Assignment 7, Problem 1

Output

```
Error in fprinted (Truth-RK4), NRK = 60
   1.0e-03 *

   0.398789293463153
  -0.987473224654423
  -0.474596608682987
  -0.094149778231412

Error in dfprinted_dxk (Truth-RK4), NRK = 60
  Columns 1 through 3

   0.000572308096395  -0.001159655457286  -0.000432789385229
  -0.001417144674122   0.002871402789140   0.001071593141972
  -0.000680893644585   0.001379609010627   0.000514846440026
  -0.000135292418975   0.000274081375373   0.000102246435489

  Column 4

  -0.001375859284138
   0.003406762923419
   0.001636876196642
   0.000325154432524

Error in dfprinted_dvk (Truth-RK4), NRK = 60
   0.000801764196950  -0.000382706990322   0.000120276205958
  -0.001985156317005   0.000947506003286  -0.000297719348332
  -0.000953823501703   0.000455157164936  -0.000143040332016
  -0.000189412739157   0.000090371925921  -0.000028311850169

Error in fprinted (Truth-RK4), NRK = 120
   1.0e-04 *

   0.260670163072518
  -0.645465154036629
  -0.310218559747000
  -0.061545895775339

Error in dfprinted_dxk (Truth-RK4), NRK = 120
   1.0e-03 *

  Columns 1 through 3
```

```
   0.037409378535358  -0.075801603543368  -0.028289484191646
  -0.092632583289287   0.187691084647668   0.070045280864406
  -0.044507024213658   0.090179007116831   0.033653316208415
  -0.008843414619264   0.017915479141095   0.006683481107217

  Column 4

  -0.089933890194516
   0.222685153630664
   0.106995302473933
   0.021254020239780

Error in dfprinted_dvk (Truth-RK4), NRK = 120
   1.0e-03 *

   0.052407640652063  -0.025015713546850   0.007861716468938
  -0.129760875154261   0.061934210805248  -0.019460391669668
  -0.062347199417445   0.029751905941566  -0.009349941812786
  -0.012381282139984   0.005907301392938  -0.001850890683386
```

```
clc;clear;close all;
format long
```

Inital State and Proc Noise, Etc..

```
N1 = 60;
N2 = 120;
t0 = 0;
tf = 3;
x0 = [-0.40; 0.85; -0.60; -1.65];
v0 =[-0.77; 1.30; 1.65];
idervflag = 1;
```

RK4 Calls

```
[xf1,dfprinted_dxk1,dfprinted_dvk1] = ...
             c2dnonlinear(x0,0,v0,t0,tf,N1,'fscript_ts01',idervflag);
[xf2,dfprinted_dxk2,dfprinted_dvk2] = ...
             c2dnonlinear(x0,0,v0,t0,tf,N2,'fscript_ts01',idervflag);
```

Truth Model

```
   A = ...
     [-0.43256481152822, -1.14647135068146,  0.32729236140865, ...
```

```
                    -0.58831654301419;...
      -1.66558437823810,  1.19091546564300,  0.17463914282092, ...
                     2.18318581819710;...
       0.12533230647483,  1.18916420165210, -0.18670857768144, ...
                    -0.13639588308660;...
       0.28767642035855, -0.03763327659332,  0.72579054829330, ...
                     0.11393131352081];
   D = ...
     [ 1.06676821135919,  0.29441081639264, -0.69177570170229;...
       0.05928146052361, -1.33618185793780,  0.85799667282826;...
      -0.09564840548367,  0.71432455181895,  1.25400142160253;...
      -0.83234946365002,  1.62356206444627, -1.59372957644748];

sysmodel_ct = ss(A,D,eye(4),zeros(4,3));
sysmodel_dt = c2d(sysmodel_ct,(3),'zoh');
[dfprinted_dxk_t,dfprinted_dvk_t] = ssdata(sysmodel_dt);
xft = dfprinted_dxk_t*x0 + dfprinted_dvk_t*v0;
```

Results

```
disp('Error in fprinted (Truth-RK4), NRK = 60');
disp((xft-xf1))

disp('Error in dfprinted_dxk (Truth-RK4), NRK = 60');
disp((dfprinted_dxk_t-dfprinted_dxk1))

disp('Error in dfprinted_dvk (Truth-RK4), NRK = 60');
disp((dfprinted_dvk_t-dfprinted_dvk1))


disp('Error in fprinted (Truth-RK4), NRK = 120');
disp((xft-xf2))

disp('Error in dfprinted_dxk (Truth-RK4), NRK = 120');
disp((dfprinted_dxk_t-dfprinted_dxk2))

disp('Error in dfprinted_dvk (Truth-RK4), NRK = 120');
disp((dfprinted_dvk_t-dfprinted_dvk2))
```

```
function [fprinted,dfprinted_dxk,dfprinted_dvk] = ...
            c2dnonlinear(xk,uk,vk,tk,tkp1,nRK,fscriptname,idervflag)
%
%  Copyright (c) 2002 Mark L. Psiaki.  All rights reserved.
%
%
%  This function derives a nonlinear discrete-time dynamics function
%  for use in a nonlinear difference equation via 4th-order
%  Runge-Kutta numerical integration of a nonlinear differential
%  equation.  If the nonlinear differential equation takes the
%  form:
%
%               xdot = fscript{t,x(t),uk,vk}
%
%  and if the initial condition is x(tk) = xk, then the solution
%  gets integrated forward from time tk to time tkp1 using nRK
%  4th-order Runge-Kutta numerical integration steps in order to
%  compute fprinted(k,xk,uk,vk) = x(tkp1).  This function can
%  be used in a nonlinear dynamics model of the form:
%
%        xkp1 = fprinted(k,xk,uk,vk)
%
%  which is the form defined in MAE 676 lecture for use in a nonlinear
%  extended Kalman filter.
%
%  This function also computes the first partial derivative of
%  fprinted(k,xk,uk,vk) with respect to xk, dfprinted_dxk, and with
%  respect to vk, dfprinted_dvk.
%
%
%  Inputs:
%
%    xk             The state vector at time tk, which is the initial
%                   time of the sample interval.
%
%    uk             The control vector, which is held constant
%                   during the sample interval from time tk to time
%                   tkp1.
%
%    vk             The discrete-time process noise disturbance vector,
%                   which is held constant during the sample interval
%                   from time tk to time tkp1.
%
%    tk             The start time of the numerical integration
%                   sample interval.
%
%    tkp1           The end time of the numerical integration
%                   sample interval.
%
%    nRK            The number of Runge-Kutta numerical integration
```

```
%                   steps to take during the sample interval.
%
%   fscriptname     The name of the Matlab .m-file that contains the
%                   function which defines fscript{t,x(t),uk,vk}.
%                   This must be a character string.  For example, if
%                   the continuous-time differential equation model is
%                   contained in the file rocketmodel.m with the function
%                   name rocketmodel, then on input to the present
%                   function fscriptname must equal 'rocketmodel',
%                   and the first line of the file rocketmodel.m
%                   must be:
%
%                   function [fscript,dfscript_dx,dfscript_dvtil] = ...
%                               rocketmodel(t,x,u,vtil,idervflag)
%
%                   The function must be written so that fscript
%                   defines xdot as a function of t, x, u, and vtil
%                   and so that dfscript_dx and dfscript_dvtil are the
%                   matrix partial derivatives of fscript with respect
%                   to x and vtil if idervflag = 1.  If idervflag = 0, then
%                   these outputs must be empty arrays.
%
%   idervflag       A flag that tells whether (idervflag = 1) or not
%                   (idervflag = 0) the partial derivatives
%                   dfprinted_dxk and dfprinted_dvk must be calculated.
%                   If idervflag = 0, then these outputs will be
%                   empty arrays.
%
%  Outputs:
%
%   fprinted        The discrete-time dynamics vector function evaluated
%                   at k, xk, uk, and vk.
%
%   dfprinted_dxk   The partial derivative of fprinted with respect to
%                   xk.  This is a Jacobian matrix.  It is evaluated and
%                   output only if idervflag = 1.  Otherwise, an
%                   empty array is output.
%
%   dfprinted_dvk   The partial derivative of fprinted with respect to
%                   vk.  This is a Jacobian matrix.  It is evaluated and
%                   output only if idervflag = 1.  Otherwise, an
%                   empty array is output.
%


%
%  Prepare for the Runge-Kutta numerical integration by setting up
%  the initial conditions and the time step.
%
   x = xk;
   if idervflag == 1
      nx = size(xk,1);
      nv = size(vk,1);
      F = eye(nx);
```

```
      Gamma = zeros(nx,nv);
   end
   t = tk;
   delt = (tkp1 - tk)/nRK;
%
%  This loop does one 4th-order Runge-Kutta numerical integration step
%  per iteration.  Integrate the state.  If partial derivatives are
%  to be calculated, then the partial derivative matrices simultaneously
%  with the state.
%

   for jj = 1:nRK
      if idervflag == 1
         [fscript,dfscript_dx,dfscript_dvtil] = ...
                   feval(fscriptname,t,x,uk,vk,1);
         dFa = ( dfscript_dx * F )*delt;
         dGammaa = ( dfscript_dx * Gamma + dfscript_dvtil )*delt;
      else
         fscript = feval(fscriptname,t,x,uk,vk,0);
      end
      dxa = fscript*delt;
%
      if idervflag == 1
         [fscript,dfscript_dx,dfscript_dvtil] = ...
                   feval(fscriptname,(t + 0.5*delt),(x + 0.5*dxa),...
                         uk,vk,1);
         dFb = ( dfscript_dx * ( F + 0.5 * dFa ) )*delt;
         dGammab = ( dfscript_dx * ( Gamma + 0.5 * dGammaa ) + dfscript_dvtil )*delt;
      else
         fscript = feval(fscriptname,(t + 0.5*delt),(x + 0.5*dxa),...
                         uk,vk,0);
      end
      dxb = fscript*delt;
%
      if idervflag == 1
         [fscript,dfscript_dx,dfscript_dvtil] = ...
                   feval(fscriptname,(t + 0.5*delt),(x + 0.5*dxb),...
                         uk,vk,1);
         dFc = ( dfscript_dx * ( F + 0.5 * dFb ) )*delt;
         dGammac = ( dfscript_dx * ( Gamma + 0.5 * dGammab ) + dfscript_dvtil )*delt;
      else
         fscript = feval(fscriptname,(t + 0.5*delt),(x + 0.5*dxb),...
                         uk,vk,0);
      end
      dxc = fscript*delt;
%
      if idervflag == 1
         [fscript,dfscript_dx,dfscript_dvtil] = ...
                   feval(fscriptname,(t + delt),(x + dxc),...
                         uk,vk,1);
         dFd = ( dfscript_dx * ( F + dFc ) )*delt;
         dGammad = ( dfscript_dx * ( Gamma + dGammac ) + dfscript_dvtil )*delt;
      else
```

```
         fscript = feval(fscriptname,(t + delt),(x + dxc),...
                         uk,vk,0);
      end
      dxd = fscript*delt;
%
      x = x + (dxa + 2*(dxb + dxc) + dxd)*(1/6);
      if idervflag == 1
         F = F + (dFa + 2*(dFb + dFc) + dFd)*(1/6);
         Gamma = Gamma + ...
                (dGammaa + 2*(dGammab + dGammac) + dGammad)*(1/6);
      end
      t = t + delt;
   end
%
%  Assign the results to the appropriate outputs.
%
   fprinted = x;
   if idervflag == 1
      dfprinted_dxk = F;
      dfprinted_dvk = Gamma;
   else
      dfprinted_dxk = [];
      dfprinted_dvk = [];
   end
```

```
function [fscript,dfscript_dx,dfscript_dvtil] = ...
                                fscript_ts01(t,x,u,vtil,idervflag)
%
%  Copyright (c) 2002 Mark L. Psiaki.  All rights reserved.
%
%
%  This function gives a dummy test case for the nonlinear numercial
%  integration function c2dnonlinear.m.  It is a linear case.
%  Equivalent outputs to those of c2dnonlinear.m should be derivable as
%
%           sysmodel_ct = ss(A,D,eye(4),zeros(4,3));
%           sysmodel_dt = c2d(sysmodel_ct,(tkp1-tk),'zoh');
%           [dfprinted_dxk,dfprinted_dvk] = ssdata(sysmodel_dt);
%           fprinted = dfprinted_dxk*xk + dfprinted_dvk*vk;
%
%  or using the old call format of c2d:
%
%           [dfprinted_dxk,dfprinted_dvk] = c2d(A,D,(tkp1-tk));
%           fprinted = dfprinted_dxk*xk + dfprinted_dvk*vk;
%
%  The differential equation in question is the following linear time-
%  invariant differential equation:
%
%    xdot = A*x + D*vtil
%
```

```
%
%  Inputs:
%
%    t               The time at which xdot is to be known.
%
%    x               The 4x1 state vector at time t.
%
%    u               The 0x1 control vector at time t.
%
%    vtil            The 3x1 process noise disturbance vector at time t.
%
%    idervflag       A flag that tells whether (idervflag = 1) or not
%                    (idervflag = 0) the partial derivatives
%                    dfscript_dx and dfscript_dvtil must be calculated.
%                    If idervflag = 0, then these outputs will be
%                    empty arrays.
%
%  Outputs:
%
%    fscript         The time derivative of x at time t as determined
%                    by the differential equation.
%
%    dfscript_dx     The partial derivative of fscript with respect to
%                    x.  This is a Jacobian matrix.  It is evaluated and
%                    output only if idervflag = 1.  Otherwise, an
%                    empty array is output.
%
%    dfscript_dvtil  The partial derivative of fscript with respect to
%                    vtil.  This is a Jacobian matrix.  It is evaluated and
%                    output only if idervflag = 1.  Otherwise, an
%                    empty array is output.
%


%
%  Set up the linear system matrices.
%
   A = ...
     [-0.43256481152822, -1.14647135068146,  0.32729236140865, ...
                   -0.58831654301419;...
      -1.66558437823810,  1.19091546564300,  0.17463914282092, ...
                    2.18318581819710;...
       0.12533230647483,  1.18916420165210, -0.18670857768144, ...
                   -0.13639588308660;...
       0.28767642035855, -0.03763327659332,  0.72579054829330, ...
                    0.11393131352081];
   D = ...
     [ 1.06676821135919,  0.29441081639264, -0.69177570170229;...
       0.05928146052361, -1.33618185793780,  0.85799667282826;...
      -0.09564840548367,  0.71432455181895,  1.25400142160253;...
      -0.83234946365002,  1.62356206444627, -1.59372957644748];
%
%  Calculate the outputs.
%
```

```
fscript = A*x + D*vtil;
if idervflag == 1
   dfscript_dx = A;
   dfscript_dvtil = D;
else
   dfscript_dx = [];
   dfscript_dvtil = [];
end
```