

# SEP1simc Manual

Concurrent Dynamics International

May 2014

# Objectives

- Part I:
  - Build a model file (SEP1simc.txt ) to simulate a satellite separating from a mothership. The satellite has 4 reaction wheels, 8 jets and two arrays
  - Build simplot1.m to view sim results
- Part II:
  - Program and compile control.dll to run with xsv01.exe according to SEP1simc.txt
  - Multiple Events Simulation Example:
    - Mothership and the attached satellite rotate at a some rate prior to separation. Satellite rate nulled after separation and its arrays deployed ACS is then enabled to maintain LVLH attitude.

# License Restrictions

License type	Buildx.exe	Xsv01.dll
Enterprise	none	none
Project	Must stay with the object count specified by license gflag <=12	Runs with model_files with license specified object count gflag <=12

- Project license permits satellite simulations of satellites 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.
- Project license permits gravity model with gflag <=12 (see page 43)

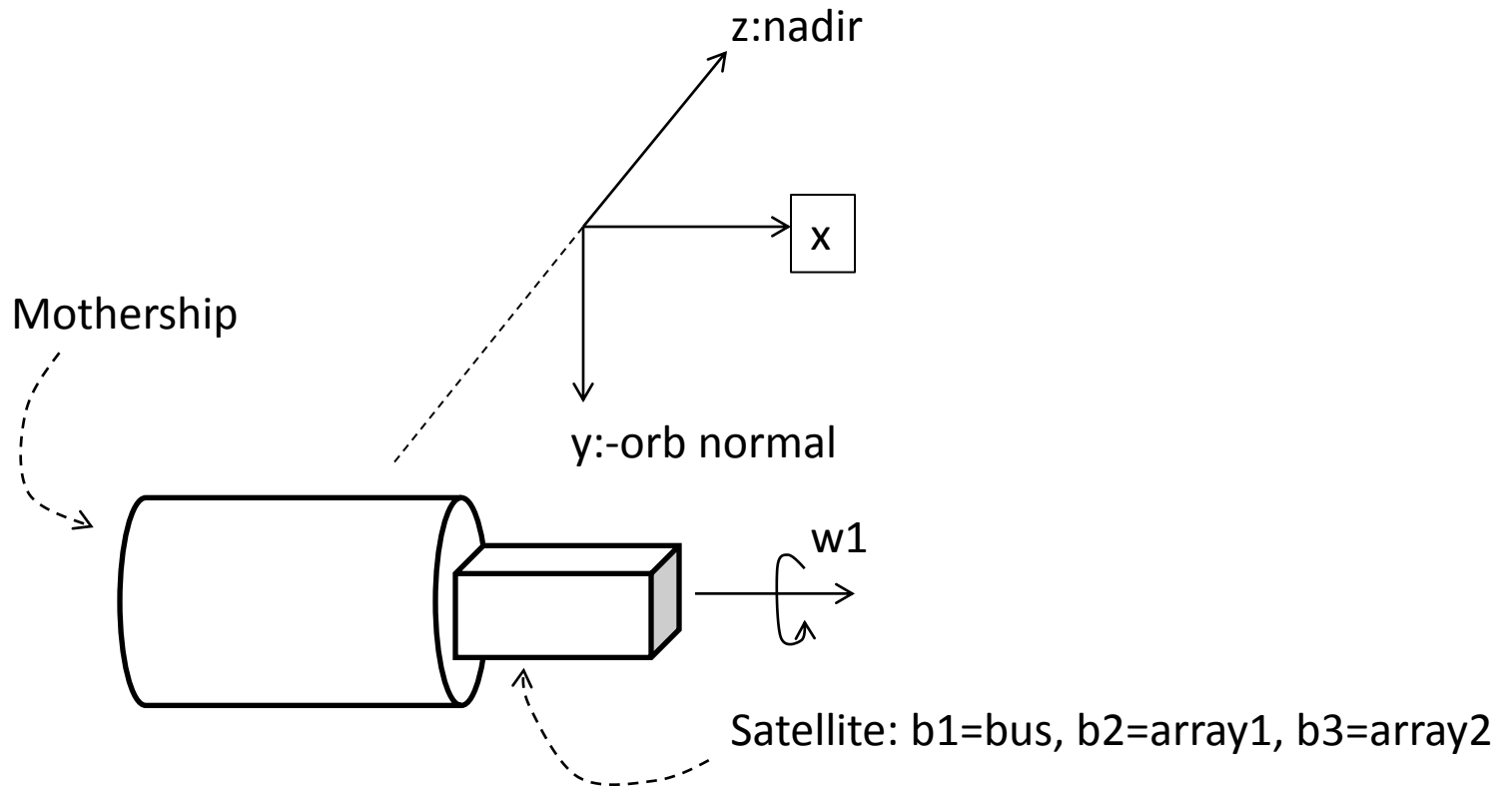
# How to Use This Manual

- This SEP1simc manual is written for Enterprise license users where no restrictions are placed on the object counts {body, wheels, forces} in creating models.
- SEP1simc.txt is a seed model\_file for Enterprise license users to create other models such as two satellites in proximity operations
- CDI has many seed models to expedite the development of more complex spacecrafts, i.e. those that use CMG's for ACS, multi-paneled solar arrays
- This manual is applicable to Project license users whose model object count is {3, 4, 8} as SEP1sim

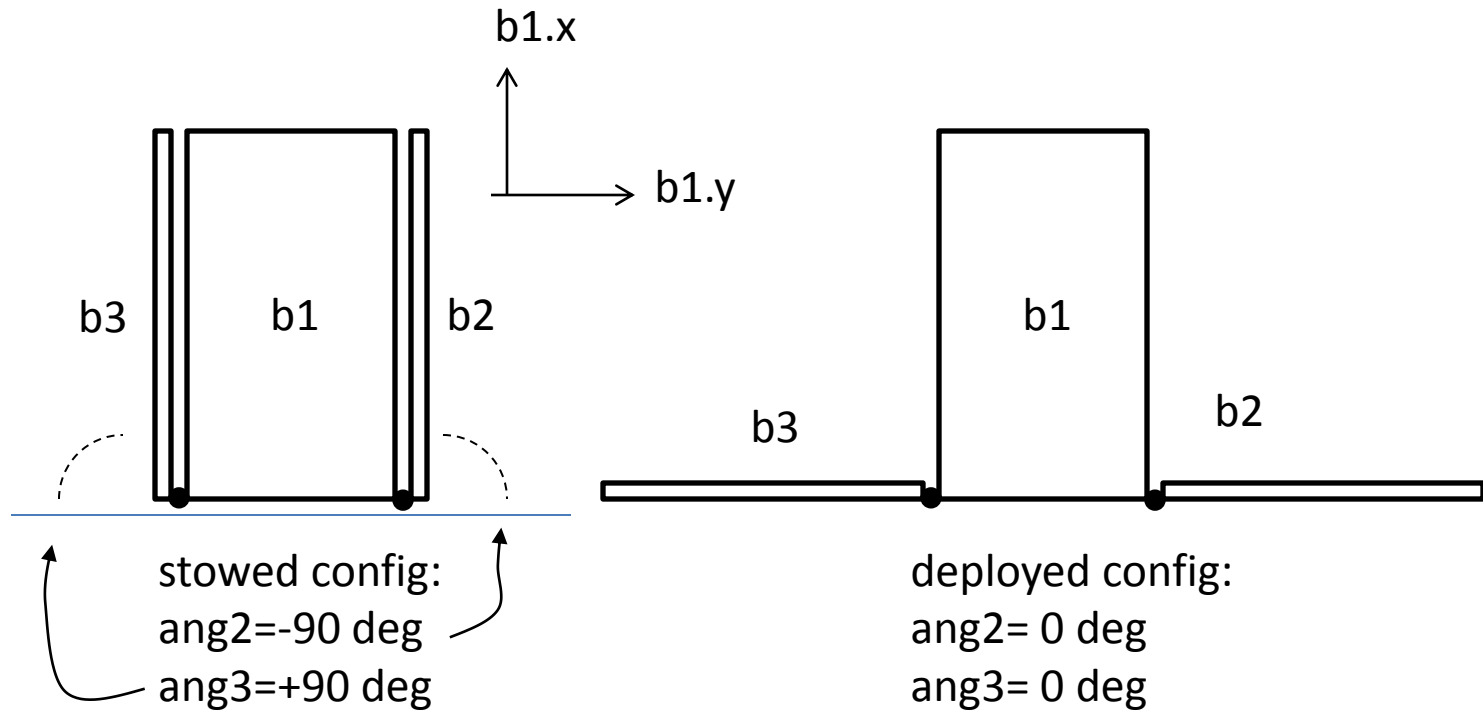
# Part I Topics

- Physical Model
- Buildx Tasks
- Key Files
- Main Menu
- Model Menus
- Body Menu
- B1 Page
- VMASS Menu
- Stage Menu
- Body Actuation Signals
- Wheel Menu
- Force Menu
- Gravity Menu
- Constraint Menu
- Lock Constraint
- Dynamics Input
- Dynamics Output
- Plot Data
- Simplot
- Save Model
- Exit Buildx
- Q & A

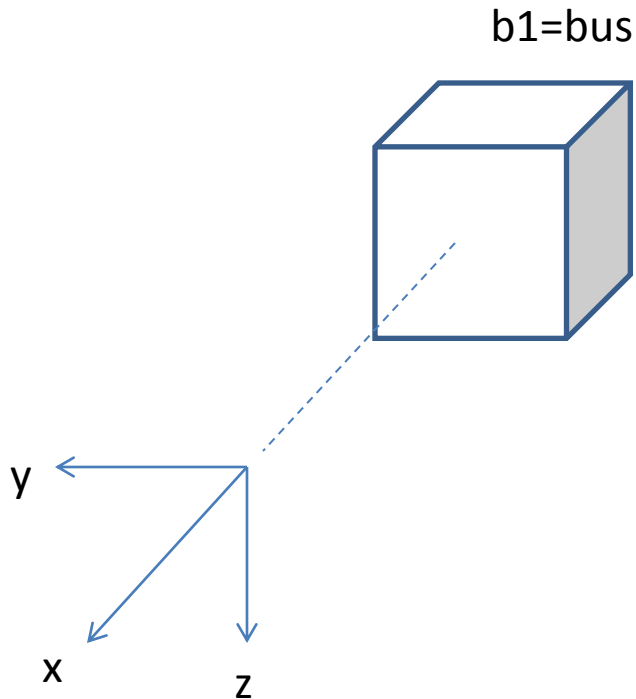
# SEP1simc Model Init Config



# Satellite in Stowed and Deployed Configurations



# Reaction Wheels and Jets on B1



reaction wheels axes on b1:

$$w1=[ 1 \ -1 \ 1]/\text{sqrt}(3)$$

$$w2=[ 1 \ 1 \ 1]/\text{sqrt}(3)$$

$$w3=[-1 \ 1 \ 1]/\text{sqrt}(3)$$

$$w4=[-1 \ -1 \ 1]/\text{sqrt}(3)$$

jet forces on b1:

$$f1.\text{pos}=[ -3 \ 3 \ 0] , f1.\text{vec}=[1 \ 0 \ 0]$$

$$f2.\text{pos}=[ -3 \ -3 \ 0] , f2.\text{vec}=[1 \ 0 \ 0]$$

$$f3.\text{pos}=[ -3 \ 0 \ 3] , f3.\text{vec}=[1 \ 0 \ 0]$$

$$f4.\text{pos}=[ -3 \ 0 \ -3] , f4.\text{vec}=[1 \ 0 \ 0]$$

$$f5.\text{pos}=[ 0 \ 3 \ 3] , f5.\text{vec}=[0 \ -1 \ 0]$$

$$f6.\text{pos}=[ 0 \ 3 \ -3] , f6.\text{vec}=[0 \ -1 \ 0]$$

$$f7.\text{pos}=[ 0 \ 0 \ 0] , f7.\text{vec}=[0 \ 0 \ 0]$$

$$f8.\text{pos}=[ 0 \ 0 \ 0] , f8.\text{vec}=[0 \ 0 \ 0]$$



# On the Simulation

- We are building a model file in Part I to support a dynamics simulation of a vehicle separating from its mothership in an orbital environment. Setup includes definition of mass property of bodies and wheels, of jet forces, and orbit. It also includes the definition of dynamics input and output signals required by the control system.
- Part II shows how to build the mdl file to carry out the simulation in Simulink. Two examples are presented. Key feature of the examples is the handling of a discontinuity at separation as explained below.
- Before the separation,  $t < t_{sep}$ , b1 represents the combined Mothership and payload satellite.
- At the separation instant,  $t=t_{sep}$ , system becomes the satellite only. Effectively, the simulation reinitializes itself at  $t=t_{sep}$  with the following conditions:
  1. System angular momentum is that of satellite only
  2. System cm position and velocity in the inertial space is that of the satellite
  3. All joint coordinates and rates in the satellite remain unchanged

- Thus, at  $t=t_{sep}$ , the following parameters are discontinuous:
  1. system angular momentum
  2. system cm position and velocity in inertial frame
  3. b1 mass property changes from the pre-separation to post-separation values
- During the simulation, a message is sent to `sim1_message.txt` file recording the model parameters that changed at separation time. These include: `syspos`, `sysvel`, `mass1`, `inr1`, `sysh` and others

# Buildx Tasks

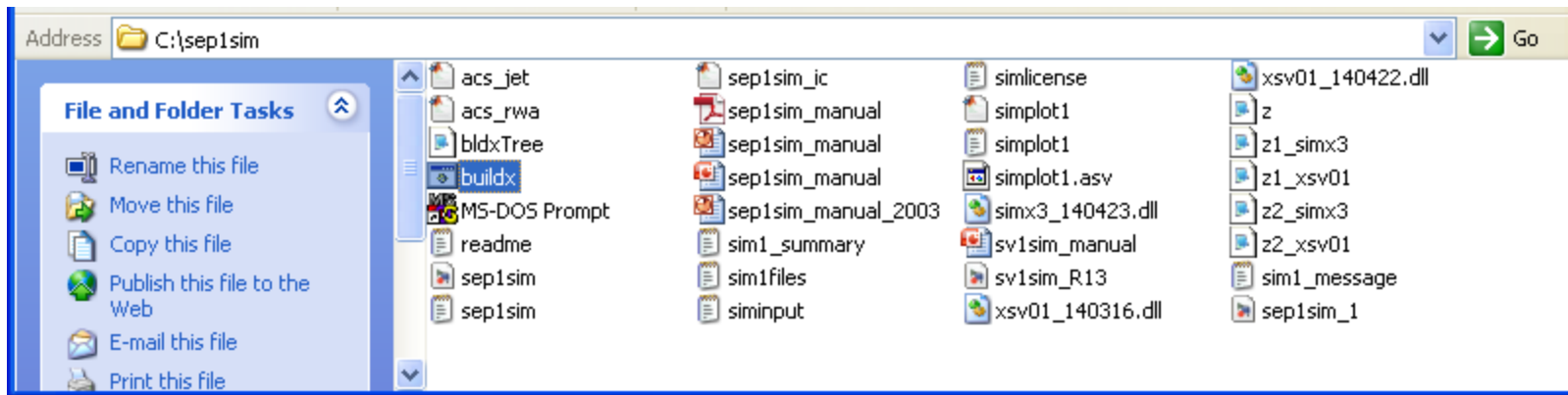
- Build a **model file** to define a satellite simulation:
  - {mass property,model connectivity, degree of freedom,xsv01.dll input/output,plot data,gravity model,...etc}
- Edit & browse the attributes of satellite components
  - {bodies,wheels,forces,markers,gravity,initial state,...}
- Build a Matlab plot script, `simplot1.m`, to view sim results

# Key Files in c:\SEP1sim

- Buildx.bat='..\Buildx'
- Input files router: siminput.txt
- Working files: sim1files.txt
- Simplot text: i.e. simplot1.txt
- License: simlicense.txt

# Starting Buildx.exe

- Click c:\SEP1sim\Buildx.bat to start Buildx.exe and see the Main Menu



# Main Menu

- Shows all working files defined in the siminputfile

```
*****
*          BBBB  U    U  I  L    DDDD  X    X          *
*          B   B  U    U  I  L    D   D  X  X          *
*          BBBB  U    U  I  L    D   D    X          *
*          B   B  U    U  I  L    D   D  X  X          *
*          BBBB  UUU   I  LLLL DDDD  X    X          *
*          ~~~~~~                                     *
*                               xsv version 1.0         *
*                               copyright 2014          *
*          concurrent dynamics international           *
*          *****                                     *

simInputFile:  sim1files.txt          < ENTERPRISE

  Model file < sep1sim.txt
  Plot file  > z.1
Summary file > sim1_summary.txt
Message file > sim1_message.txt
  plotDt    = .200000E+01

[ xsv   open   save   model   plot   plotDt ]
[ sumry mssg  reset                help     x   ]
>
```

- Type 'xsv' to go to Model Menus

# Model Menu

- See model parts size or choose menus to edit/browse

```
~ Model Menu ~
System Graph:
b1<B>+-b2<A>
  |
  +-b3<A>
  |
  +-w[1,2,3,4]

total bodies:      7      ; reg. bodies& wheels:    3,  4
ext. forces,torque: 8,  0 ; pos.& dir markers:    0,  0
system units:      FPS    ; constraints:          2
sflag,gflag:      1, 10 ; input (param,size): 13, 13
dscrt,odes:       0,  0 ; output(parmm,size):  4,  8
accels,gyros:     0,  0 ; plot (parmm,size):  25, 50
vmass,pmass:      1,  0 ; swiches,states:     0, 21

License: ENTERPRISE

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

- SEP1simc has 3 regular body, 4 wheels and 8 forces

# Sep1simc Model Items

Defined Items	Menu	Parameters/data
3 body	Body	Mass, inr, svec,dvec, dcm0, axis,ang,wrel,parent,type
4 wheels	Wheel	Winr, axis, wspd, parent, type
8 forces	Force	Fvec,fpos, parent,type
2 constraints	CN	2 lock constraints for array deployment
Plant input	Input	Xv01.dll input data list
Plant output	Output	Xsv01.dll output data list
Plot data	Plot	Xsv01.dll plot data list
Orbit	Gravity	Vehicle's orbit position and velocity

- Attributes of the defined items can be seen by going to the Menus specified above. Use the Model Menu commands to do that.
- Part I gives the instructions on how to define the above parameters and data
- [If already familiar with Part I, go to Part II on page 64](#)



# Model Menu Commands

- Type 'body' to go to xsv.Body Menu
- Type 'force' to go to xsv.force menu
- Type 'grav' to go to xsv.Gravity Menu
- Type 'pmkr' to go to xsv.position marker menu
- Type 'dmkr' to go to xsv.direction marker menu
- Type 'input' to go to xsv.input menu
- Type 'output' to go to xsv.output menu
- Type 'plot' to go to xsv.Plot Menu
- Type 'simplot' to go to xsv.simPlot Menu
- Type 'time' to go to xsv.timing menu
- Type 'help' to get definition on data and commands
- Type 'x' to exit menu

# Body Menu

- See body summary, type 'body' from Model Menu
- SEP1simc has 3 bodies

```
~ Body Menu ~
no. of bodies : 3
sysh_eci      : -.551523E-04  .134245E+02  .287889E+02
sysh_b1      :  .317650E+02  .000000E+00  .000000E+00
syscm       : -.742574E+01  .000000E+00  .000000E+00

dvec =  .000000E+00  .000000E+00  .000000E+00
svec = -.750000E+01  .000000E+00  .000000E+00
hpos =  .000000E+00  .000000E+00  .000000E+00
rpos = -.750000E+01  .000000E+00  .000000E+00
wrel =  .100000E+01  .000000E+00  .000000E+00
inr  =  .180000E+04  .139500E+05  .139500E+05
      .000000E+00  .000000E+00  .000000E+00

idx name      pa  u  fl  vm  tp  ax  --  angle  -  ----mass----
=>  1  b1       0  FPS  0  *  B  x   .000  .200000E+03
    2  arrayS   1  FPS  0  -  A  z  -90.000 .100000E+01
    3  arrayN   1  FPS  0  -  A  z   90.000 .100000E+01

[ 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  doIc  save  help  zero  x   ]
> _
```

# Body Menu Data & Commands

- Sysheci system angular momentum in ECI frame
- Syshb1 sysheci in b1 frame
- Syscm system cm in b1 frame
- Dvec-inr position, rate and inertia of the body pointed by '=>' arrow
- Column data
- Pa parent body index
- U units of measure {FPS=English, MKS=Metric}
- Tp joint type {a-h}
- Ax local axis for 1 dof joints {x, y or z}
- Ang initial joint angle (deg)
- Mass body mass

- Body Menu Commands:
- Type 'add<j>' to add bodies to bj : i.e. 'add1' to add bodies to b1
- Type 'rem<j>' to remove bj from list
- Type 'name<j>' to edit bj.name
- Type 'par<j>' to edit bj.parent
- Type 'type<j>' to edit motion type, bj.type: {a...h}
- Type 'axis<j>' to edit bj.axis: {x, y or z}
- Type 'ang<j>' to edit initial inboard bj.ang for 1 dof rotational joints
- Type 'wrel<j>' to edit bj.angular\_rate
- Type 'mass<j>' to edit bj.mass
- Type 'svec<j>' to edit bj.svec
- Type 'dvec<j>' to edit bj.dvec
- Type 'inr<j>' to edit bj.inr
- Type 'edit<j>' to see all data on bj
- Type 'vmass' to go to variable mass bodies menu
- Type 'help' to get definition on data and commands
- Type 'x' to exit menu

# 3 Ways to Edit Body Data

- Direct edit: type the following commands for immediate edit
  - name<j>, type<j> axis<j>, ang<j>, mass<j>
  - units<j>
  - i.e. 'name1' to change b1 name
- Parameter menu edit: type any of the following parameters and go to named menu and then edit
  - {dvec, svec, inr, wrel, dpos, dvel, ... }
- Selected body edit: Type 'edit<j>' to edit all parameters of bj

# Inertia Menu

- Need define moi of each body bj about the bj.cm
- Type 'inr' from Body Menu to see a summary of moi data

```
nof vm bodies = 1
b1 dry inertia = 1000.000 1200.000 1200.000
                .000 .000 .000
var mass inertia= 800.000 12750.000 12750.000
                  .000 .000 .000
*b1 inr below is composite inertia about cm1 in b1 frame.

  idx name  -   ixx  - -   iyy  - -   izz  -
            ---- ixy --- ---- ixz --- ---- iyz ---
=>  1 b1    .180000E+04 .139500E+05 .139500E+05
      .000000E+00 .000000E+00 .000000E+00
      2 arrayS .100000E+01 .100000E+01 .100000E+01
      .000000E+00 .000000E+00 .000000E+00
      3 arrayN .100000E+01 .100000E+01 .100000E+01
      .000000E+00 .000000E+00 .000000E+00
```

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

# Dvec Menu

- Need define  $dvec(j)$ ,  $bj.hinge.position$  in  $bj.parent$  frame, for each  $j$
- Type 'dvec' from Body Menu and see a dvec summary

```
idx name      u -----dvec-----
=>  1  b1       FPS  .0000000E+00  .0000000E+00  .0000000E+00
    2  arrayS   FPS - .3000000E+01  .3000000E+01  .0000000E+00
    3  arrayN   FPS - .3000000E+01 - .3000000E+01  .0000000E+00
```

- Note: by design  $b1.dvec=0$
- Type 'dvec<j>' to edit  $bj.dvec$
- Type 'x' to exit menu

# Svec Menu

- Need define svec(j), bj.position.cm in bj.local frame, for each j
- Type 'svec' from Body Menu and see an svec summary

```
vm_size      = 1
b1_dry_svec  =      .000      .000      .000
*b1 svec below is composite cm1 position in b1 frame.

  idx name      u -----svec-----
=>  1 b1        FPS -.750000E+01 .000000E+00 .000000E+00
    2 arrayS    FPS  .000000E+00 .300000E+01 .000000E+00
    3 arrayN    FPS  .000000E+00 -.300000E+01 .000000E+00

[ sel  units  sVec(x/y/z)  azel  vec  rae  ]
[ up   down   help      x      ]
>
```

- Note: b1.cm here is collocated with b1.hinge point, but can be nonzero
- Type 'svec<j>' to edit bj.svec
- Type 'x' to exit menu



# Pos Summary

- See where bj.cm is in b1 frame for all j
- Type 'rpos' in Body Menu to see a summary of bj.cm

```
  idx name      -----  ---rpos---  -----  
=>  1  b1        -7.500      .000      .000  
    2  arrayS     .000       3.000     .000  
    3  arrayN     .000      -3.000     .000
```

- Note: b1.cm here is composite b1.vm cm position in b1 frame

# All b1 Data

- See all data on b1, type 'edit1' from Body Menu

```
idx  name      par  gflg  u  tp  ax      ang      mass
  1  b1         0    10  FPS  b  x      .000    .20000E+03

> inertia<inr>= .18000E+04  .13950E+05  .13950E+05
                .00000E+00  .00000E+00  .00000E+00
> dVec          =          .000          .000          .000
  hngVec        =          .000          .000          .000
> sVec          =         -7.500          .000          .000
  rVec          =         -7.500          .000          .000
> dcm0          =    1.000000    .000000    .000000
                .000000    1.000000    .000000
                .000000    .000000    1.000000

  Euler seq    =123
  Euler angs   =          .000          .000          .000

  b2i matrix   =    -.000002    .000000    -1.000000
                .422618    .906308    -.000001
                .906308    -.422618    -.000002

> wRel         =    1.0000    .0000    .0000
> dPos         =          .000          .000          .000
> dVel         =          .000          .000          .000
  force on b   =          .000          .000          .000
  torque on b  =          .000          .000          .000
```

# Bj Page Info

- 1<sup>st</sup> block: all attributes of bj, {idx, ... torque on b}
- 2<sup>nd</sup> block: commands to change attributes in block1 or to go to another Body Menu: {idx, axis,... x}
  - ldx<j>: goes to another bj page
  - Dcm0: edit relative dcm of bj
    - for j=1, this is the b1 attitude wrt LVLH frame
  - Svec: edit the bj cm position in bj coord, etc...
  - Help: see data and command definitions
  - X: exit this page

# VMASS Menu

- Need to embed Mothership mass property in b1
- Type 'vmass' from Body Menu and see the VMASS (variable mass) summary shown on the next page
- For Sep1sim, b1 has only on vmass body= Mothership
- Data on the VMASS page:
  - b1.mass0 = b1 mass without vmass items(i.e. satellite only)
  - b1.mass> = mass of composite b1
  - b1.inr> = inertia of composite b1
  - b1.svec> = cm position of composite b1 in b1 frame
  - b1.w = angular rate of composite b1
  - b1.hvec = system angular momentum in b1 frame
  - vm.\* = attributes of the selected vmass object (=>)
  - fuel = total fuel carried by vm object
  - ff> = fuel fraction expended
  - n = number of interpolation points used for vm.mass, vm.svec and vm.inr calculations

- $Vm.ff$  are the interpolation coefficients corresponding to the interpolation svec and inr columns, values are between 0 and 1
- $Vm.[Sx, Sy, Sz].col(1:n)$  are the interpolation svec of the selected vm
- $Vm.[lxx, lyy, lzz, lxy, lxz, lyz].col(1:n)$  are the interpolation inr of the selected vm
  
- For Sep1sim,  $vm(1).n=1$ , so
- $Vm(1).svec= vm.(Sx,Sy,Sz).col( 1)$
- $Vm(1).inr= vm.(lxx, lyy, lzz, lxy, lxz, lyz).(col.1)$
  
- Commands:
- Type 'add' to add a vm object
- Type 'namej' to set  $vm(j).name$
- Type 'remj' to remove  $vm(j)$
- Type 'nj' to set  $vm(j).n$
- Type 'ff>j' to set  $vm(j).ff>$
- Type 'ffj' to set the interpolation ff coefficients, between 0. and 1.0
- Type 'svecj' to set the jth interpolation  $vm.svec.col(j)$ ; j is between 1 and 5
- Type 'inrj' to set the jth interpolation  $vm.inr.col(j)$ ; j is between 1 and 5

```

~ Variable Mass Menu ~
vm.bodies      : 1
vm.total_mass  : 100.000
vm.actual_mass : 100.000
stages        : 1

b1.mass0=      100.000
b1.mass>=      200.000
b1.svec>=      -5.000          .000          .000
b1.inr>=       1300.000        6800.000        6800.000
              .000          .000          .000

b1.w           = 1.000          .000          .000
b1.hvec        = 23.038         .000          .000

vm  name      tp  loc  n  tot.mass  fuel  ff>  act.mass
=> 1 Mship    0   0   1  100.000  .000  .0000 100.000

vm.ff>         = .000
vm.svec>=      .000          .000          .000
vm.inr>=       300.000        600.000        600.000
              .000          .000          .000
vm.dvec        = -10.000        .000          .000
vm.pos         = -10.000        .000          .000

-----
vm.ff          .00000

vm.Sx          .000
vm.Sy          .000
vm.Sz          .000

vm.Ixx         300.000
vm.Iyy         600.000
vm.Izz         600.000
vm.Ixy         .000
vm.Ixz         .000
vm.Ixz         .000

```

# Stage Menu

- Need to define vm objects to eject or separate from b1
- Type 'stage' from VMASS Menu and see the stage events as follows for Sep1sim

```
~ Stages Menu ~
stages: 1
=> #      name      n:  elist
    1    scsep1     1:  1,
-----
post stage b1 property:
mass = .000000E+00
inr  = .000000E+00 .000000E+00 .000000E+00
      .000000E+00 .000000E+00 .000000E+00
svec = .000000E+00 .000000E+00 .000000E+00
[ sel  add  rem  name  n  list  mass  inr  svec  help  x  ]
>
```

- SEP1simc has 1 Stage event
- stg.name = 'scsep1'
- stg.n = number of vm objects to separate from b1
- stg.elist = list of vm objects to eject for stgj, j=1: stages

# Stage Menu Commands

- Type 'add' to add a stage event
- Type 'namej' to set stgj.name
- Type 'remj' to remove stgj
- Type 'nj' to set stgj.n
- Type 'listj' to set stgj.elist(1:stgj.n, j)
- Type 'help' to see data and command definitions
- Type 'x' to exit the menu
  
- The input control signal name for a staging event is 'xstg,j'
- Input processing of xstgj:

Input signal	size	Sim action
xstgj ==1	1	Removes all vm bodies in stgj.elist



# Body Actuation Signals

- B<sub>j</sub> 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 b<sub>j</sub> depends on b<sub>j</sub>.type. They are processed as follows:

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

# Wheel Menu

- See wheel summary, type 'whl' from Model Menu
- SEP1simc has 4 wheels, all mounted on b1 (see pa)

```
~ Wheel Menu ~
nof wheels      :      4
sysh_eci        :  -.551523E-04  .134245E+02  .287889E+02
syscm           :  -.742574E+01  .000000E+00  .000000E+00

w_inr           :  .100000E+00
w_spd(d/s)     :  .000000E+00
w_mom          :  .000000E+00

az_axis        :      2
axis(1) az(d)  :  45.000000
axis(1) el(d)  : -35.264390

units          :  FPS

idx name      pa t ----- ---axis--- ----- --winr- -w(rpm)-
=>  1 whl1     1 A  .5773503  -.5773503  .5773503  .1000  .0
    2 whl2     1 A  .5773503  .5773503  .5773503  .1000  .0
    3 whl3     1 A - .5773503  .5773503  .5773503  .1000  .0
    4 whl4     1 A - .5773503 - .5773503  .5773503  .1000  .0

[ sel name  units  par  type  inr  wspd  axis  azel  azAxis  long  ]
[ show order  idx   add  rem  copy  save  help  x      ]
>
```

# Wheel Menu Data & Commands

- Data:
  - `whlj.parent= 1`, for all `j`, means all wheels are on `b1`
  - `whlj.type=A` means `whlj.input` is scalar wheel torque
  - `whlj.axis= [x y z]` , `whlj` spin axis in parent frame
  - `whlj.speed=` initial `whlj` speed
  - `whlj.inr=whlj` spin axis inertia
- Commands:
  - Type `'add<j>'` to add wheels to `bj`
  - Type `'rem<j>'` to remove `whlj`
  - Type `'name<j>'` to edit `whlj.name`
  - Type `'type<j>'` to edit `whlj.type={A or C}`
  - Type `'axis<j>'` to edit `whlj.axis`
  - Type `'wspd<j>'` to edit `whlj.wspd`
  - Type `'help'` to get definitions of data and commands
  - Type `'x'` to exit menu

# Wheel Control Signals

- Dynamics Input signals for whlj: whltqj, whlaccj
- Run time input processing of wheel control signals:

whlj.type	Input signal	size	Sim action
A	whltqj	1	whlj.tq= whltqj
C	whlaccj	1	whlj.acc= whlaccj

# Force Menu

- See force summary, type 'force' from Model Menus
- SEP1simc has 8 jet forces
- fj.parent=1 for all j, means they all impinge b1
- fj.type=1 for all j means that the xsv01.dll input for them are 1/0 signals

```
idx name      p t  c  ---f mag---  ---f x---  ---f y---  ---f z---
=>  1 f1        1 1  1    1.000    1.000    .000    .000
    2 f2        1 1  1    1.000    1.000    .000    .000
    3 f3        1 1  1    1.000    1.000    .000    .000
    4 f4        1 1  1    1.000    1.000    .000    .000
    5 f5        1 1  1    1.000    .000    -1.000  .000
    6 f6        1 1  1    1.000    .000    -1.000  .000
    7 f7        1 1  1    .000    .000    .000    .000
    8 f8        1 1  1    .000    .000    .000    .000
```

- Note: f7,f8 are not given force values because they are not needed by acs\_jet.m of SEP1sim1

- Type 'pos' to see the position of defined forces in b1 coord as next

```

=>  idx name      p t c  ---f mag---  ---posx---  ---posy---  ---posz---
      1 f1        1 1 1    1.000    -3.000    -3.000     .000
      2 f2        1 1 1    1.000    -3.000     3.000     .000
      3 f3        1 1 1    1.000    -3.000     .000     3.000
      4 f4        1 1 1    1.000    -3.000     .000    -3.000
      5 f5        1 1 1    1.000     .000     3.000     3.000
      6 f6        1 1 1    1.000     .000     3.000    -3.000
      7 f7        1 1 1     .000     .000     .000     .000
      8 f8        1 1 1     .000     .000     .000     .000

```

- Type 'rx' to see the torque of defined forces in b1 coord

```

=>  idx name      ---tqmag---  ---tqx---  ---tqy---  ---tqz---
      1 f1        3.000     .000     .000     3.000
      2 f2        3.000     .000     .000    -3.000
      3 f3        3.000     .000     3.000     .000
      4 f4        3.000     .000    -3.000     .000
      5 f5        3.000     3.000     .000     .000
      6 f6        3.000    -3.000     .000     .000
      7 f7        .000     .000     .000     .000
      8 f8        .000     .000     .000     .000

```

# Force Menu Data & Commands

- Data:
  - `fj.parent= 1`, for all `j`, means all jets are on `b1` (bus)
  - `fj.type=1` means `fj.input` is an on/off signal
  - `fj.fvec= [x y z]` ,directional vector of `fj`
  - `fj.fmag=` magnitude of `fj` when activated
  - `fj.fpos=[x y z]`, impact position of `fj` in parent frame
  
- Commands:
  - Type '`add<j>`' to add external forces to `bj`
  - Type '`rem<j>`' to remove force(`j`) or `fj`
  - Type '`name<j>`' to edit `fj.name`
  - Type '`type<j>`' to edit `fj.type={1,2 or 3}`
  - Type '`fvec<j>`' to edit `fj.fvec`
  - Type '`fpos<j>`' to edit `fj.fpos`

- Type 'par<j>' to edit fj.parent
  - Type 'pos' to show all force impact positions in b1 frame
  - Type 'rx' to show torque of all forces about f\_ref in b1 frame
  - Type 'help' to get definitions of data and commands
  - Type 'x' to exit menu
- 
- Dynamics Input signal for fj is x fj
  - Run time input processing of x fj:

fj.type	x fj	Size	Sim action
1	1=on, 0=off	1	fj.force= pre-set value or zero
2	Scalar magnitude of fj.force	1	fj.force= x fj*fj.unitv
3	3 x 1 force vector in b1 frame	3	fj.force= x fj



# Gravity/Orbit Menu

- Need to specify the vehicle orbit
- Type 'grav' from Model Menu to enter Gravity Menu
- SEP1simc orbit :
  - 35 deg inclined circular orbit
  - True anomaly= 90 deg
  - orbit period is 100 minutes
  - gflag= 10, meaning spherical earth gravity model
  - epoch: 12.23.2009 (see sunpos menu) relates to LST

# Gravity Menu

```
> units      (U)=  FPS
> syspos    =      37130491.887      27.245      58.427
> sysvel    =      -.034      8232.778      17655.249

> refpos    =      37130491.887      27.245      58.427
> refvel    =      -.034      8232.778      17655.249

gravity:
> gx gy gz  = -10.2101349554    -.0000074919    -.0000160664
mu          = .140764418E+17

ephemeris:
> semimajor (U)=      37167659.547
> ecc        =      .001000
> incl      (deg)=      65.000000
> rasc      (deg)=      .000000
> argp      (deg)=      .000000
> t_anom    (deg)=      .000099
e_anom      (deg)=      .000099
m_anom      (deg)=      .000099
m_motion(d/s)= .03000000

> LST_ang (deg)= 268.291360
> LST(h:m:s) = 17:53: 9.9

> period (min)=      200.000; revs/day=      7.200
> period (sec)=      12000.000
range      (U)=      37130491.887
equ. radius =      20925646.325; J2= .108263E-02
prg.altitude =      16204845.562; apg.altitude=      16279180.881
we (d/s,r/s)=      .00417807      .00007292

> sysacc flag = 1
> gravity flag = 10
> atd option = LULH attitude unchanged
```

# Gravity Model Selections

- Gravity flag: gflag
  - 0 g is fixed as given by [gx gy gz] (flat earth)
  - 10 g at bj.cm is defined by syspos (spherical earth)
  - 11 g at bj.cm is defined by bj.pos.eci
  - 12 same as #11 plus gravity gradient torque on bj
  - 20 g at bj.cm with J2 is defined by syspos (oblate earth)
  - 21 g at bj.cm with J2 is defined by bj.pos.eci
  - 22 same as #21 plus gravity gradient torque on bj
  - 30 g at bj.cm with J2, J3 and J4 is defined by syspos (oblate earth)
  - 31 g at bj.cm with J2, J3 and J4 is defined by bj.pos.eci
  - 32 same as #31 plus gravity gradient torque on bj

# Gravity Menu Commands

- Type 'spos' to edit orbit position in eci coordinates
- Type 'svel' to edit total velocity in eci coordinates
- Type 'grav' to edit gravitational acceleration, [gx, gy, gz]
- Type 'semi' to edit semimajor axis
- Type 'ecc' to edit eccentricity, ... and so forth
- Type 'perm' to edit orbit period in minutes
- Type 'sflag' to run simulation in prescribed(spos,svel) mode or in force determined(spos,svel) mode
- Type 'gflag' to select the gravity model for the simulation
- Type 'help' to get definitions of data and commands
- Type 'x' to exit menu
- Buildx automatically updates all orbit parameters when one of them is altered, i.e. changing 'perm' results in a new (spos, svel, ephemeris) ...etc.

# Constraint Menu

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

	idx	type	ic	ln	ov	h1	h2	f1	f2	d1	d2	d3	p1	p2
	---	---	--	--	--	--	---+---	---+---	---	---	---	---+---	---	
=>	1	LOCK	0	1	0	2	0: 0	0: 0	0	0	0: 0	0	0	
	2	LOCK	0	1	0	3	0: 0	0: 0	0	0	0: 0	0	0	

- SEP1simc needs the following:
  - a. Lock constraints for b2 when ang2==0
  - b. Lock constraints for b3 when ang3==0

- Cn Data:
- $cnj.name$ = 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  $cnj$

- Cn Commnads:
- 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
  
- Dynamics Input signal for cnj= cnj
- Input processing of cnj:

Cnj	size	Sim action
1	1	Enables the constraint represented by cnj
0	1	Disable cnj

- See constraint selection menu, type 'add' from Constraint Menu

```
~ Constraint Menu ~  
  
Select a constraint type?  
1. body-to-body gear constraint [bb]  
2. wheel-to-wheel gear constraint [ww]  
3. body-to-wheel gear constraint [bw]  
4. u_dot_r constraint [r ]  
5. u_dot_ri constraint [ri]  
6. u-dot(pos1-pos2) constraint [rr]  
7. u-dot-u constraint [uu]  
8. hinge lock constraint [lk]  
9. body1 & 3 wheels constraint [w1]  
10. wheel-to-ground constraint [wg]  
    exit menu [x ]  
  
>> _
```

- Type 'bb' to select body-to-body gear constraint
- Type 'ww' to select wheel-to-wheel gear constraint
- ans so forth
- Type 'lk' to select a hinge lock constraint
- Type 'x' to exit this menu



# Lock Constraint

- Constraint(j) is controlled by the dynamics input signal cnj
- Constraint(j) is enabled when cnj=1
- If cnj.type=Lock & cnj=1, then cnj.body.joint is locked
- For the example here, cnj.body=3 and 8
  
- To define cnj.lock for b2 and b3:
  1. Type 'add' from Constraint Menu
  2. Select 'LK' for constraint
  3. complete 'LK' dialog by replying 2 to obtain a lock constraint on body 2
  4. repeat 1-3 for body 3

# Dynamics Input

- Need input data to xsv01.dll to actuate the vehicle dynamics during run time
- Type 'input' from Model Menus to open the input menu to select data
- Use 'newlist' to get a suggested list for the current model
- Use 'add' and 'rem' command to modify current input list (udata)
- SEP1simc input list is as follows:

```
Udata list:
```

```
1> whltq,1      | 2> whltq,2      | 3> whltq,3
4> whltq,4      | 5> xf,1         | 6> xf,2
7> xf,3         | 8> xf,4         | 9> xf,5
10> xf,6        |11> cn,1         |12> cn,2
13> xstg,1
```

- whltq, 1:4= four rwa torque
- xf,1:6 = six jet on/off signals
- xstg,1 = staging or separation signal 1

# 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

- Procedure to define input signals for CMG4sim:
  1. Type 'add' from Input Menu and see an Input Selection Menu
  2. Type 'sel17' for 'whltq' variable
  3. Reply the prompt with '1, 4' to select htqax1:4
  4. Type 'sel22' for 'xf' variable
  5. Reply the prompt with '1, 6' to select xf1:6
  6. Type 'x' to return to Input Menu
  7. Type 'x' to exit Input Menu

# Dynamics Output

- Need output motion signals from the dynamics engine to drive control system during run time
- Type 'output' from Model Menus to open the output menu to select signals
- Use 'newlist' to get a suggested list for the current model
- Use 'add' and 'rem' command to modify current output list (ydata)
- SEP1simc output list is as follows:

```
Ydata list:
```

```
1> WREL,1           ! 2> B2OSML,1           ! 3> angle,2  
4> angle,3
```

- wrel, 1 = bus angular rate in b1 frame
- b2osml, 1= small angle b1.attitude\_err wrt LVLH frame
- angle,2:3 = array angles

# Output Menu Commands

- Type 'add' to add new variables to the end of ydata list
- 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

- Procedure to define output signals for CMG4sim:
  1. Type 'add' from Input Menu and see an Output Selection Menu
  2. Type 'sel65' for 'wrel' variable
  3. Reply with '1, 1' to select wrel1
  4. Type 'sel3' for 'b2osml' variable
  5. Reply with '1, 1' to select b2osml1
  6. Type 'x' to return to Input Menu
  7. Type 'x' to exit Input Menu

# Plot Data

- Sample selected data from the dynamics engine to plotfile during run time
- Type 'plot' from Model Menus to open the Plot Menu to select plot data
- Use 'newlist' to get a suggested list for the current model
- Use 'add' and 'rem' command to modify current plot data list (odata)
- SEP1simc plot list is as follows:

## Odata list:

```
1> QUAT,1           | 2> WREL,1         | 3> whlspd,1
4> whlspd,2        | 5> whlspd,3      | 6> whlspd,4
7> whltq,1         | 8> whltq,2       | 9> whltq,3
10> whltq,4        |11> syshmom       |12> SYSPOS
13> SYSUEL         |14> SYSACC        |15> B20123,1
16> syshb1         |17> angle,2       |18> angle,3
19> wrelax,2       |20> wrelax,3      |21> htqax,2
22> htqax,3        |23> mass,1        |24> inr,1
25> svec,1
```

- quat, 1= b1.attitude\_quaternion in eci frame
- wrel, 1= bus angular rate in b1 frame ... etc.
- syshb1= system angular momentum in b1 frame
- mass1, inr1, svec1= composite b1.(mass,inertia,cm position)



# 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 '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

# Simplot

- Need to construct simplot1.m to view sim results in Matlab, then type 'simplot' from Model Menu (page 15) or Plot Menu (page 56)
- All plot data were selected in Plot Menu
- A. steps from simPlot Menu:
  1. Type 'add' to add figures and respond with '4' to create 4 figures
  2. Type 'title1' to set figure (1) title: i.e. reply with 'system'
  3. for 'title2' and 'title3' commands, respond with 'bus', 'wspd' and 'wtq' respectively
  4. Type 'vars1' to define variables to be plotted in fig 1. this opens the plot variables page
- B. steps from vars menu:
  1. Type 'addv11' and reply with 4 to add 4 variables starting with the variable(11), syshmom
  2. Type 'addv16' and reply with 1 to add 'syshb1' to the variable list
  3. Type 'addp' and respond with '1,5' to create five subplots for figure(1)
  4. Type 'format' and respond with '3,2' to plot 6 subplots in 3 rows and 2 columns format
  5. Type 'x' to go back to simPlot Menu
- Repeat steps A.3 and all B steps with proper indexing to define subplots of other figures for SEP1simc (i.e. 'bus', 'wspd' and 'wtq')
- 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.

- Procedure to define Plot Data:
  1. Similar to those given for Input and Output data selection
  2. quat, 1= b1.attitude\_quaternion in eci frame
  3. angle, 2= b(2).ang; first CMG angle ... etc.
  4. syshb1= system angular momentum in b1 frame

# Save Model Data

- Need to save current data to a model file
- Type 'save' from Model Menu (page 15) and complete the save dialog as follows

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

Commands:

1. save model data to sep1sim.txt
  2. save model data to another file
  3. Cancel save
- Select 1:3? 1

model data has been saved to sep1sim.txt

- On hitting return, current model data would have been saved to SEP1simc.txt
- Try option 2

# Exit Buildx

- 3 ways to exit Buildx:
  - go to Model Menus or Main Menu 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 60

# 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
- Can jet forces placement and alignment be specified like in real vehicles?
  - Yes, one must in every case match the jet controller with the torque characteristics of the jets
- Are all Project licenses strictly for object count {1, 4, 8}?
  - no, for example a Project license for a vehicle with 4 CMG's has an object count of {5, 4, 8} and a unique parent-child relation between bodies, wheels and forces
- How to setup SEP1simc as a 3 wheel system under a Project license?
  - go to xsv.wheel menu and set whl(4).type=3, whl(4).wspd=0
  - edit whlj.axis, whlj.inr, whlj.wspd for j=1:3 as necessary
- How to setup satsim with < 8 jets under a Project license?
  - as an example, let's disable jet(7:8):  
go to xsv.forcemenu and set frc(7:8).type=1 and not include these in xsv.input list (udata)

- 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 there '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

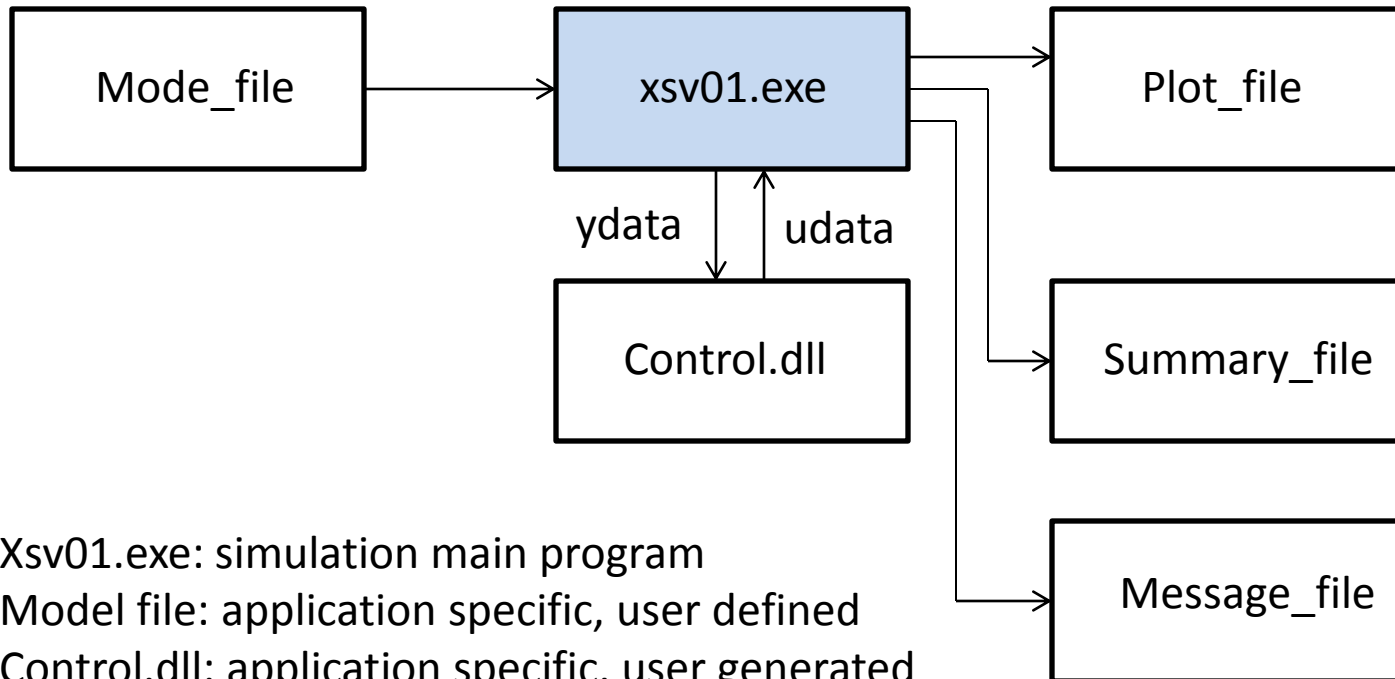
- Xsv01.exe
- How Control.dll Works
- User\_code.c
- Vehdata.h
- Utilc.lib subroutines
- Acsc.lib subroutines
- Control System
- Example1
- Example2
- Adjustable Simulation Parameters
- Exercises
- Simulation Notes
- Summary



# XSV01.exe

- Xsv01.exe is the main program. It comes with the SEP1simc package. Its functions are to:
  - Read data from model file to initialize the simulation database
  - Integrate numerically the motion equations required by the model file while passing motion signals to and receiving actuation signals from the control system represented by control.dll.
- Control.dll is the application specific control system that user provides to cause the vehicle motion respond in a desired manner.

# Xsv01.exe Dataflow



- Model\_file: defines all parameters of the vehicle for the simulation
  - Summary\_file: records all the model parameters and initial condition of an xsv01.exe run
  - Plot\_file: a file of sampled plot data recorded during a simulation
  - Message\_file: messages from xsv01.exe during a run
- > Part I has already described how to build the model file for sep1simc.
- > Part II begins next

# Part II Introduction

- The following charts show how to build a trial control.dll that has the framework of the control system needed for the application.
- Two parts need to be built: control subroutines and header file
- At a minimum, we would use Buildx.Codegen Menu to construct 5 key control subroutines and their header files.
- Any discrete/analog process in the control system counts as an additional control subroutine that Codegen must account for.
- Special control subroutine names cause Codegen to insert practical details into those routines
- Constants and utility routines inserted by Codegen in the trial control.dll need be replaced with those based on analysis and application specific algorithms later.
- A functional control.dll should be obtained after a few iterations of running it with xsv01.exe and correcting errors.
- From that point on, user can expand the size and details of the control subroutines to satisfy their simulation needs.

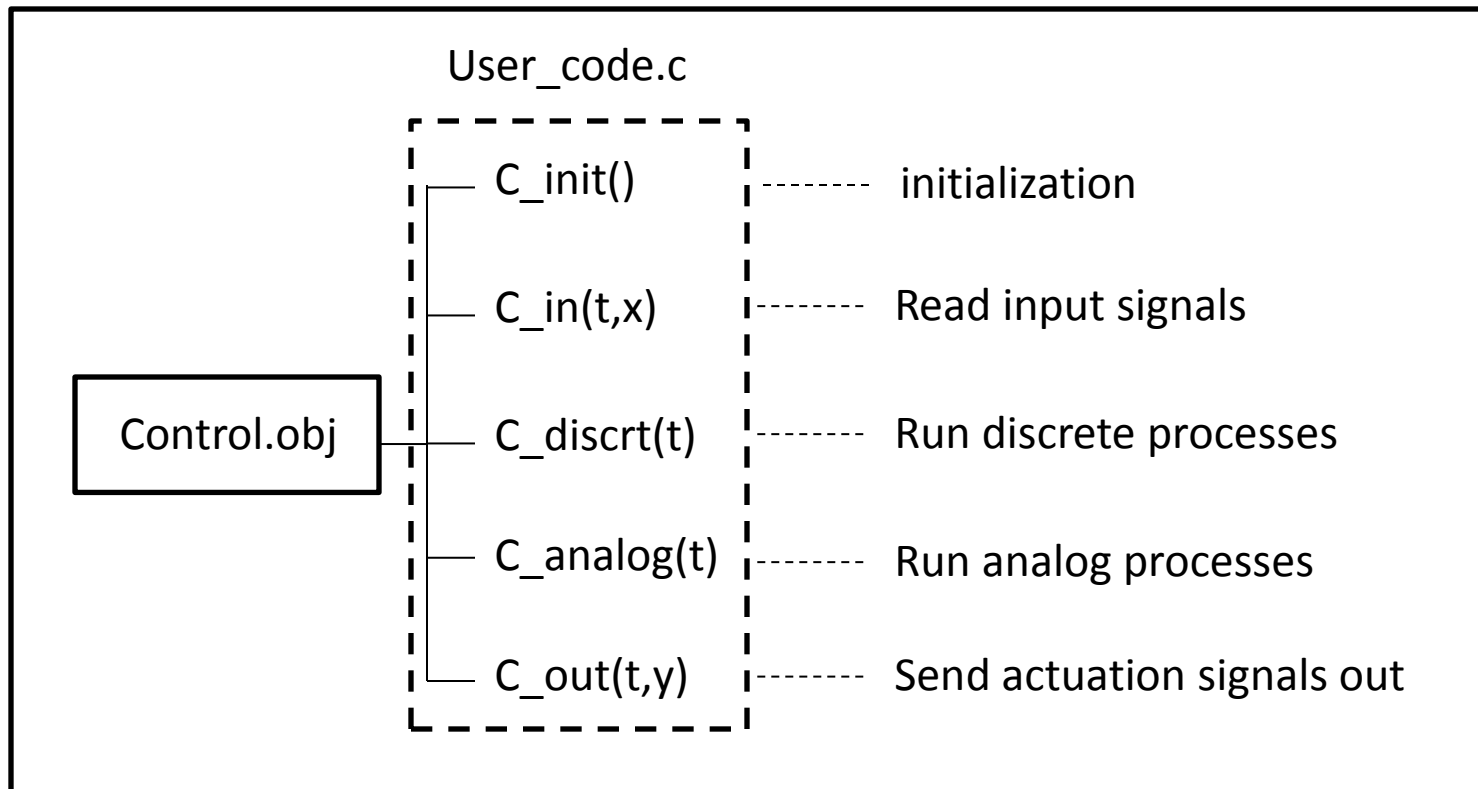
# How Control.dll Works

- It calls five subroutines: `c_init`, `c_in`, `c_discret`, `c_analog`, `c_out`
- The purpose and frequency of call to these subroutines are shown on the following pages.
- The header file used by these subroutines are defined in the `vehdata.h`
- Files in the package that help in the construction of `control.dll` are:

Files in the package	Description
<code>Control.obj</code>	Object code of <code>control.c</code>
<code>Utilc.lib</code>	Matrix-vector math utility routines
<code>Utilc.h</code>	<code>Utilc</code> subroutine call prototypes
<code>Acs_c.lib</code>	Attitude control system subroutines
<code>Acs_c.h</code>	<code>Acs_c</code> subroutine call prototypes

# Primary Control.dll Subroutines

Control.dll



# Primary C-Routines

subroutine	Call frequency	Function
C_init	Once at start of simulation	Initialize control related parameters
C_in*	At top of every derivative calculation	Read the motion signals from the plant dynamics
C_discret	At top of every derivative calculation	Execute all discrete events at discrete times
C_analog	At every derivative calculation	Execute all analog computations for signal processing and control
C_out*	At end of every derivative calculation	Send actuation signals back to the plant dynamics

# Building Control.dll

- A Short description of how to build control.dll:
  1. Go through a 'Codegen' procedure given next using Buildx.exe to generate a user\_code.c.tmp and the related header file vehdata.h.tmp. The former is a template of the five subroutines called by control.dll
  2. User would add details to user\_code.c.tmp and vehdata.h.tmp as required by the application.
  3. At the end of that process change user\_code.c.tmp to user\_code.c
  4. Likewise, change vehdata.h.tmp to vehdata.h
  5. Compile control.dll with the following CL command at DOS prompt
    - 'CL control.obj user\_code.c util.lib acsc.lib /LD'



# Codegen Procedure

- Purpose: generate user\_code.c.tmp, vehdata.h.tmp
- Given: vehicle model file from Part I

Preprocessing:

1. Go to Buildx >xsv > Model Menus
2. Use 'discrt' command to define discrete/analog processes : name and sample period (as necessary); period=0 => analog process
3. Use 'ode' command to define application specific 'ordinary differential equations': name, and state size (as necessary)
4. Use 'switch' command to define switches for events: switch name and simple switch function (as necessary)
5. Type 'cgen' to open the Codegen Menu

# sep1simc Discrete Processes

- For SEP1simc, we defined 2 processes: acs\_rwa(discret), acs\_jet(discret)
- This definition is done by using 'add', 'name' and 'period' commands

```
~ Discrete Process Menu ~  
Nof discrete process: 2  
Idx Name          Period      Start_t  
=>  1 acs_rwa      .1000E+00  .0000E+00  
    2 acs_jet      .1000E+01  .0000E+00  
[ sel add rem copy idx name order period start x ]  
>
```

- Need to save the new model data to sep1simc.txt
- Next, Go to Codegen Menu by typing 'cgen' from Model Menus
- Note: acs\_rwa and acs\_jet are two special discrete process names, because they cause Codegen to insert functional details in the generated code

# ACS\_C.lib Subroutines

- These are subroutines written for the examples, by that they need be replaced when configuration or mass property change from those given.

Subroutine	purpose	Examples
Get_acscmd	Compute ACS command signal to null attitude error	sep1simc, arraysimc, cmg4simc
Get_6jettimes	Compute 6 jet on/off times to null attitude error command	sep1simc, arraysimc
Get_hmgr6jettimes	Compute 6 jet on/off times to null system angular momentum	cmg4simc
Get_cmgtq	Compute CMG input torque to null attitude error	cmg4simc

# Special Names

- These special subroutine names when given to discrete processes cause functional codes to be generated for those .c.tmp routines

Process name	Purpose	Acs_c.lib routines invoked
acs_rwa	Generate RWA torque for ACS	Get_acscmd
acs_cmg	Generate CMG torque for ACS	Get_acscmd Get_cmgtq
acs_jet	Compute jet on/off times for ACS	Get_acscmd Get_6jettimes
hmgr_jet	Compute jet on/off times to null system angular momentum, syshb1	Get_hmtr6jettimes

# Codegen Menu

- Data on this page need be edited to assist in code generation

```
System Graph:
b1<B>+-w[1,2,3,4]

      Size      Hinges
sa      :      0      0  0
cmg     :      0      0  0  0  0  0
rwa     :      4
jets    :      6
gyros   :      3
cmdx_dt: .100000E+00
gyro_dt: .100000E+00
jet_dt : .100000E+01
rwa_dt : .100000E+00

generate:
cmdx code: no
gyro code: yes
rwa code: no

Commands:
[sa cmg rwa jets gyros dt cmdxC gyroC rwaC codeGen ]
[editc edith read save help x ]
```

- Type 'sa' to define number of arrays (max=2) and the driver hinges (bodies)
- Type 'cmg' to define number of CMG's and the rotor platform bodies
- Type 'rwa' to define number of reaction wheels
- Type 'jets' to define number of jets
- Type 'gyros' to define number of gyros
- Type 'dt' to define the sample period of 4 discrete processes
- Type 'codegen' to generate components of user\_code.c.tmp and vehdata.h.tmp
- Type 'gyroc' to toggle 'generate gyro code' flag
- Type 'rwac' to toggle 'generate rwa code' flag
- Type 'cmdxc' to toggle 'generate cmdx code' flag
- Type 'editc' to view and edit all \*.c.tmp codes
- Type 'edith' to view and edit all \*.h.tmp codes
- Type 'save' to save codegen data to 'codegendata.txt'
- Type 'read' to read codegen data from 'codegendata.txt'
- Type 'help' to get definition of menu data and commands
- Type 'x' to exit menu

- Let's concentrate on the two discrete processes that was defined for sv1sim1c.txt
- Next, type 'codegen' to generate the .c.tmp and .h.tmp files needed to build user\_code.c.tmp and vehdata.h.tmp

```
> codegen
created ctrl_input.h.tmp
created ctrl_output.h.tmp
created global_vars.h.tmp
created c_in.c.tmp
created c_init.c.tmp
created c_discret.h.tmp
created c_discret.c.tmp
created acs_rwa.h.tmp
created acs_jet.h.tmp
pgain & vgain for rwa control? 10,200
start and end times of asc_rwa? 2000 0
created acs_rwa.c.tmp
start and end times of asc_jet? 0 2000
created acs_jet.c.tmp
created c_gyro.c.tmp
created c_analog.c.tmp
created c_out.c.tmp
```

- You will be prompted for some gain and timing information in this dialog. Supply as appropriate.
- End time less than Start time means execute the code for 't > Start time'
- Start time= End time= 0 means ignore time window code request

- Next, type 'editc' to view user\_code.c.tmp components and to assemble user\_code.c.tmp

```
                Edit Cpp Files Menu

1. c_in.c.tmp
2. c_init.c.tmp
3. c_discret.c.tmp
4. acs_rwa.c.tmp
5. acs_jet.c.tmp
6. c_analog.c.tmp
7. c_out.c.tmp

Commands:
[ edit  add  name  rem  reset  assemble  help  x ]

>> _
```

1. One can view and edit each one of the \*.c.tmp code using the 'editj' command where j is the index preceding the displayed code.
  2. Type 'assemble' to assemble all displayed \*.c.tmp subroutines into user\_code.c.tmp. The latter becomes item 8 in the above list.
  3. Type 'edit8' in this case to view user\_code.c.tmp
- Just edit user\_code.c.tmp from here (i.e. 'edit8') until it is acceptably complete
  - Type 'Accept' to copy user\_code.c.tmp to user\_code.c, and exit (see Appendix A)



- Type 'edith' to view the generated vehdata.h.tmp components and to assemble vehdata.h.tmp

```
                                Edit .h Files Menu

1. ctrl_input.h.tmp
2. ctrl_output.h.tmp
3. global_vars.h.tmp
4. c_discret.h.tmp
5. acs_rwa.h.tmp
6. acs_jet.h.tmp

Commands:
[ Edit  Add  Name  Rem  Reset  Assemble  Help  x]

>>
```

1. One can view and edit each one of the \*.h.tmp code using the 'editj' command where j is the index preceding the displayed code.
  2. Type 'assemble' to assemble all displayed \*.h.tmp subroutines into vehdata.h.tmp. The latter becomes item 7 in the above list.
  3. Type 'edit7' in this case to view vehdata.h.tmp
- Just edit vehdata.h.tmp from here on (i.e. 'edit7') until it is acceptably complete
  - Type 'Accept' to copy vehdata.h.tmp to vehdata.h, and exit (see Appendix B)

# Compile Control.dll

- After obtaining user\_code.c and vehdata.h files, exit buildx.exe
- Type 'mkcontrolc.bat' to compile control.dll
- Fix any compilation error that appear in user\_code.c or in vehdata.h
- The xsv01.exe simulation is now ready to run

# Example

- This example shows the LVLH control with jet for  $t$  between 1000:3000 and with rwa for  $t \geq 3000$ .
- The satellite is separated from the mothership at  $t=500$
- It is pushed off for 1 sec at  $t=500$
- The satellite rate is first nulled with jets from  $t=1000$  to 2000
- Array is deployed at  $t=500$
- Jet ACS then captures and maintains LVLH attitude until  $t=3000$
- RWA ACS takes over the LVLH control for  $t > 3000$
  
- The attitude command signal is generated from true b1 angular rate ( $w1$ ) and xsv01.exe computed LVLH attitude error ( $b2osml1$ )
  
- Buildx procedure:
  - start Buildx and go to body menu
  - set 'wrel1' to [.1, .2, .3]degrees
  - from gravity menu, set [ecc,incl,period]=[0.,35 deg, 100 min]
  - save model data to sep1simc.txt from Model Menus

# Run xsv01.exe

- Click c:\sep1simc\xsv01.bat and see a running display as follows.

```
C:\sv1simc>..\xsv01
xsv01> t= .0000000E+00
xsv01> tuna> tunaIc+
          tPrint,tPlot,dtPrnt,dtPlot=
          .50000E+02 .20000E+01 .50000E+02 .20000E+01
simMethod= RK2
tuna> tuna_rk2(*),t= .0000000E+00
h= .1000000E-01; tf= .3000000E+04 <RK2>
model file= sv1simc.txt
plot file = z.1
summary   = sim1_summary.txt
xsv01> t= .5000000E+02
xsv01> t= .1000000E+03
xsv01> t= .1500000E+03
xsv01> t= .2000000E+03
xsv01> t= .2500000E+03
```

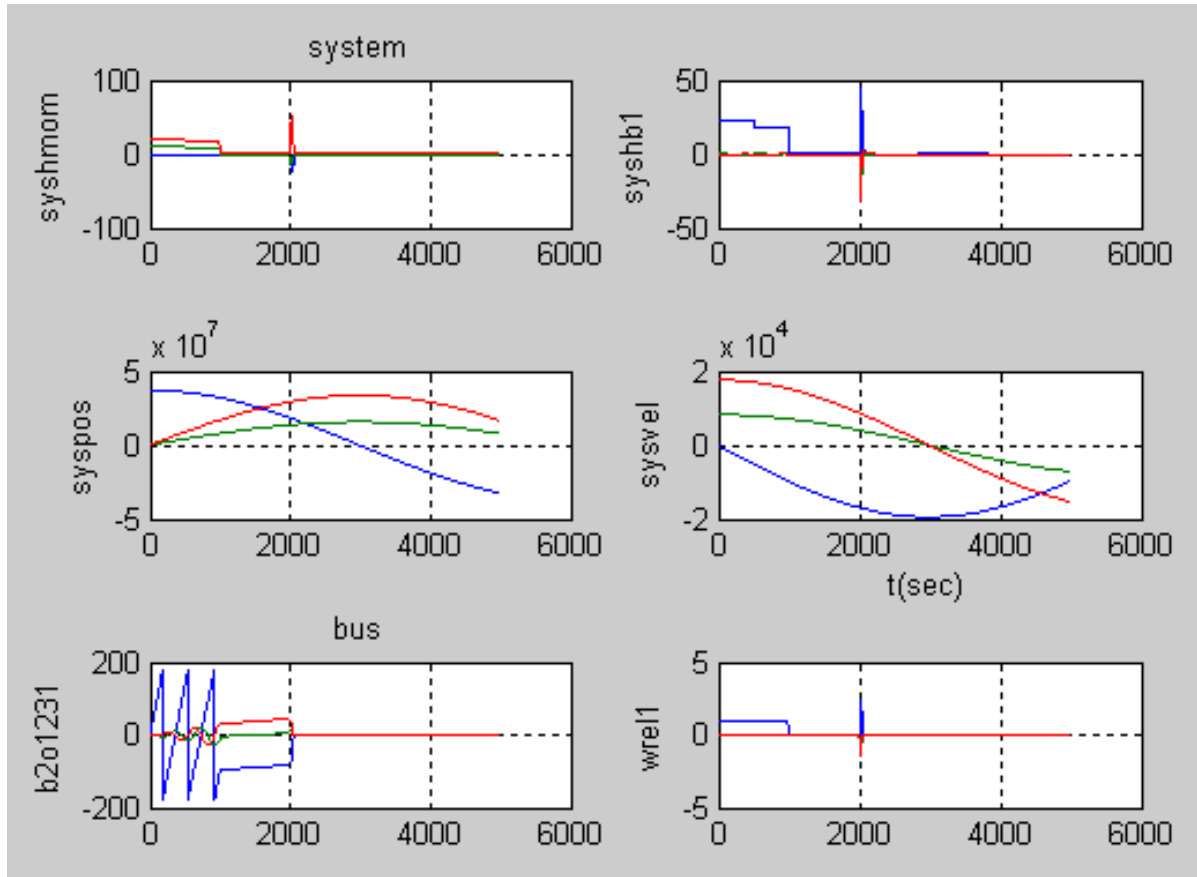
...

```
xsv01> t= .2900000E+04
xsv01> t= .2950000E+04
xsv01> t= .3000000E+04
C:\sv1simc>
```

# View Sim Results

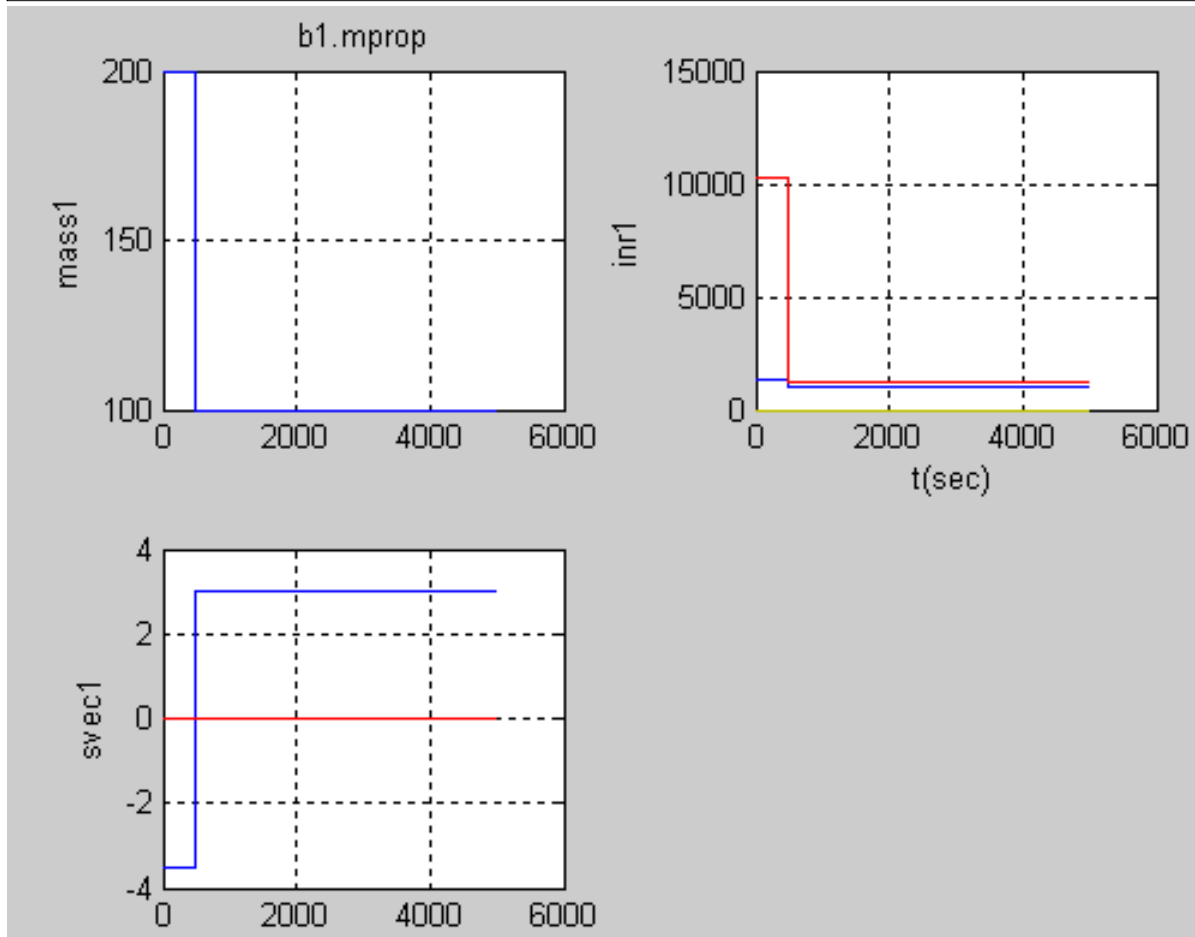
- Type 'load z.1' from Matlab window to read in sim result
- Type 'simplot1(z)' to view result
  - simplot1.m is a script that was constructed using Buildx.exe (see page 46)

# Fig.1 System Motion



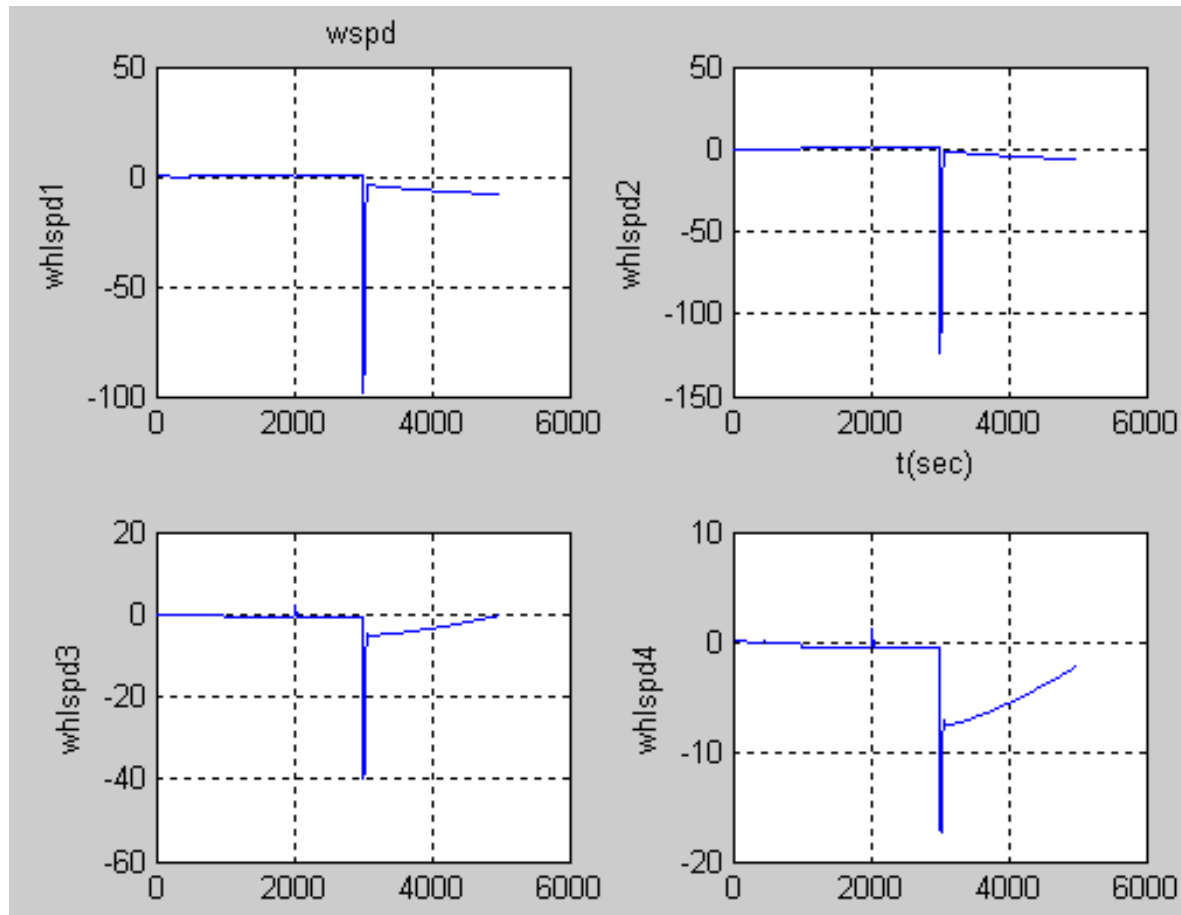
- orbit:  
35 deg incl  
ecc=.001  
6000 sec orbit period  
gravity: gflag=10
- sysmmom:  
momentum changed at  $t=500$  when sep occurred  
again at  $t=1000$  when ACS jet nulled veh rate  
transient at  $t=2000$  is when ACS jet capture LVLH attitude
- syshb1:  
sysmmom in b1 coord

# Fig.2 B1 Mass Property Change



- b1 mass property changed at t=500

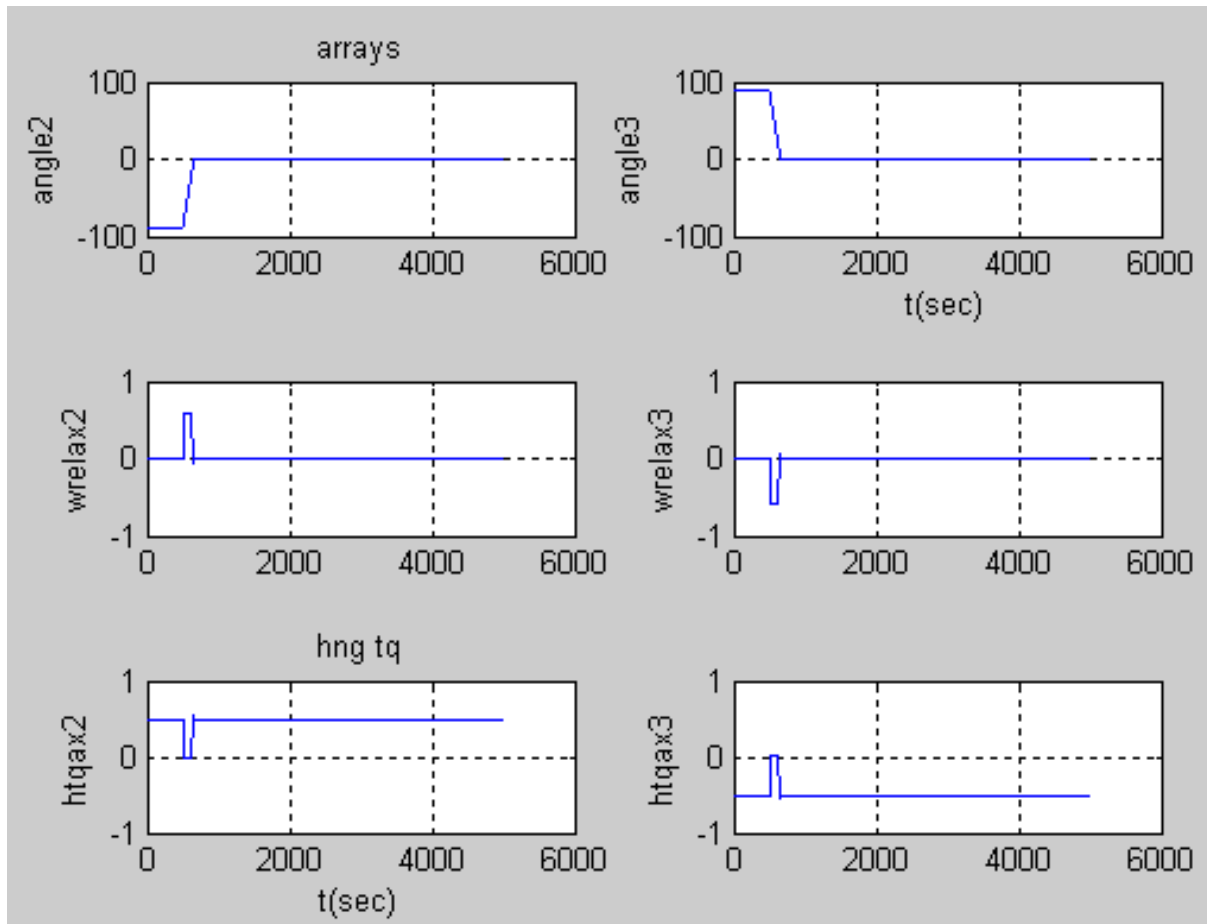
# Fig.3 Wheel Speed



- wheel speeds jumped when ACS transitioned from jet to RWA at  $t=3000$



# Fig.4 Arrays Deployment



- Array deployment started at  $t=500$
- Array 1 ang= angle2
- Array 2 ang= angle3
- Htqax2:3= hinge torque

# Comment

- This example showed a sequence of events that started with the satellite separation from its mothership. Afterwards, the ACS\_jet was invoked to null the vehicle rate and capture LVLH attitude. In parallel, two arrays are deployed. Past,  $t=3000$ , the ACS\_rwa is enabled to control the LVLH attitude. All these events occurred as programmed.
- All events are controlled by the c\_switch.c subroutine (Appendix A)

# Adjustable Sim Parameters

- Dt: simulation integration step size
  - plotDt: plot data sample period
  - printDt: time display sample period
  - T: simulation period
  - Method: Integration method, RK2 or RK4
- 
- Edit procedure:
    1. Go to Main menu
    2. Use 'stepsize', 'dtplot', 'printdt', 'endtime', 'method' commands to change the
    3. sim parameters
    4. Type 'save!' to save changes to siminputfile
    5. Reply with 'sim1files.txt'
    6. exit buildx

# Exercises

actions	parameters	reference
change mass property	mass, inr, svec, dvec	pages:15-20
change initial condition	ang,wrel,wrelax, dcm0	pages:15-17
change orbit	ephemeris, orbit period	pages:33-36
add /remove bodies*		pages:15-17
add /remove wheels*		pages:25-27
add/remove forces*		pages:29-32
modify input (udata)		pages:37-39
modify output (ydata)		pages:40-42
modify plot (odata)		pages:43-45
modify simplot1.m		page:46

\* Not for project licenses

actions	changes	control system config
design your own rwa controller	<ul style="list-style-type: none"> <li>•may need new ydata</li> <li>•may need less than 4 wheels</li> </ul>	<ul style="list-style-type: none"> <li>•likely identical to sep1simc</li> <li>•adjust i/o mux/dmux</li> </ul>
design your own jet controller	<ul style="list-style-type: none"> <li>•may need new ydata</li> <li>•may need nof jets other than 8</li> </ul>	<ul style="list-style-type: none"> <li>•likely identical to sep1simc</li> <li>•adjust i/o mux/dmux</li> </ul>

# Simulation Notes

Subject	sep1simc	comments
gforces	comment applies to sep1simc	gravity forces are auto-computed by sim engine for all bj in system
jet forces	force locations and vectors were selected for sep1simc to cause each jet torque to align with a particular b1.xyz axes to get a simple jet controller	generally jets are not placed ideally as in sep1simc because of other design considerations. as such, associated jet controller can be complex
wheels	sep1simc chose a 4 corner pyramid configuration	pyramid base to height ratio can vary and the center axis of pyramid need not be along any particular b1.xyz axis
geometry	sep1simc is a gyrostat and b1.cm is same as system.cm regardless of how b1.svec is defined	position and orientation of body parts are defined by their dvec, svec ,dcm0 and joint coordinates

Subject	sep1simc	comments
b1.dcm0	comment applies here also	Dcm0 of b1 is the initial LVLH attitude of the vehicle
mass, inertia	comment does not apply to sep1simc since it has only one regular body	mass and inertia can be set to zero for non-terminal bodies to represent ideal massless joints
sunb1,nadirb1	sun vector and nadir vector in b1 frame were not used in sep1simc control system	these two ydata are available for for sun sensor and earth sensor modeling
b2osml	this small angle LVLH attitude error of b1 computed by the sim engine is the rpy signal in the examples meaning (roll,pitch, yaw) angles.	this ydata signal is available as a functional convenience to ACS analysis. A more realistic rpy signal needs be built from sunb1 and nadirb1
syshb1	this is the system angular momentum in b1 frame, not used in sep1simc	this ydata signal is available as a functional convenience to momentum mgmt design

# Summary

- Activities simulated in the example are both serial and parallel. The successful execution of these events showed that xsv01.exe with a properly constructed control.dll can simulate the dynamics of a vehicle with jet and RWA ACS in maintaining LVLH attitude, payload separation and array deployment.
- The value of sep1simc is that its mass property can be changed to fit the vehicle of interest . It requires that C/C++ compiler be installed on the PC to help compile control.dll. With the Buildx editor, one can define a variety of Dynamics Input/Output signals to design and test his application specific control system. More importantly, it can be very effective in the design of any vehicle ejecting objects, its operations and control system.



# Appendix A

## User\_code.c

- `// This user_code.c is generated by Buildx.exe`
- `//`
- `// These subroutines manage the data in:`
- `// vehdata.h variables & constants used by`
- `// control.dll`
- `// utilc.h have the prototypes of matrix-vector`
- `// subroutines used in the generated code`
- `#include <stdio.h>`
- `#include <math.h>`
- `#include "vehdata.h"`
- `#include "utilc.h"`
- `// Control Input Mapping: x->local data`
- `void c_in(double *x,double *T){`
- 
- `equal(w1,&x[0],3 );`
- `equal(b2osml1,&x[3],3 );`
- `angle2= x[6] ;`
- `angle3= x[7] ;`
- `t= T[0] ;`
- `}`
-

- // initialization procedure.....
- void c\_init(){
- worb[0]= 0.;
- worb[1]= -.523599E-03; //(rps),temporary
- worb[2]= 0.;
- }
- 
- // c\_discret procedure.....
- void c\_discret(double \*T){
- int jet; // jet index
- 
- discrete(acs\_rwa, T, &acs\_rwa\_t);
- discrete(acs\_jet, T, &acs\_jet\_t);
- discrete( c\_gyro, T, &gyr\_t );
- for (jet=0; jet < 6; jet++){
- xf[jet]=0;
- if( T[0] > jet\_on[jet] & T[0] <= jet\_off[jet] )
- xf[jet]=1;
- }
- }

- void acs\_rwa( double \*ts ){
- //This procedure is called at time= \*ts
- double G[12]={ .433013E+00, .433013E+00,-.433013E+00,-.433013E+00,
- -.433013E+00, .433013E+00, .433013E+00,-.433013E+00,
- .433013E+00, .433013E+00, .433013E+00, .433013E+00};
- double cmd[3];
- double gain[2]={ .100000E+02, .200000E+03}; //temporary
- 
- double sc=-1 ;
- 
- if(\*ts > 3000 ){
- subv( worb, w1, tempv );
- get\_acscmd( gain, b2osml1, tempv, cmd); //see acsc.h
- mtxmv( G, cmd, wtq, 4, 3); //wtq=G\*cmd
- 
- mtxsv( &sc, wtq, wtq, 4 ); //wtq=-wtq
- }
- 
- \*ts= \*ts + acs\_rwa\_dt ;
- }

- void acs\_jet( double \*ts ){
- //This procedure is called at time= \*ts
- double cmd[3];
- double gain[2]={10,200}; //temporary
- double hjet= .300000E+01; //jet impulse/sec, temporary
- double sc=-200;
  
- if(\*ts >= 1000 & \*ts < 3000 ){
- if(\*ts < 2000 )
- multsv( &sc, w1, cmd );
- else {
- subv( worb, w1, tempv );
- get\_acscmd( gain, b2osml1, tempv, cmd); } //see acsc.h
  
- get\_6jettimes(ts, cmd, hjet, jet\_on, jet\_off ); /\*TEMP CODE\*
- }
- 
- \*ts= \*ts + acs\_jet\_dt ;
- }
-

- // switches procedure.....
- void c\_switches(double \*t){
- if( \*t < 500 ){
- xstg1= 0;
- on1 = 1;
- on2 = 1;}
- else if ( \*t-500 < 0.05 ){
- xstg1= 1;
- on1 = 0;
- on2 = 0;}
- else {
- fswc1= angle2 ;
- fswc2= angle3 ;
- if( on1==0 ) on1=cswitch(1,&fswc1,&sav1,t); //latch switch
- if( on2==0 ) on2=cswitch(2,&fswc2,&sav2,t); //latch switch
- }
- }
-

- `// Control analog procedure...`
- `void c_analog(double *T){`
- 
- `}`
- 
- 
- `// Control Output Mapping: u->dx/dt`
- `void c_out(double *T,double *u){`
- 
- `equal( &u[0],wtq,4 );`
- `equal( &u[4],xf,6 );`
- `u[10]= on1 ; // cn1=on1, array hinge lock`
- `u[11]= on2 ; // cn2=on2, array hinge lock`
- `u[12]= xstg1 ; // stage signal`
- `}`
-

# Appendix B

## Vehdata.h

```
/* control data: Udata */
double whltq1; /* wheel( 1) torque*/
double whltq2; /* wheel( 2) torque*/
double whltq3; /* wheel( 3) torque*/
double whltq4; /* wheel( 4) torque*/
double xf1; /* external force on body( 1)*/
double xf2; /* external force on body( 2)*/
double xf3; /* external force on body( 3)*/
double xf4; /* external force on body( 4)*/
double xf5; /* external force on body( 5)*/
double xf6; /* external force on body( 6)*/
double cn1; /* constraint( 1) enable flag*/
double cn2; /* constraint( 2) enable flag*/
double xstg1; /* constraint( 2) enable flag*/
double jet_on[100] ; /* jet start time */
double jet_off[100] ; /* jet end time */
double xf[20] ; /* jet [1/0] array */
double worb[3] ; /* orb rate in b1 frame */
double wtq[5] ; /* wheel torque(max 5) */
double cmgtq[5] ; /* cmg.input.tq(max 5) */
double cmg_ang[5] ; /* cmg.input.ang(max 5) */
double cmg_rate[5] ; /* cmg.input.rate(max 5)*/
int current_jet ; /* jet being processed */
```

- `/* control data: Ydata */`
- `double w1[3]; /* Body( 1) ANG RATE IN Body(1) FRAME*/`
- `double b2osml1[3]; /* Ref body to orbit attitude SMALL(XYZ) angles*/`
- `double angle2; /* Body( 2) HINGE ANGLE*/`
- `double angle3; /* Body( 3) HINGE ANGLE*/`
- 
- `/* Global variables & constants */`
- `double tempv[3],tempw[3];`
- `double pi= 3.14159265358979 ;`
- `double twopi= 6.28318530717959 ;`
- `double d2r= 0.01745329251994 ;`
- `double r2d= 57.29577951308232 ;`
- `double t ;`
- 
- `void acs_rwa( double * ) ;`
- `double acs_rwa_data[10];`
- `double acs_rwa_t= .000000E+00;`
- `double acs_rwa_dt= .100000E+01;`
- 
- `void acs_jet( double * ) ;`
- `double acs_jet_data[10];`
- `double acs_jet_t= .000000E+00;`
- `double acs_jet_dt= .100000E+01;`



- // switches procedure.....
- void c\_switches(double \*);
- double on1,sav1,fswc1,ena1=1;
- double on2,sav2,fswc2,ena2=1;
- double on3,sav3,fswc3,ena3=1;

# Appendix C

## Utilc.h

- `void addmtv( double *, double *, double *, double * );`
- `void addmv( double *, double *, double *, double * );`
- `void addsv ( double *, double *, double *, double * );`
- `void addsx ( double *, double *, double *, double *, int );`
- `void addv( double *, double *, double * );`
- `void addx( double *, double *, double *, int );`
- `void cross( double *, double *, double * );`
- `double cswitch ( int, double *, double *, double * );`
- `void dc2q( double *, double * );`
- `void dcsmall( double *, double * );`
- `void discrete( void (*)(double *), double *, double * );`
- `double dot( double *, double * );`
- `double dotx( double *, double *, int n );`
- `void equal( double *, double *, int );`

- void mtxsv( double \*, double \*, double \*, int );
- void mtxmtv( double \*, double \*, double \*, int, int );
- void mtxmv( double \*, double \*, double \*, int, int );
- void multmm( double \*, double \*, double \* );
- void multmtv( double \*, double \*, double \* );
- void multmv( double \*, double \*, double \* );
- void multsv( double \*, double \*, double \* );
- void null( double \*, int );
- void qdot( double \*, double \*, double \* );
- void qinv( double \*, double \* );
- void qiqmult( double \*, double \*, double \* );
- void qmult( double \*, double \*, double \* );
- void q2dc( double \*, double \* );
- void submtv( double \*, double \*, double \*, double \* );
- void submv( double \*, double \*, double \*, double \* );
- void subv( double \*, double \*, double \* );
- double unitvec( double \*, double \* );
- void xyzrot ( int \*, double \*, double \*, double \* );