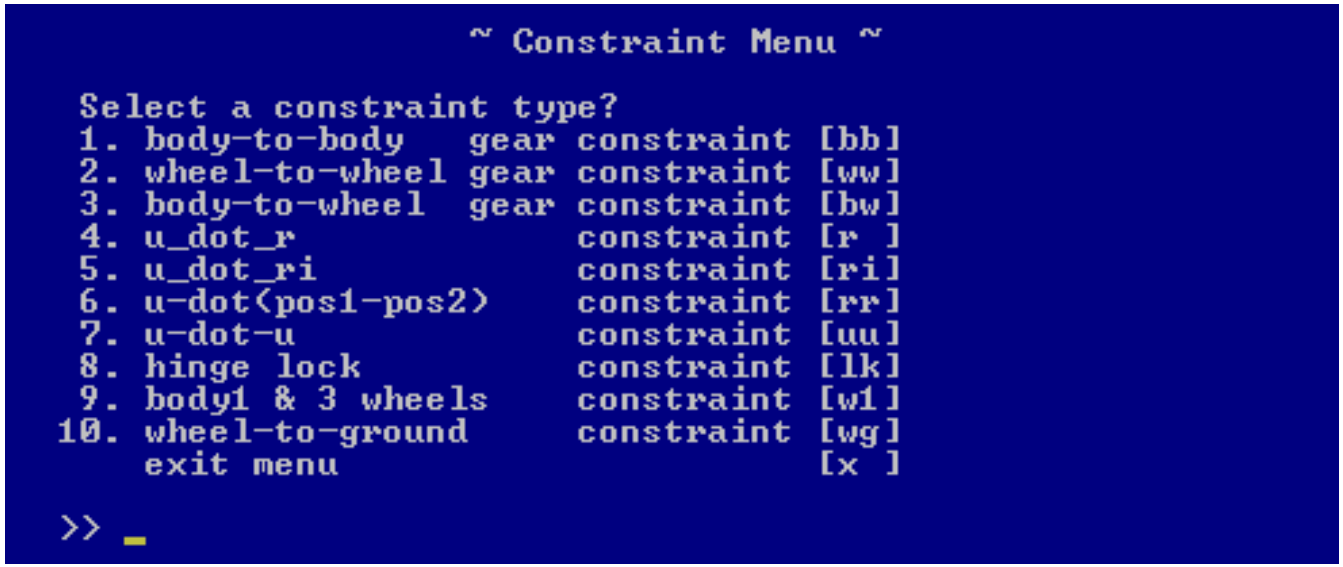
- OneCylinder has one position constraint: DotDelr
    a. Locks piston to end of Rod via pmkr’s p1 and p2

- cnj.name= constraint name
- cnj.ic= 1 means enable cnj at t=0, 0 means otherwise
- cnj.ln=number of cn equations for cnj
- cnj.(b1, b2)= two bodies involved in cnj
- cnj.(f1, f2)= two forces involved in cnj
- cnj.(d1, d2,d3)= unit vectors involved in cnj
- cnj.(p1, p2)= position markers involved in cnj

- Type ‘add’ to add a constraint
- Type ‘rem<j>’ to remove cnj
- Type ‘edit<j>’ to edit cnj
- Type ‘body<j>’ to edit cnj.(b1,b2)
- Type ‘pmkr<j>’ to edit cnj.(p1,p2)
- Type ‘frc<j>’ to edit cnj.(f1,f2)
- Type ‘dmkr<j>’ to edit cnj.(d1,d2,d3)
- Type ‘ic<j>’ to toggle cnj.ic between 0 and 1

- Type ‘help’ to get definitions of data and commands
- Type ‘x’ to exit menu

- See constraint selection menu, type ‘add’ from Constraint Menu



- Type ‘rr’ to select cn(1)=u_dot(pos1-pos2)=0 constraint
- Complete the dialog with cn1.(p1,p2)=(1,2), cn1.(d1,d2)=(1,2)
- p1, p2 are position markers 1 and 2, d1, d2 are direction markers 1 and 2
- Type ‘x’ to exit this menu
- Type ‘x’ again to exit Constraint Menu

Force Menu

- Need specify a force for combustion acting on piston
- Type 'force' from Model Menus page to see force summary
- f1 represents the combustion force on the pistion (p=4)
- f1.mag supplied by control system at run time (t=2)

```
                        ~ Force Menu ~

  no. of forces:   1
          units: FPS
  system CM pos:          .0000000E+00   .1851852E-02   .0000000E+00
  f_ref point  :          .0000000E+00   .0000000E+00   .0000000E+00

  fmag   :         1.000
  funit  :       .000000    1.000000      .000000
  fpos   :          .000         .000         .000
  ftq    :          .000         .000         .000
  az-axis:   1

  idx name       p t  c ---fmag--- ----fx---- ----fy---- ----fz----
=>  1 f1         4 2  1       1.000        .000       1.000        .000

  [ sel   idx   name  par   type units  fpos  fvec  fmag ]
  [ xFace yFace zFace refp  azel azAxis coord shift      ]
  [ sumry vec   rae   pos   rxf  fsum   tqsum            ]
  [ add   addF  copy  order rem  help   save  x          ]
  >
```

Force Menu Data & Commands

- f1.parent= b4 (piston)
- f1.type=2 means f1.mag is supplied at simx3r.dll.input
- f1.fvec= [0 1 0] , stroke axis in b1 frame
- f1.fmag= 1, this is just temporary for buildx session
- f1.fpos=[0 0 0], means f1 is applied at b4.origin

- Type 'add<j>' to add external forces to bj
- Type 'rem<j>' to remove force(j)
- Type 'name<j>' to edit force(j).name
- Type 'type<j>' to edit force(j).type={1,2 or 3}
- Type 'fvec<j>' to edit force(j).fvec
- Type 'fpos<j>' to edit force(j).fpos
- Type 'pos' to show all force impact positions in b1 frame
- Type 'rxf' to show torque of all forces abour f_r_ref in b1 frame
- Type 'help' to get definitions of data and commands
- Type 'x' to exit menu

Gravity Menu

- Need to define gravity in workspace
- Type 'grav' from Model Menus page to open gravity menu (see below)
- Gravity accel is defined by [gx gy gz]

```
                    ~ xmr Gravity Menu ~
> units       (U) =  FPS
> sysPos          =  .000000000E+00  .185185185E-02  .000000000E+00
> sysVel          =  .000000000E+00  .000000000E+00  .000000000E+00

> refPos          =  .000000000E+00  .000000000E+00  .000000000E+00
> refVel          =  .000000000E+00  .000000000E+00  .000000000E+00

    gravity (down):
> gx gy gz        =  .000000000E+00 -.322000000E+02  .000000000E+00

> sysacc flag     =  0
> gravity flag    =  0  <fixed for xmr>

[spos  svel  rpos  rvel  grav  sflag  gflag  units  opt ]
[save  help  x                                         ]
>
```

Gravity Menu Commands

- Type 'spos’ to edit orbit position in workspace coordinates
- Type 'svel’ to edit total velocity in workspace coordinates
- Type 'rpos’ to edit b1.reference_position in workspace coordinates
- Type 'rvel’ to edit b1.reference_velocity in workspace coordinates
- Type 'grav’ to edit gravitational acceleration in worksapce coordinates
- Type 'units’ to change the units of coordinates and mass properties
- Type 'sflag’ to run simulation in prescribed(spos,svel) mode or in force determined(spos,svel) mode
- gflag=0 by default
- Type 'help’ to get definitions of data and commands
- Type 'x’ to exit menu
- Buildx automatically updates all menu parameters when one of them is altered, i.e. changing ‘rpos’ results in a new (spos)…etc.

Dynamics Input

- Need input actuation signals to move modeled mechanism
- OneCylindeEnginer has the following input list

```
  Udata list:

   1) htqax,2          ; 2) xf,1
```

- htqax,2: crank torque arising from starter motor and run time friction
- xf,1: combustion chamber force during compression and combustion phases

Input Menu Commands

- Type 'add’ to add new variables to the end of udata list
- Type 'add<j>’ to insert new variables at udata(j)
- Type 'rem’ to remove a group of variables
- Type 'rem<j>’ to remove udata(j)
- Type 'chg<j>’ to change udata(j)
- Type 'len’ to see ordinal position of udata and their length
- Type 'x’ to exit udata menu

- A variable selection menu appears on commands {add, chg}
- Type 'sel<j>’ to select var(j) to add or chg
- Type 'x’ to return to udata menu

Dynamics Output

- Need output motion signals required by the control system
- OneCylindeEnginer has the following output list

```
 Ydata list:

  1) ANGLE,2          ¦ 2) ANGLE,3          ¦ 3) hposrax,4
  4) hvelrax,4        ¦ 5) wrelax,2
```

- angle,2: crank shaft angle
- angle,3: rod angle wrt crank shaft
- hposrax,4: piston displacement
- hvelrax,4: piston displacement rate
- wrelax,2: crank shaft angular rate

Output Menu Commands

- Type 'add<j>’ to insert new variables at ydata(j)
- Type 'rem’ to remove a group of variables
- Type 'rem<j>’ to remove ydata(j)
- Type 'chg<j>’ to change ydata(j)
- Type 'len’ to see ordinal position of udata and their length
- Type 'x’ to exit ydata menu

- A variable selection menu appears on commands {add, chg}
- Type 'sel<j>’ to select var(j) to add or chg
- Type 'x’ to return to ydata menu

Plot Data

- Need save plot data sampled every plotdt sec to plotfile
- Plot data (odata) for this example are

```
Odata list:

 1) ANGLE,2          | 2) ANGLE,3          | 3) WRELAX,2
 4) WRELAX,3         | 5) hposrax,4        | 6) hvelrax,4
 7) HTQAX,2          | 8) HTQAX,3          | 9) hfrcax,4
10) SYSHMOM          |11) SYSPOS           |12) SYSVEL
13) cnf,1            |14) mkrpos,1         |15) mkrpos,2
16) mkrlos,1         |17) mkrlos,2         |18) xf,1
19) ctorque,2        |20) cforce,4         |21) ctorque,5
22) angle,5          |23) wrelax,5         |24) htqax,5
25) ctorque,5        |26) cnrows           |27) w,2
28) w,5
```

- angle,2: crank shaft angle …
- htqax,2: crank shaft torque
- hfrcrax,4: piston force
- cnf,1: constraint 1 error …
- w5: load speed

Plot Menu Commands

- Type 'add’ to add new variables to the end of odata list
- Type 'add<j>’ to insert new variables at odata(j)
- Type 'rem’ to remove a group of variables
- Type 'rem<j>’ to remove odata(j)
- Type 'chg<j>’ to change odata(j)
- Type 'len’ to see ordinal position of udata and their length
- Type ‘Simplot’ to invoke Simplot Menu to build Simplot1.m
- Type 'x’ to exit odata menu

- A variable selection menu appears on commands {add, chg}
- Type 'sel<j>’ to select var(j) to add or chg
- Type 'x’ to return to odata menu

Save Model Data

- Need save data to a model_file after making model changes
- Type 'save' from Model Menus page (page 11) and complete the save dialog as follows

```
[body   force  torque  pmkr  dmkr  input  output plot        ]
[simplot flex  jnt cnx wheel accel gyro   grav   sunPos     ]
[times  vmass  pmass  discrt  ode  switch states sumry units]
[compute  cn   tree(f/t/p/d)       cgen   help   save  x     ]
> save

Commands:
1. save model data to oneCylinder.txt
2. save model data to another file
3. Cancel save
Select 1:3? 1

model data has been saved to oneCylinder.txt
```

- On hitting return, current model data would have been saved to OneCylinder.txt
- Try option 2

Simplot

- Type 'Simplot' from Model Menus (page 13) or plot menu to define plot script for post-sim viewing in Matlab (page 33)
- All plot data were selected in plot menu
- A. steps from Simplot menu:
  1. Type 'add' to add figures and respond with '3' to create 3 figures
  2. Type 'title1' to set figure (1) title: i.e. reply with 'system'
  3. Type 'vars1' to define variables to be plotted in fig 1. this opens the plot variables page
- B. steps from vars menu:
  1. Type 'addv9' and reply with 3to add 3 variables starting with the variable(10), syshmom
  2. Type 'addv26' and reply with 1 to add 'cnrows' to the variable list
  3. Type 'addp' and respond with '1,4' to create four subplots for figure(1)
  4. Type 'format' and respond with '2,2' to plot 4 subplots in 2 rows and 2 columns format
  5. Type 'x' to go back to Simplot menu
- Repeat steps A.2, A.3 and all B steps with proper indexing to define other figures for sat3w2a
- Final steps from Simplot menu:
  1. Type 'save' and reply with 'Simplot1.txt' to save Simplot data to Simplot1.txt
  2. Type 'make' to create Simplot1.m, see completion message
  3. Type 'x' to exit Simplot menu. Simplot1.m is ready.

Exit Buildx

- 3 ways to exit buildx:
  - go to Model Menus and type ‘q’ <return>
  - go to main menu and type ‘x’
  - click the ‘x’ on top right corner of the buildx window

- Note: buildx.exe does not save model data automatically. see save procedure on page 30

Q & A

- Can one add and delete bodies, wheels and forces?
  - Yes if you have enterprise license, and no if you have a project license
- Are all Project licenses restricted to object count {5, 0, 1}?
  - No, for example arm6x Project license has an object count of {7, 0, 0} and a unique parent-child relation between bodies
- Must all joints in the mechanism be 1 dof?
  - No. but the joint dof must be consistent with the intended mechanism
  - Use 'type<j>' to define the motion of bj relative to its parent
- Can the engine block be mounted on a moving platform?
  - Yes, in that case one must insert a free body between ground and crank and call it the engine block
  - This engine block must have mass/inertia and some spring-damper forces to model dynamic reaction of chasis to engine motion
  - More complex chasis & suspension system are possible

- How can one see all the available input, output and plot variables when choosing them from udata, ydata and odata menus?
  - All available variable list is shown when one types ‘add’ command from the menu
  - Type 'defj’ to get the definition of variable(j) in that list
  - Type 'selj’ from the add menu to select variable(j) to the list

- How does one change the plot data sample period?
  - Type 'plotdt’ from the main menu or from the times menu to do that

- Why are ‘dt’ and other time specification in the times menu?
  - Those time specifications are not used for the Simulink applications, they are for the Fortran and C implementation of xsv01 engine

Part II Topics

- Key Files
- OneCylinder.mdl
- Twostroke.m
- Fourstroke.m
- Engine_ic.m
- Example1
- Example2
- Adjustable Sim Parameters
- Exercises
- Simulation Notes
- Summary

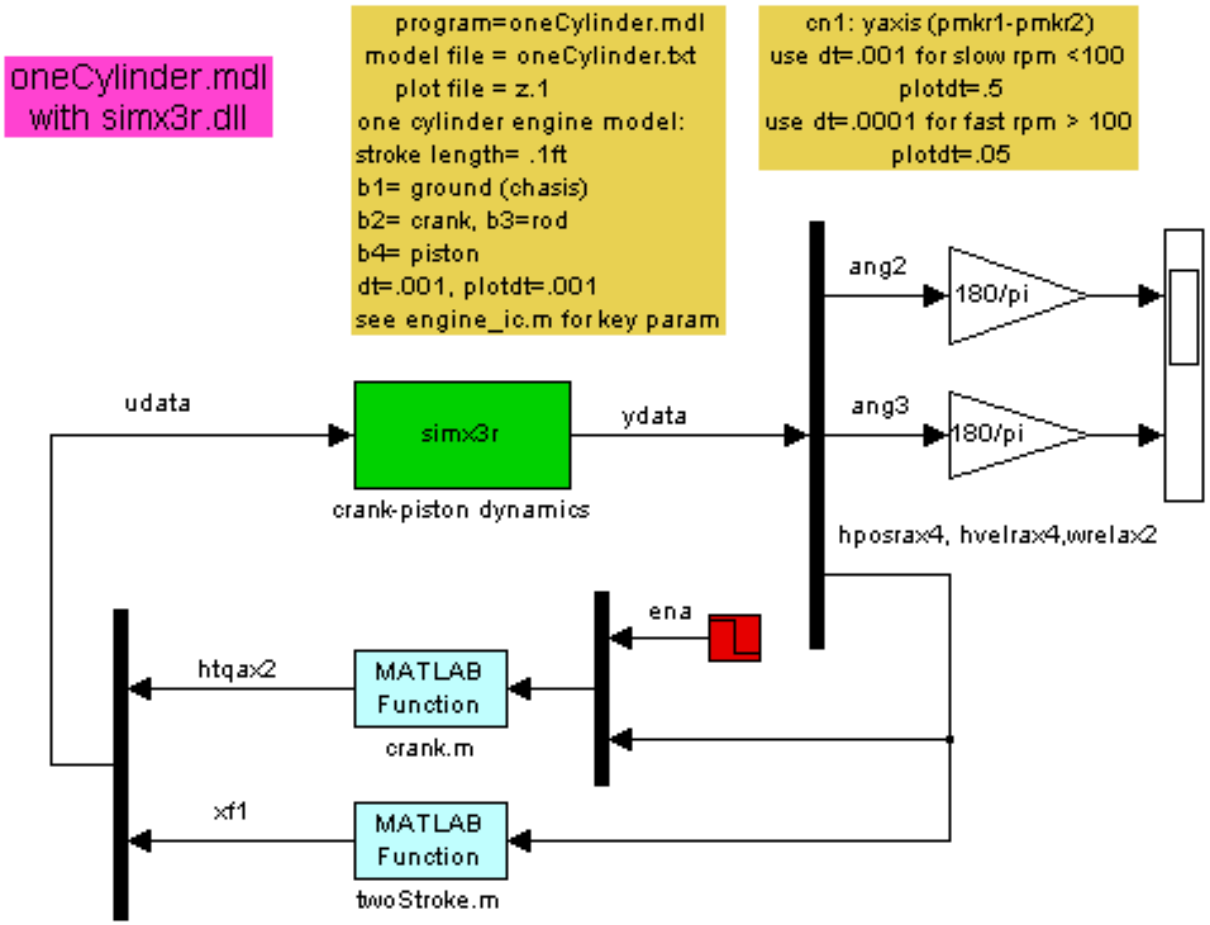
Key Files

- Simulation Program : OneCylinder.mdl
- Control Scripts : crank.m, twostroke.m, fourstroke.m
- Initialization Script: Engine_ic.m
- Simulation Engine : Simx3r.dll
- Main Input File : siminput.txt (points to sim1files.txt)
- Matlab Script to View Results: Simplot1.m
- Simulation Input/Output Files: Sim1files.txt
- Model File: OneCylinder.txt

OneCylinder.mdl

- Need OneCylinder.mdl to run an simx3r.dll based sim according to OneCylinder.txt in Simulink
- Sends ydata to control system and latter sends udata back to xsv01.dll
- Crank.m , twostroke.m, fourstroke.m are user supplied and application specific
- Go to c:\OneCylinder and click on OneCylinder.mdl to open it in Simulink workspace

OneCylinder.mdl with Twostroke.m



Controls

- Need 3 scripts to define forces and torque to actuate the engine model

- Crank.m :
  - In: ena, hposrax4, hposvelrax4, wrelax2
  - Out: htqax2
  - Provide initial crank torque (t < 3 sec)
  - Define the friction torque during normal operation

- Twostroke.m for running engine in two stroke mode:
  - In: ena, hposrax4, hposvelrax4, wrelax2
  - Out: hfrcax4
  - Defines compressive and combustive forces over two strokes

- Fourstroke.m for running engine in four stroke mode:
  - In: ena, hposrax4, hposvelrax4, wrelax2
  - Out: hfrcax4
  - Defines compressive and combustive forces over four strokes

Crank.m Code

```
function htqax2=crank(u)
global   frc_crank

ena =u(1)       ; % enable switch
dontcare=u(2:3) ;
wr2 =u(4)       ; % crank rate

if ena==1
   htqax2= .1    ; % initialize wr2(crank rate) to 600 rpm
else
   htqax2=-frc_crank*wr2;   % friction
end
```

Twostroke.m Code

- function y=combustion(u)
- global f_max_compress
- global frc_piston
- global f_tdc
- global stroke
- global phase
-
- posr =u(1); % piston position
- velr =u(2); % piston velocity
- wr   =u(3); % crank angular rate
-
- alfa = posr/stroke; % alfa is < 0 always:
  range 0:-1
- y   = 0;
-

```
if  phase==1
   if alfa > -.5 && velr > 0  % up stroke  (stroke 1)
      y= -f_max_compress*(alfa+.5)/.5;  % compression force
      if abs(alfa) < 0.001
         if abs(wr) > .1*pi/180 && f_tdc > 0
              phase=2;
         end
      end
   end
elseif phase==2
   if alfa > -.001 && velr > 0
      y= -f_max_compress*(alfa+.5)/.5;  % compression force
   else
    if alfa > -.5
      if velr < 0           % down stroke (stroke 2)
          y= -f_tdc*(1-((alfa+.1)/.4)^2)-f_max_compress;   % combustion
      else
         phase=1;
      end
    end
   end
end
y= y -frc_piston*velr;
```

Fourstroke.m Code

```
function y=fourStrokeCombustion(u)
global f_max_compress
global f_exhaust
global f_intake
global frc_piston
global f_tdc
global stroke
global phase

posr  =u(1); % piston position
velr  =u(2); % piston velocity
wr    =u(3); % crank angular rate

alfa  = posr/stroke;    % alfa is < 0 always: range 0:-1
y     = 0;
```

```
if    phase==1
   if alfa > -.5 && velr > 0          % up stroke (stroke 1)
      y= -f_max_compress*(alfa+.5)/.5;  % compression force
      if abs(alfa) < 0.001
         if abs(wr) > .1*pi/180 && f_tdc > 0
              phase=2;
         end
      end
   end
elseif phase==2
   if alfa > -.001 && velr > 0
      y= -f_max_compress*(alfa+.5)/.5;  % compression force
   else
     if alfa > -.5
      if velr < 0                  % down stroke (stroke 2)
          y= -f_tdc*(1-((alfa+.1)/.4)^2)-f_max_compress;   % combustion
      else
         phase=3;
      end
     end
   end
```

```
elseif phase==3
   if alfa > -.5 && velr > 0   % up stroke (stroke 3
      y= -f_exhaust;         % exhaust compression force
      if abs(alfa) < 0.001
         if abs(wr) > .1*pi/180
              phase=4;
         end
      end
   end
elseif phase==4
   if alfa > -.001 && velr > 0
      y= -f_exhaust;  % compression force
   else
     if alfa > -1.0           % stroke 4
      if velr < 0            % down stroke
         y= -f_intake*velr;   % suction force
      else
         phase=1;
      end
     end
   end
end
y= y -frc_piston*velr;
```

Initialization Script

- Need engine_ic.m to initialize parameters for control scripts through ‘global’
- Set File.Model Properties.Callbacks.InitFcn=‘engine_ic.m’ to call it at initialization

engine_ic.m listing

```
% engine parameters:
global cycle
global y
global ignition
global f_max_compress
global f_exhaust
global f_intake
global frc_piston
global frc_crank
global f_tdc
global stroke
global phase

cycle   =0;
y       =0;
ignition=0;
f_max_compress=0.5 ; %max compression force
f_exhaust     =0.1 ; %exhaust compress force
f_intake      =0.01; %intake suction force coeff
frc_piston=.001   ; %piston friction coeff
frc_crank =.001   ; %crank friction coeff
f_tdc     = 10    ; %max combustion force (tdc=top dead center)
stroke    = .1    ; %stroke length
phase     = 1     ; %upstroke
```

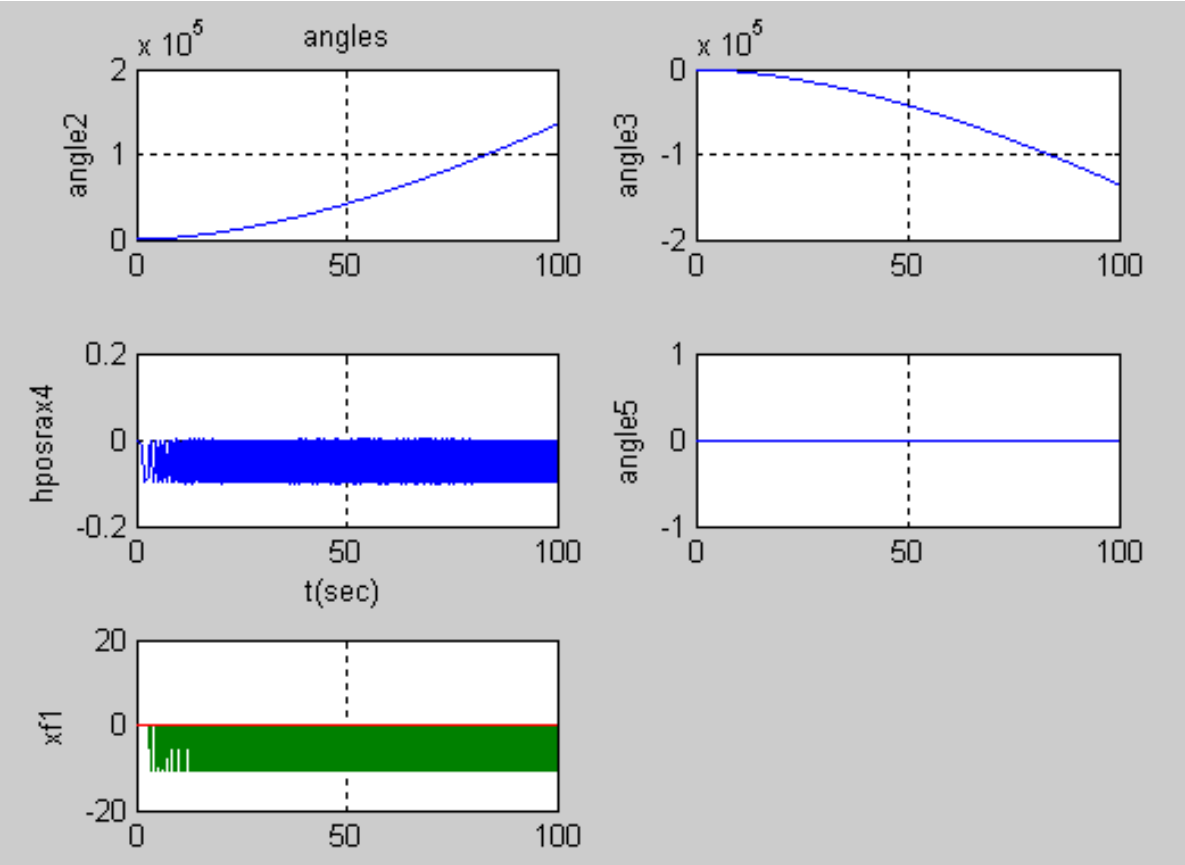
Example1

- All preparations up to here is for example1. it runs OneCylinder.mdl in two stroke mode.
- Click on c:\OneCylinder\OneCylinder.mdl to open it
- Define the xf1(piston force) in ~.mdl program by twostroke.m
- Use simulation dialog box to set configuration parameters as needed,i.e.:
  - solver options > step size, method
  - end time
- Click run button (black triangle) to run simulation

View Sim Results

- Type 'load z.1’ from Matlab window to read in sim result saved to z.1
- Type 'Simplot1(z)’ to view result
- See page 30 on how Simplot1.m was created for OneCylinder

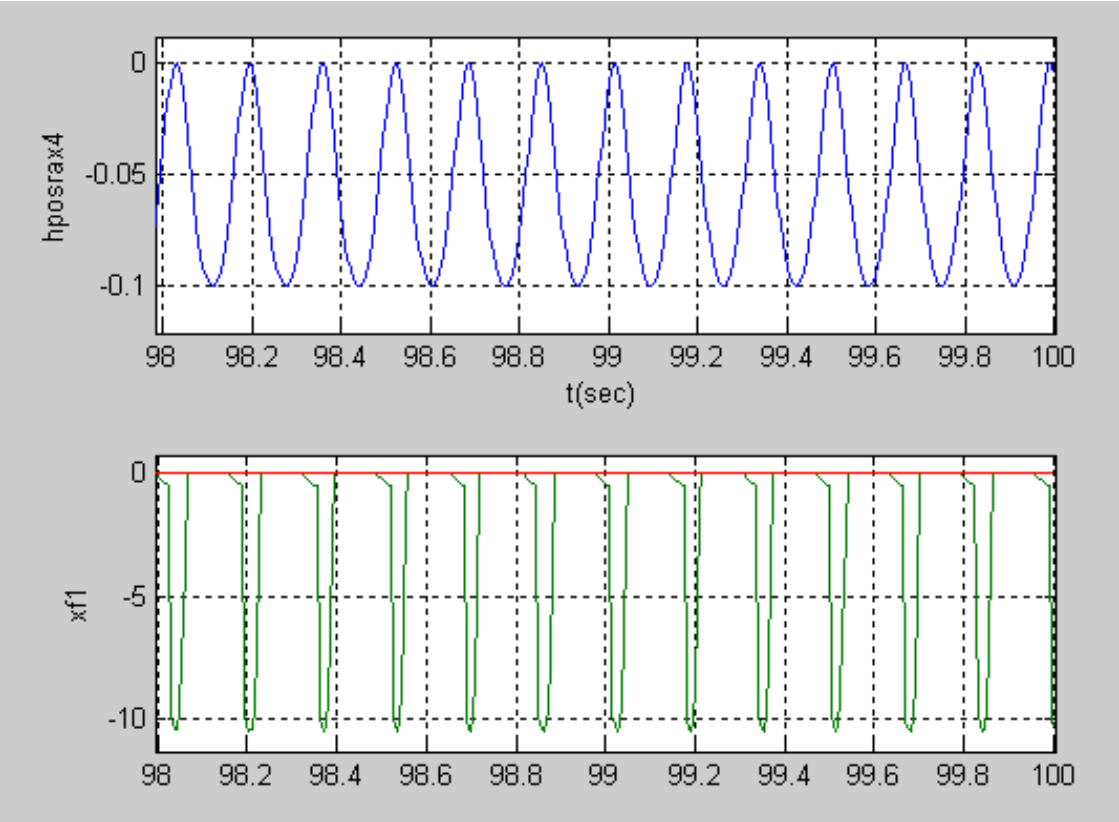
Fig.1 Engine Motion-1



- angle2= crank angle
- angle3= rod angle
- hposrax4= piston y motion
- angle5= load ang wrt crank
- xf1=force on piston

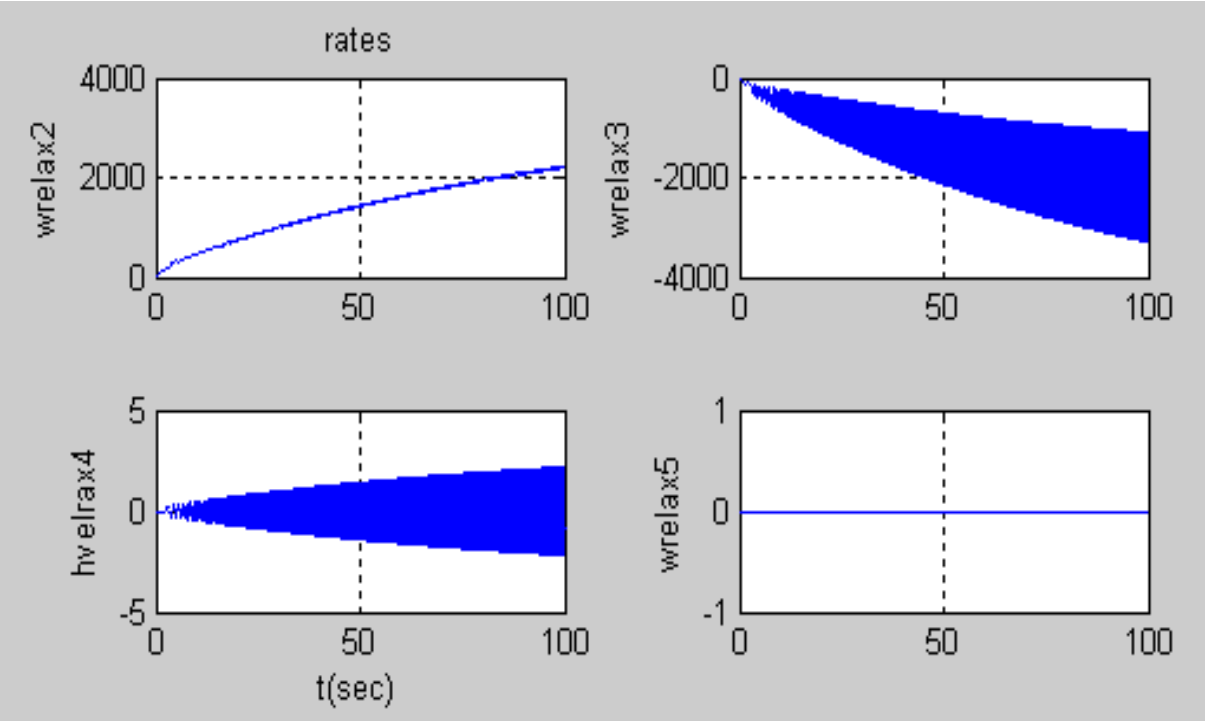
- hposrax lies in [0: -0.1]
- xf1 lies between [0, -10.5]

Fig.2 Hposrax4 vs Xf1 Expanded



- xf1 spikes on every down stroke

Fig.3 Rates-1



- wrelax2: crank spin rate
- wrelax3: rod ang rate
- hvelrax4: piston pos rate
- wrelax5: load rel rate

- @ t=100, crank rate is 2215 d/s= 369 rpm

Example2

- Example2 runs OneCylinder.mdl in four stroke mode
- All setup is same as that for example1
- Click on c:\OneCylinder\OneCylinder.mdl to open it
- Define the xf1(piston force) in ~.mdl program by fourstroke.m
- Use simulation dialog box to set configuration parameters as needed,i.e.:
  - solver options > step size, method
  - end time
- Click the black triangle to run simulation
- Use Simplot1.m as in example1 to view results

OneCylinder.mdl with Fourstroke.m

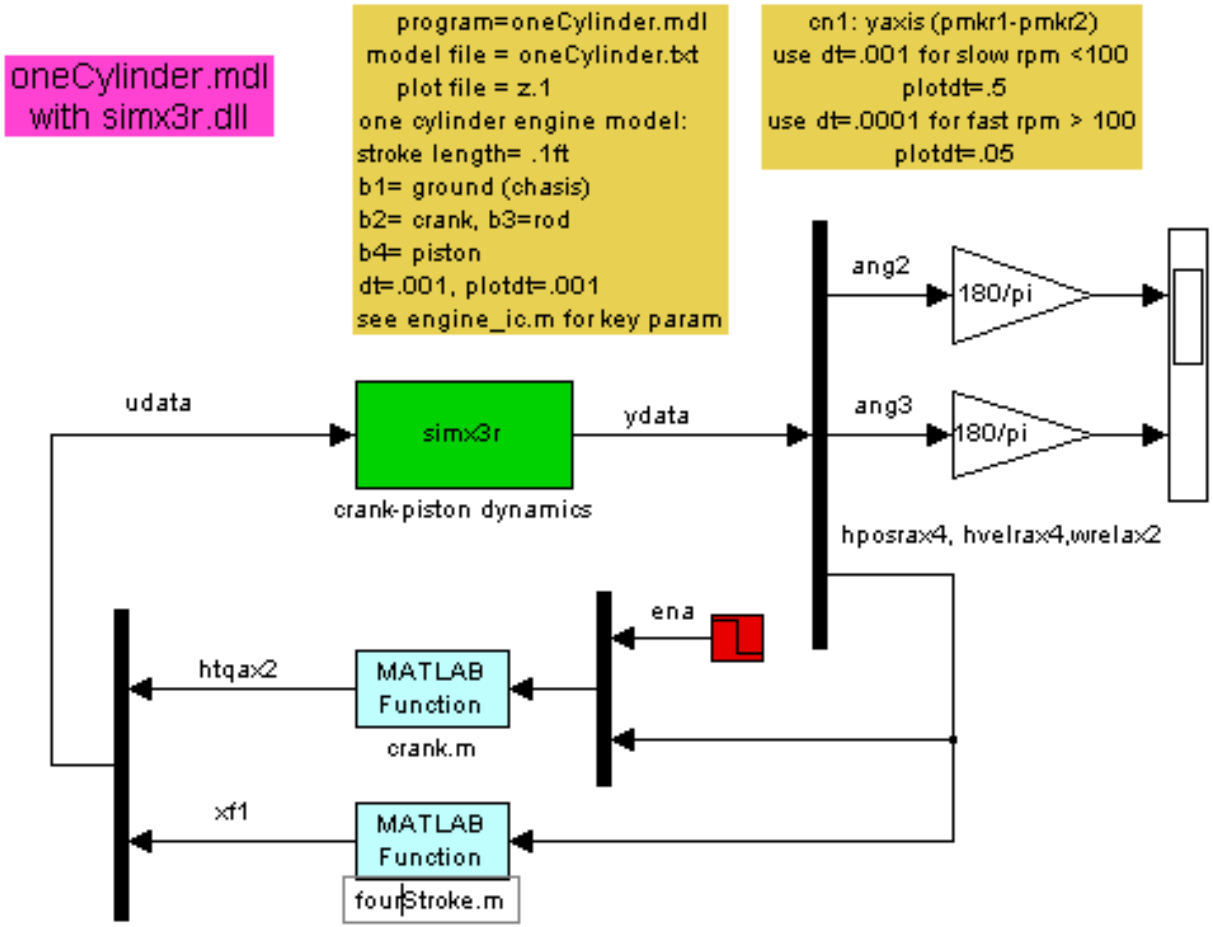
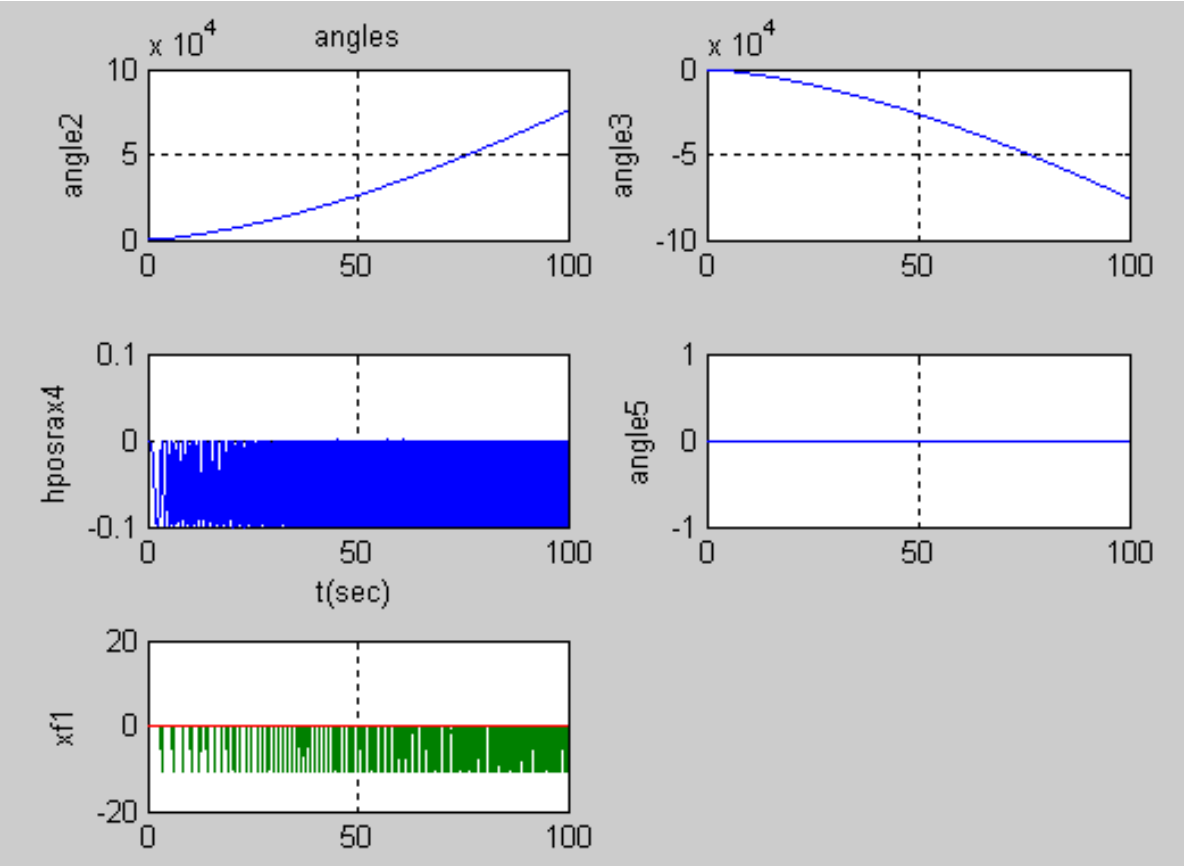


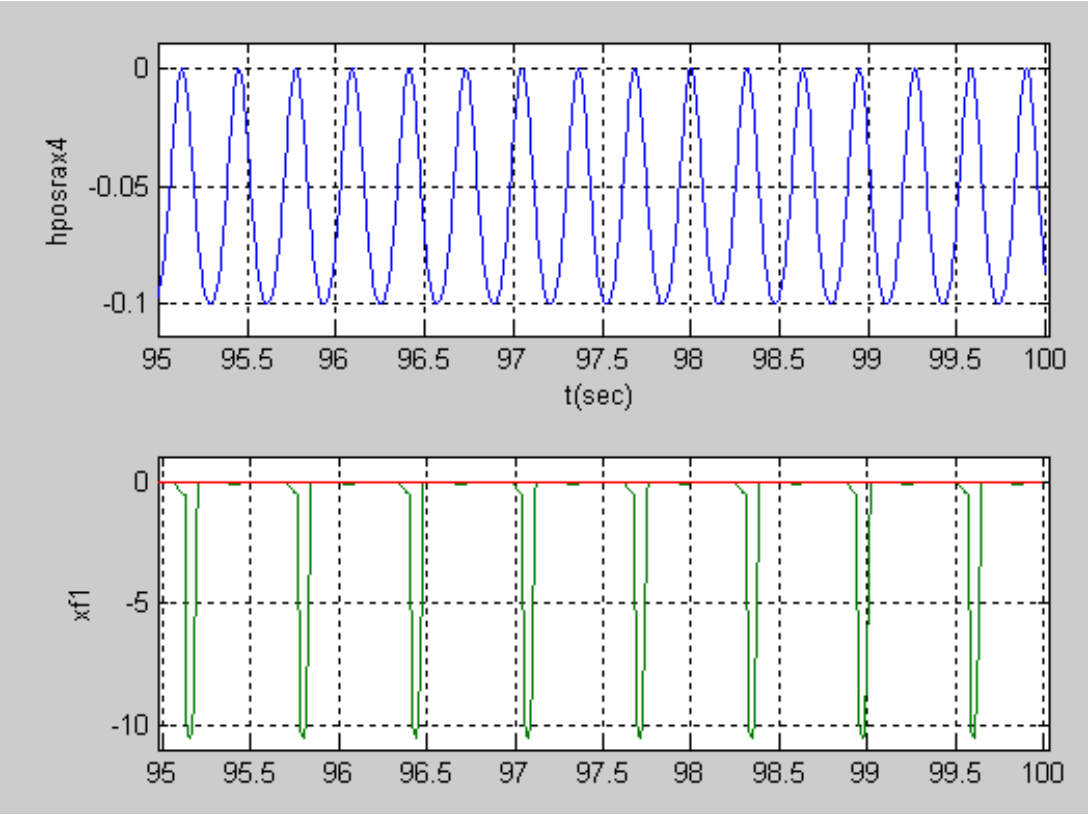
Fig.4 Engine Motion-2



- angle2= crank angle
- angle3= rod angle
- hposrax4= piston y motion
- angle5= load ang wrt crank
- xf1=force on piston

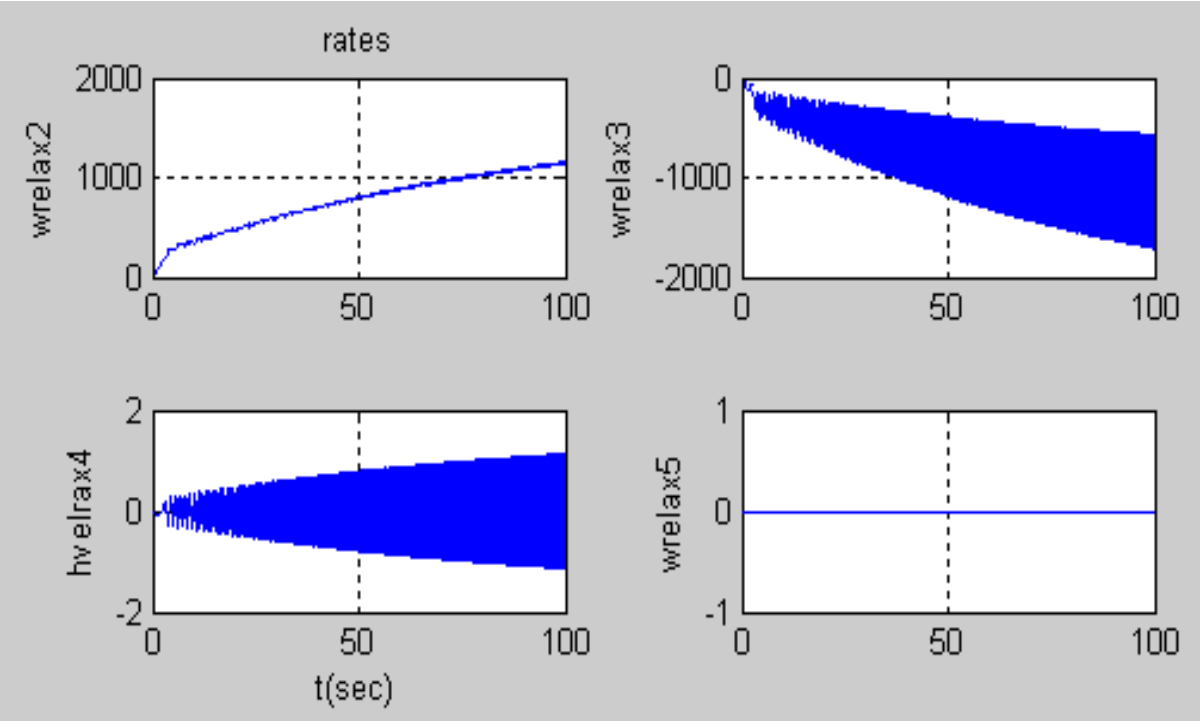
- hposrax lies in [0: -0.1]
- xf1 lies between [0, -10.5]

Fig.5 Hposrax4 vs Xf1 Expanded



- xf1 spikes on every other down stroke

Fig.6 Rates-2



- wrelax2: crank spin rate
- wrelax3: rod ang rate
- hvelrax4: piston pos rate
- wrelax5: load rel rate

- @ t=100, crank rate is 1150 d/s= 191 rpm

Adjustable Sim Parameters

- Dtplot: plot data sample period
    a. Type 'plotdt' from the main menu page to edit

- Dt: simulation integration step size
    a. Change it under the 'simulation' button in the Simulink program
    b. dt=0.01 for arm6a.mdl, try larger stepsize until response diverges
    c. Generally choose dt < 1/(5*f) where f=freq of fastest process in model

- T: simulation period
    a. Change 'stop time' under the 'simulation' button in the Simulink program

- Integration method:
    a. examples given here works with ode2 and rk4

Exercises

actions	parameters	reference
change mass property	mass, inr, svec, dvec	pages:16-21
change initial condition	ang,wrel,wrelax, dcm0	pages:16,17
change joint type, axis	type, axis	pages:16,17
change gravity	gx, gy, gz	pages:29
add /remove Bodies,constraints,forces*		pages:16,26,29
modify input (udata)		pages:33-34
modify output (ydata)		pages:35-36
modify plot (odata)		pages:37-39
modify Simplot1.m		page:40

* body, wheels, and forces add/rem are not for project licenses

actions	changes	control system config
design your own combustion force and friction models	•may need new ydata	•could be different from given control system

Simulation Notes

Subject	Arm6x	comments
gforces	comment applies to all bodies in OneCylindeEnginer	gravity forces are auto-computed for a given mechanism by simx3r.dll; so, no xf's are needed for them
wheels	No reaction wheels needed for OneCylindeEnginer	wheels can be invoked to represent angular momentum due to high speed DC motors
geometry	OneCylindeEnginer has a crank-rod-piston chain of 3 bodies with engine block fixed to workspace and a 1dof joint between bodies	position and orientation of body parts are defined by their dvec, svec ,dcm0 and joint coordinates
b1.dcm0	comment applies here	b1.dcm0 is the orientation of base body in workspace
mass, inertia	These can be zero for b1 if it is locked to workspace	Needs be defined for each bj. Can be zero for internal bodies to represent massless joints

Summary

- Two examples given here demonstrated that simx3r.dll can simulate a one cylinder engine operating in one stroke or two stroke mode.
- The value of OneCylinderEngine is that its mass property can be varied to fit the engine of interest. User can define a variety of combustion force, piston and crank frictions to design and test the chosen engine mass property and load. Another perspective is that OneCylinderEngine can be used to examine the trade space in the design of a single cylinder engine, its mass property, operations and control system.