HW7 Problem 1 Final

Spencer Freeman

AOE 5784

12/17/2024

**Results:**

HW7-P1-Final

120 RK Steps

State Estimate:

fprinted =

1.0e+02 \*

0.620701210018915

-1.529444304253257

-0.705857744405961

-0.103775385949128

Partial Derivative wrt xk Estimate:

dfprinted\_dxk =

1.0e+02 \*

0.728877498626523 -1.472776463825067 -0.548136654443188 -1.753681011366440

-1.802276348157174 3.647524833920750 1.359299650355381 4.334068375928841

-0.870935186580203 1.754235467078160 0.646202266188082 2.077782611804557

-0.167654300505863 0.348202994601513 0.134817316015435 0.406122909801801

Partial Derivative wrt vk Estimate:

dfprinted\_dvk =

1.0e+02 \*

1.033625821871495 -0.478443883796926 0.161916522846359

-2.522534149758227 1.183721601826497 -0.385332453603791

-1.193825976395165 0.538038300656368 -0.143777850356074

-0.233714858740136 0.144761934545640 -0.037248695108266

60 RK Steps

State Estimate:

fprinted =

1.0e+02 \*

0.620697482796144

-1.529435074986165

-0.705853308658435

-0.103774505997242

Partial Derivative wrt xk Estimate:

dfprinted\_dxk =

1.0e+02 \*

0.623180322654275 -1.258552351010682 -0.468159647121948 -1.499667430804622

-1.540539945582156 3.117091929068315 1.161314212345298 3.704913219247961

-0.745300454509505 1.499428602715482 0.550897764793195 1.775392564654726

-0.142571006994151 0.297577105024353 0.116036029385876 0.345874841165753

Partial Derivative wrt vk Estimate:

dfprinted\_dvk =

1.0e+02 \*

0.885599711582125 -0.407521904148829 0.139724403897416

-2.155886828683541 1.008425181308094 -0.330347147756422

-1.017332286172073 0.453506086768662 -0.116838467575323

-0.198737287889482 0.128531955988640 -0.032161523410142

Error in state estimate is 10 times larger halving the number of steps. Error in derivatives is nearly doubled.

>>

**Script hw7\_prob1\_final.m :**

%% Fix the code to compute the required partial derivatives.

% Spencer Freeman, 12/17/2024

% AOE 5784, Estimation and Filtering

%

% This script solves number 1 of problem set 7

% -------------------------------------------------------------------------

clear;clc;close all

disp('HW7-P1-Final')

format long

% Test your code using the supplied test function "fscript\_ts01.m".

% Test the function by numerically integrating from a random initial

% condition and using a random process noise vector. Integrate over a

% time span of 3 seconds, and use 120 4th-order Runge-Kutta numerical

% integration steps. Compare your results with the exact results, which

% you can computed as outlined in the initial comments section of

% "fscript\_ts01.m". Test the results again, but this time use only 60

% 4th-order Runge-Kutta steps for the same inputs. Does the error for

% this second case change as you expect it to change in comparison with

% the error for the first numerical integration case?

tk = 0; % s

tkp1 = 3; % s

% xk = rand(4, 1);

% vk = rand(3, 1);

% uk = [];

xk = [-0.40; 0.85; -0.60; -1.65];

uk = [];

vk = [-0.77; 1.30; 1.65];

idervflag = true;

fscriptname = 'fscript\_ts01';

[~, A, D] = fscript\_ts01(tk,xk,uk,vk,idervflag);

[dfprinted\_dxk\_true, dfprinted\_dvk\_true] = c2d(A, D, (tkp1 - tk));

fprinted\_true = dfprinted\_dxk\_true\*xk + dfprinted\_dvk\_true\*vk;

%% many steps

nRK = 120;

[fprinted, dfprinted\_dxk, dfprinted\_dvk] = ...

...

c2dnonlinear(xk,uk,vk,tk,tkp1,nRK,fscriptname,idervflag);

disp(string(nRK) + " RK Steps")

disp('State Estimate:')

fprinted\_true

fprinted

disp('Partial Derivative wrt xk Estimate:')

dfprinted\_dxk\_true

dfprinted\_dxk

disp('Partial Derivative wrt vk Estimate:')

dfprinted\_dvk\_true

dfprinted\_dvk

%% few steps

nRK = 60;

[fprinted, dfprinted\_dxk, dfprinted\_dvk] = ...

...

c2dnonlinear(xk,uk,vk,tk,tkp1,nRK,fscriptname,idervflag);

disp(string(nRK) + " RK Steps")

disp('State Estimate:')

fprinted\_true

fprinted

disp('Partial Derivative wrt xk Estimate:')

dfprinted\_dxk\_true

dfprinted\_dxk

disp('Partial Derivative wrt vk Estimate:')

dfprinted\_dvk\_true

dfprinted\_dvk

disp('Error in state estimate is 10 times larger halving the number of steps. Error in derivatives is nearly doubled.')

**Function c2dnonlinear.m :**

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); % ANSWER

Gamma = zeros(nx, nv); % ANSWER

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; % ANSWER

dGammaa = (dfscript\_dx \* Gamma + dfscript\_dvtil)\*delt; % ANSWER

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)\*delt; % ANSWER

dGammab = (dfscript\_dx \* Gamma + dfscript\_dvtil)\*delt; % ANSWER

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)\*delt; % ANSWER

dGammac = (dfscript\_dx \* Gamma + dfscript\_dvtil)\*delt; % ANSWER

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)\*delt; % ANSWER

dGammad = (dfscript\_dx \* Gamma + dfscript\_dvtil)\*delt; % ANSWER

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