Exam Solution

Course: SOW-MKI96 Neuromorphic Computing
Assignment Source:brightspace
Assignment Due Date: 20-09-28 10-30-00

Author:
collaborative effort

06-09-2019

0 Introduction

This contains an exam solution. If you wish to contribute to this exam solution:

1. Create a github account, (you can create an "anonymous" one).
2. git clone ...
3. edit your changes in the document.
4. open cmd, and browse to inside the folder you downloaded and edited
5. git pull (updates your local repository=copy of folder, to the latest version in github cloud)
6. git status shows which files you changed.
7. git add "/some folder with a space/someFileYouChanged.tex"
8. git commit -m "Included solution to question 1c."
9. git push

It can be a bit initimidating at first, so feel free to click on "issue" in the github browser of this repository and ask :) (You can also use that to say "Hi, I'm having a bit of help with this particular equation, can someone help me out?")

If you don't know how to edit a latex file on your own pc iso on overleaf, look at the "How to use" section of `https://github.com/a-t-0/AE4872-Satellite-Orbit-Determination`.

0.1 Consistency

To make everything nice and structured, please use very clear citations:

1. If you copy/use an equation of some slide or document, please add the following data:
   (a) Url (e.g. if simple wiki or some site)
   (b) Name of document
   (c) (Author)
   (d) PAGE/SLIDE number so people can easily find it again
   (e) equation number (so people can easily find it again)
2. If you use an equation from the slides/a book that already has an equation number, then hardcode that equation number in this solution manual so people directly see which equation in the lecture material it is, this facilitates remembering the equations.
3. Here is an example is given in eq. (10.32[1]) (See file references.bib [1]).

$$R_n^E = \sum_{t=1}^{n} l_m(p_t, z_t) - \min_i \sum_{t=1}^{n} {z_t}^i \tag{10.32[1]}$$

1 Programming

Conclusion

References

[1] Some author. *Advanced tree dynamics*, volume lecture 5 of *AE2344 Some course*, page 15. Accessed: 2019-04-27.

A Appendix: ExportPlotToLatex.m

```
%% clear console and data
clear all
close all
clc

%% example problem: generating a halo orbit
% you can replace this part with your own code.

% declare and initialise parameters
mp = 1000;
mq = 1;
alpha_p = [-1; 0];
alpha_q = [1000; 0];
G = 1;
mu_p = G * mp;
mu_q = G * mq;
d_p = 1;
d_q = 1000;
n = sqrt((mu_p + mq)/(d_p + d_q)^3);

% write down integration span
t_span = [0:1:500000];

% create initial state (a_1,a_1_dot,a_2,a_2_dot):
alpha_init = [-1001; 0; 0;0];

% call differential equations with ODE45
[t,alpha] = ode45(@(t,alpha) odefcn18_3(t,alpha,alpha_p,alpha_q,mu_p,mu_q,d_p,d_q,n), t_span,

%% This is where you configures and create the plot (keep this)
% declare dataseries (in this case 3, can be more)
x_series = java.util.ArrayList(); %omg can has java in matlab
y_series = java.util.ArrayList();
z_series = java.util.ArrayList();

% declare axis scales
axisScales = java.util.ArrayList();

% declare axis domains
axis_domains = java.util.ArrayList();

% declare and initialise plot parameters
currentFolder = "/code/";
latexDestination ="latex/images/";
fileName = 'plot_1d';
relativePath = '../latex/Images/'; % the ../ goes up one folder
exportType = 'eps'; % can be eps or jpeg
lineColours = 'blue';
nrOfDimensions = 2;

% set axis labels
axisLabelRotation =- ones(3,1)*90; % 0 for no rotation (currently only rotates y-axis)
y_axis_label = '$\displaystyle\frac{X{g_0}}{C_{eff}^2}$';
axisLabels = ["example x axis label", y_axis_label];

% set custom axis domains:
setAxisDomain = true;
x_axis_domain = [-1090 1080];
y_axis_domain = [-1100 1070];
axis_domains.add(x_axis_domain)
```

```
axis_domains.add(y_axis_domain);

% create custom axis scales
setCustomScales = true; % set to false to disable custom axis scales
x_axis_scale = [0,10,-4,5];
y_axis_scale = ['a';'b';'c'];
z_axis_scale = [0:1:10];
axisScales.add(x_axis_scale);
axisScales.add(y_axis_scale);
axisScales.add(z_axis_scale); % can also do this in loop

% create x-series (can be as much as you like)
x_series1 = alpha(:,1);
x_series2 = [2,3,100];
x_series.add(x_series1);
x_series.add(x_series2);

% create y-series (can be as much as you like)
y_series1 = alpha(:,3);
y_series2 = [6,7,6];
y_series3 = [6,7,700];
y_series.add(y_series1);
y_series.add(y_series2);
y_series.add(y_series3);

% put data series in java ArrayList() object
dataSeries = java.util.ArrayList();
dataSeries.add(x_series);
dataSeries.add(y_series);

% create legends for dataseries
y_series1_label = "halo orbit path of 3rd body";
y_series2_label = '$\displaystyle\frac{X{g_0}}{C_{eff}^2}$';
y_series3_label = "third line";

legend = [y_series1_label;y_series2_label;y_series3_label];
legendLocation = 'best'; % left doesn't work yet
plotType = "lines"; % scatter doesnt work yet

% create plot object containing all info for plot
plotData = PlotData(fileName,relativePath,exportType,...
        dataSeries,lineColours, nrOfDimensions,axisLabels,legend,...
        legendLocation, plotType,axisScales,currentFolder,...
        latexDestination,setAxisDomain,axis_domains, setCustomScales,...
        axisLabelRotation);

% plot the dataseries automatically to latex
obj_mult = PlotMultipleLines;
plot_altitudes(obj_mult,plotData);

%% Create a quick 2nd figure:
x_series.clear(); % java
x_series.add(alpha(:,1)) % java
y_series.clear(); % java
y_series.add(alpha(:,3)) % java
filename = "different_picture";
exportType = 'jpeg'; % can be eps or jpeg
setCustomScales = false; % set to false to disable custom axis scales

plotDataTwo = PlotData(fileName,relativePath,exportType,...
        dataSeries,lineColours, nrOfDimensions,axisLabels,legend,...
        legendLocation, plotType,axisScales,currentFolder,...
```

```
        latexDestination,setAxisDomain,axis_domains, setCustomScales,...
        axisLabelRotation);

plot_altitudes(obj_mult,plotDataTwo);

%% ODE equations to compute orbit, you can remove this, it's for an example
function dalphadt = odefcn18_3(t,alpha,alpha_p,alpha_q, mu_p,mu_q,d_p,d_q,n)
    % declare and initialise parameters
    dalphadt = zeros(4,1);
    r_p = sqrt((alpha(1)-alpha_p(1))^2+(alpha(3)-alpha_p(2))^2);
    r_q = sqrt((alpha(1)-alpha_q(1))^2+(alpha(3)-alpha_q(2))^2);

    % Implement ODE
    dalphadt(1) = alpha(2);
    dalphadt(2) = -mu_p*((alpha(1) + d_p)/(r_p)^3) - mu_q*((alpha(1) - d_q)/((r_q)^3)) + 2*n*
    dalphadt(3) = alpha(4);
    dalphadt(4) = -mu_p*((alpha(3))/(r_p)^3) - mu_q*((alpha(3))/((r_q)^3)) - 2*n*alpha(2) + n
end
```