SIMULATION EXAMPLES

A TEMPLATE for defining TACS Simulation Models

n1, n2 …. node numbers identifying state variables Y()
value ….. integers assigned to model parameters defined as variables
k1, a, etc integers assigned TO model parameters defined as constants
<.....> name of functions and models

pub (<model>) ; \ Define this word to establish a location in the dictionary where a model definition begins.
\ Execute **FORGET (<model>)** to remove from the dictionary.

DECIMAL \ Execute to be sure the default decimal mode has not changed.

const1 == k1 const2 == a etc. \ Set model parameters, defined as constants, that are included in model definition.

pub <model> …..... words …..... ; \ Define the model.

pub <function> ….....words..... ; \ Define functions for precalc or postcalc use. *Optional*

n NODES \ Set the number (n) of nodes (ie, state variables) in the model.

value1 node1 value2 node2 …... ASSIGN-IC \ Assign initial conditions to integrators.

value >DT value TO comint \ Set appropriate system parameter values.

value1 1 >K() value2 2 >K() …. \ Set model predefined scalar values included in model.

n1 n2 …. OUT-NODES \ Assign output nodes

SET-TO integr <name> \ Set the integrator.

SET-TO model <model> \ Set the model that will be simulated.

SET-TO postcalc <name> \ Set the specific post and/or pre calculation required by the model. *Optional*
SET-TO precalc <name>

SET-TO sinput <name> \ Required only if a special input function is part of the model.

CHECK \ Execute to establish that the initial conditions at each node in the model are correct.

SET-TO output DISPLAY \ Use the **DISPLAY** output mode for the first simulation run, to ensure that the
n SRUN \ data makes sense. Note that results will be listed in decimal mode.

SET-TO output <name> \ Set desired output mode.
gphx (x = 1, 2, 3 or 4) \ Set LCD graphics screen. See output modes in user manual.
n SRUN ….

Example 1 First Order Differential Equation

This first example is a simple first order differential equation with one parameter: dz/dt = -k*z. This example demonstrates how the postcalc command is used to automatically change a system parameter between runs

pub (FORDER) ;

DECIMAL

pub FORDER 4 1 K() 5 POT 5 4 integr ;

pub CPAR 1 K() 2 / 1 >K() ;

2 NODES

Y0 5 ASSIGN-IC

100 >DT
10 TO comint

-5000 1 >K()
4 5 OUT-NODES

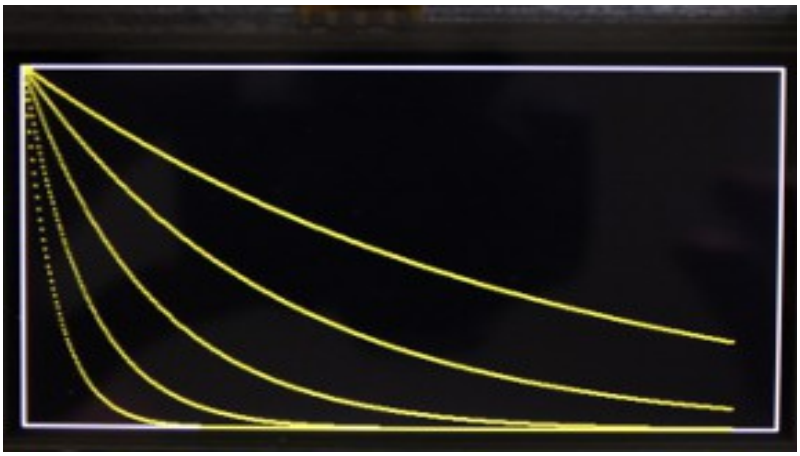
SET-TO integr EULER
SET-TO model FORDER

SET-TO postcalc CPAR

CHECK
The results should be: **4: -500000 5: 1000000** for this model.

Execute the following commands to plot simulation results at different parameter values.
SET-TO output PLOTY \ "z" will be plotted vs time.

gph1 \ Set graphics display mode.
We will run the simulation 5 times in succession, each time dividing the parameter1 K() in half. 450 points will be plotted in each run.
5 450 XRUN



Example 2 Van der Pol Equation

This is a nonlinear 2nd degree differential equation of the form:

$$dz^2/dt^2 + k(1-z^2)dz/dt + z = 0$$

pub (VDPOL) ;

DECIMAL

pub VDPOL 4 11 integr 5 4 integr 6 5 SQR 7 6 -REF SUM 8 1 K() 7 POT
9 4 8 MULT 10 9 5 SUM 11 10 INV ;

8 NODES

200000 5 ASSIGN-IC

10 >DT

40 TO comint

10000 1 >K()

11 4 5 OUT-NODES

SET-TO integr EULER

SET-TO model VDPOL

CHECK
4: 0
5: 200000
6: 40000
7: -960000
8: -960000
9: 0
10: 200000
11: -200000

SET-TO output PLOTY2 \ plots dz/dt & z

gph2

480 SRUN

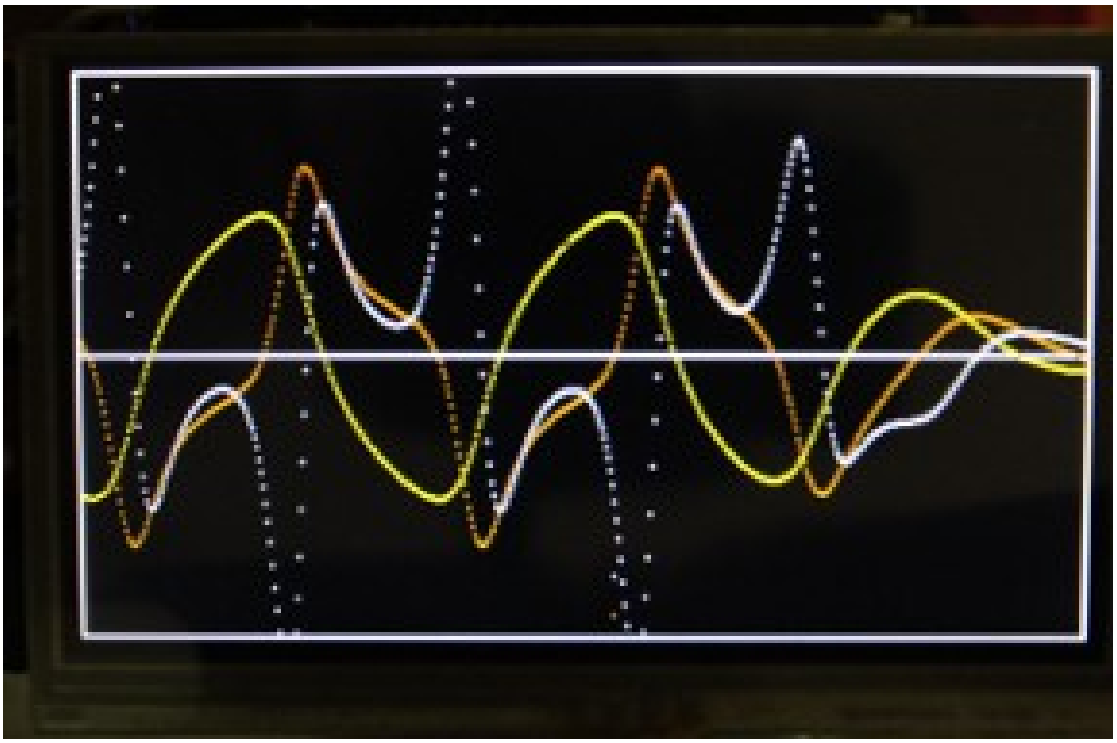
11 >YNODE

SET-TO output PLOTY \ adds d^2z/dt^2 plot

white >COLOR

480 SRUN

z – yellow dz/dt - orange d^2z/dt^2 - white



Example 3 Multiple Dosing (A problem in Pharmacokinetics)

This example demonstrates an approach to determining optimum drug dosing, using a simulation model of drug absorption-elimination. The model is represented as A → B → E, where A is the amount of drug at the absorption site, B is the amount in the body, and E is the amount eliminated. This one-body compartmental model assumes first-order kinetics.

dA/dt = ka*A dB/dt = ka*A – ke*B dE/dt = ke*B where D (the dose) = A+B+E

ka = absorption rate constant ke = elimination rate constant

pub (1BODY) ;

DECIMAL

Define a new block, DOSE (n1 n2 –), to enable multiple dosing.
Y(n1) = Y(n2) at integer values of 0 Y()/period else Y(n1) = 0.

pub DOSE 0 Y() 0= IF DUP 0 SWAP >Y() THEN Y() OVER Y() + SWAP >Y() ;

pub 1BODY 4 IMPULSE 5 4 DOSE 5 6 integr 6 ka 5 POT 7 6 INV 8 7 10 SUM
9 8 integr .10 ke 9 POT 11 10 integr 12 11 INV ;

9 NODES

Y0 5 ASSIGN-IC \ Set initial dose to 100 mg

100 >DT
15 TO comint

-1400 == ka \ ka = 1.4 hrs^{-1}
-2300 == ke \ ke = 0.23 hrs^{-1} (3 hr Biological Half-Life)

60000 TO period \ impulse frequency (6 hrs)
Y0 TO ye \ impulse amplitude = 100 mg

5 9 12 OUT-NODES

SET-TO integr EULER
SET-TO model 1BODY

CHECK
4: 0
5: 1000000
6: -140000
7: 140000
8: 140000
9: 0
10: 0
11: 0
12: 0

gph1

SET-TO output PLOTY

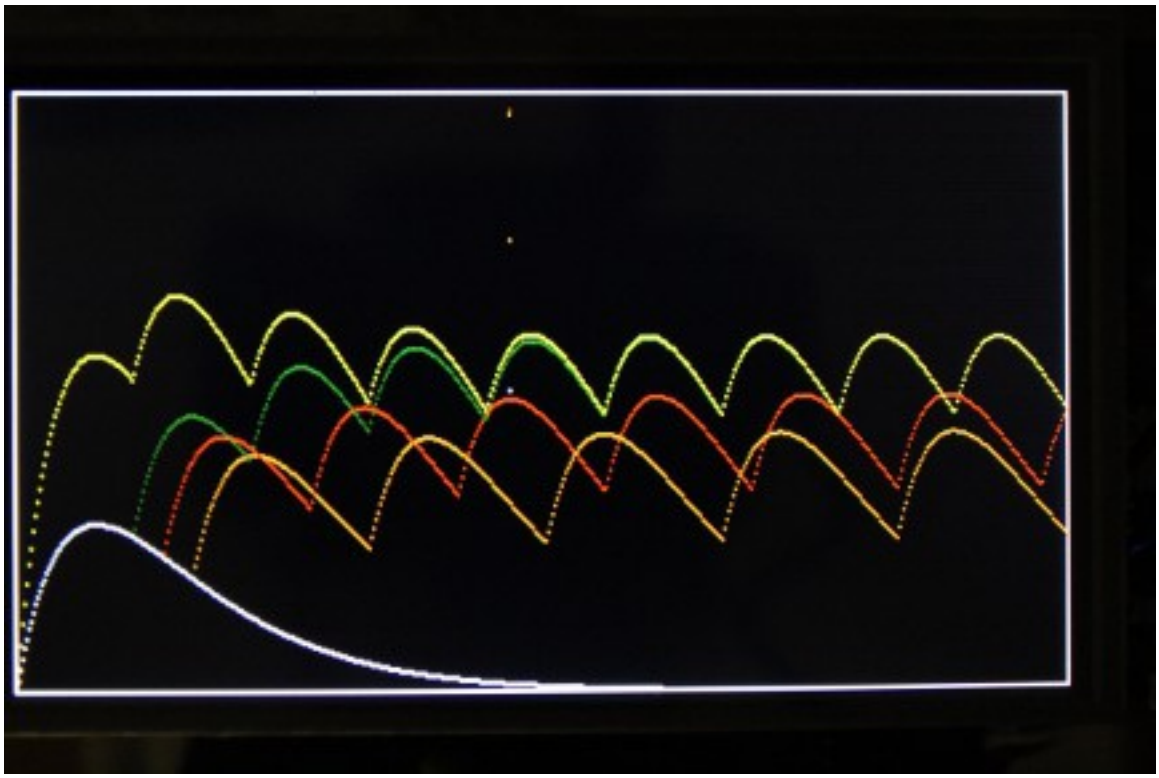
What is the effect of changing the dosing period?
white >COLOR
120000 TO period
0 TO xo
480 SRUN

Y0 TO xo
orange >COLOR
120000 TO period
480 SRUN

red >COLOR
100000 TO period
480 SRUN

green >COLOR
80000 TO period
480 SRUN

What is the effect of doubling the initial dose?
yellow >COLOR
80000 TO period
2 Y0 * 5 ASSIGN-IC
480 SRUN



Example 4 Lorenz Attractor - The Butterfly Effect

Equations: dx/dt = a(y-x) dy/dt = bx -xz – y dz/dt = xy - cz

```
pub (LORENZ) ;

DECIMAL

13 NODES

pub LORENZ  4 7 integr  5 4 INV   8 13 integr  6 5 8 SUM  7 1 K()  6 POT  9 2 K()  4 POT
      10 4 8 MULT  14 16 integr  11 4 14 MULT  12 8 9 11 SUM  13 12 INV
      15 3 K() 14 POT   16 10 15 SUM  ;

10000 8  ASSIGN-IC

1000000000 TO Ym

#30000  1 >K()      \ a – Prandtl Number
-#280000 2 >Y()     \ b – Raleigh Number
-#10000  3 >K()     \ c -- Proportion

20 >DT
10 TO comint

14 8 4 OUT-NODES

SET-TO output DISPLAY
SET-TO integr EULER
SET-TO model LORENZ

CHECK
   4:  0
   5:  0
   6:  10000
   7:  30000
   8:  10000
   9:  0
  10:  0
  11:  0
  12:  10000
  13:  -10000
  14:   0
  15 :  0
  16:   0
```

SET-TO output PLOTXY \ y-axis = node 4 x-axis = node 8

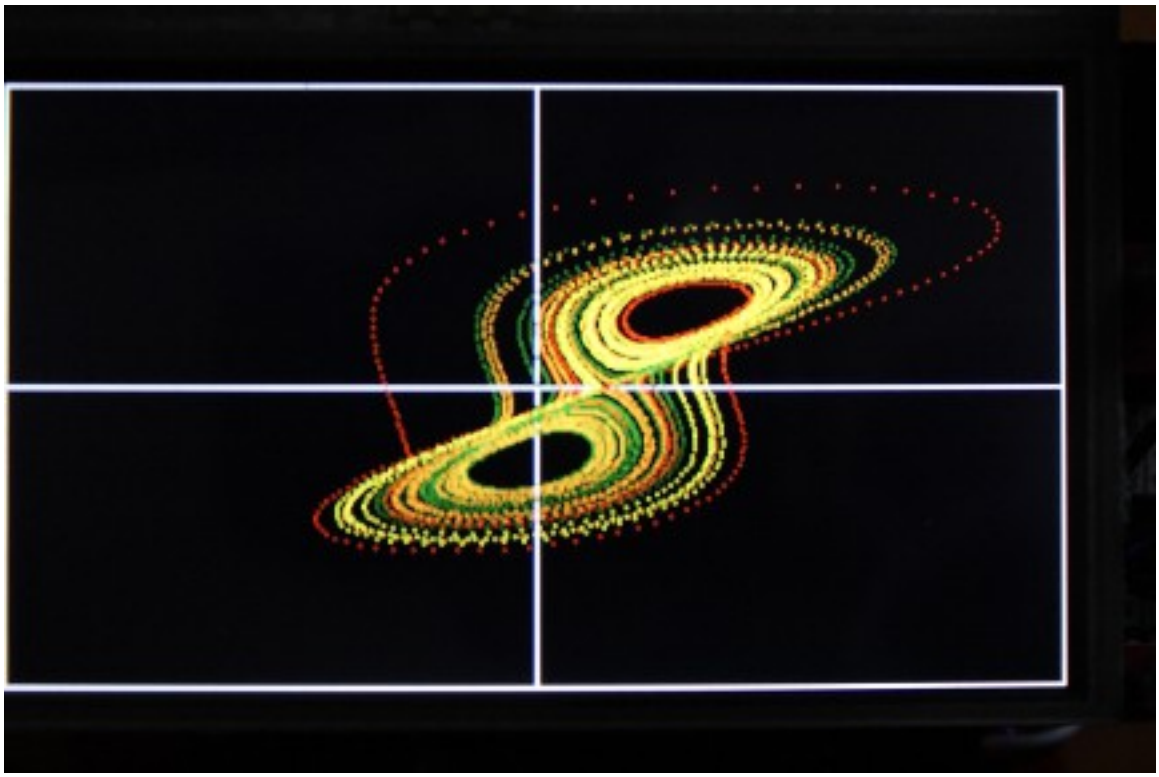
gph3 \ 250 >XGAIN 500 >YGAIN

red >COLOR
2000 SRUN

yellow >COLOR
2000 CONT

orange >COLOR
2000 CONT

green >COLOR
2000 CONT



Example 5 Automobile Suspension System (1 wheel)

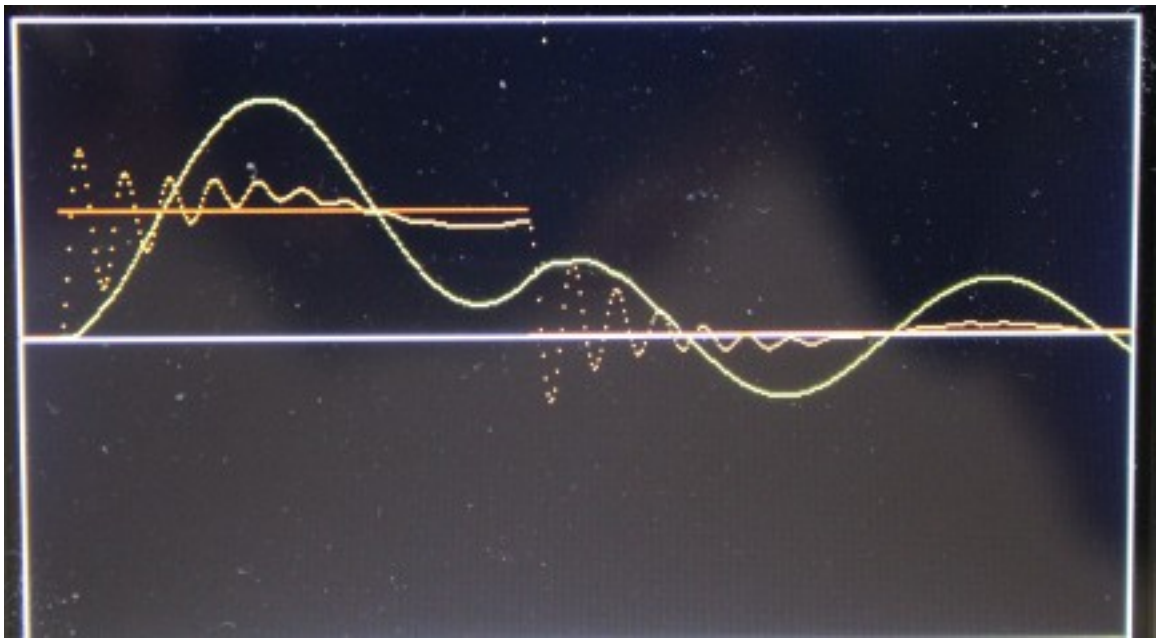
Equations: $-d^2x1/dt^2 = (D/M1)*(dx1/dt - dx2/dt) + (K1/M1)*(x1-cx2)$
$-d^2x2/dt^2 = (D/M2)*(dx2/dt – dx1/dt) + (K1/M2)*(x2-x1) + (K2/M2)*(x3-x2)$
x1 is the displacement of the chassis in feet
x2 is the displacement of the axle in feet
Y0 = 1 foot Time Scale = seconds

```
pub (AUTOSS) ;

DECIMAL

18 NODES

pub AUTOSS
 4 10 EULER  5 4 EULER  6 5 13 SUM  7 1 K() 6 POT  11 4 12 SUM
 8 3 K() 11 POT   9 7 8 SUM  10 9 INV  12 20 EULER  13 12 EULER
 14 13 INV  21 sinput 15 13 21 SUM  17 2 K() 6 POT  16 5 K() 15 POT
 18 4 K() 11 POT  19 17 18 16 SUM  20 19 INV  ;

400000 12  ASSIGN-IC

25 == M1      \  mass of chassis in slugs
2 == M2       \ mass of axle+wheel in slugs
1000 == K1    \ spring constant of shock absorber lbs/ft
5000 == K2    \ spring constant of tire lbs/ft
200 == D      \  shock absorber dampening factor lbs/ft/sec

pub CPAR
 K1 scale M1 */ 1 >K()   K1 scale M2 */ 2 >K()
 D scale M1 */ 3 >K()    D scale M2 */ 4 >K()    K2 scale M2 */ 5 >K()  ;

20 >DT
10 TO comint

1000 >XT
6000 TO period
400000 TO ye
0 TO yo

16 5 14 OUT-NODES

SET-TO sinput PULSE
SET-TO model AUTOSS
SET-TO precalc CPAR
SET-TO output PLOTY2

CHECK

 4 – 0              5 – 0              6 – 0              7 – 0
 8 – 3200000        9 – 3200000        10 - -3200000      11 – 400000    12 – 400000
13 – 0             14 – 0             15 – 0             16 – 0         17 – 0
18 – 40000000      19 – 40000000      20 - -40000000     21 – 0
```

gph2

The horizontal line (x3) represents a bump the tire rides over, simulated as a pulse.

20 TO D 480 SRUN



200 TO D 480 SRUN