ASTE 586 Computer Project Part 1

Introduction

This report details the methodology and solution used to complete the ASTE 586 Spacecraft and Attitude Dynamics Computer Project (Part 1).

The goal of this project was to numerically integrate a given initial value problem and compare the approximated solution to an exact analytical solution.

The initial value problem was given as a linear system of four Ordinary Differential Equations:
$\dot{x_1} = 2x_2(t)$
$\dot{x_2} = -2x_1(t)$
$\dot{x_3} = 200x_4(t)$
$\dot{x_4} = -200x_3(t)$

The initial values were given as:
$x_1(0) = 1$
$x_2(0) = 0$
$x_3(0) = 1$
$x_4(0) = 0$

The numerical and analytical solutions were to be compared to at least $t = 10$, ultimately achieving a maximum allowed difference of 10^-3 or less for all four components of x.

The final demonstration to assess the accuracy of the numerical solution was to plot $x_1^2(t) + x_2^2(t) + x_3^2(t) + x_4^2(t) - 2$ versus time to demonstrate the result is very close to zero.

Analytical Solution

To solve the initial value problem, I recognized it was given in the form $\frac{d}{dt}\vec{x}(t) = [A]\vec{x}(0)$, which has a solution of $\vec{x}(t) = e^{At}\vec{x}(0)$.

To find e^{At} I found the eigenvalues of A by solving $det(A - \lambda I) = 0$ for λ. This resulted in 4 eigenvalues:

$\lambda_{1,2} = +/- 2i$
$\lambda_{3,4} = +/- 200i$

Next I solved $(A - \lambda_i I)\vec{v}_i = \vec{0}$ to find the eigenvectors corresponding to each eigenvalue.

$\vec{v}_1 = [1, i, 0, 0]$
$\vec{v}_2 = [1, -i, 0, 0]$
$\vec{v}_3 = [0, 0, 1, i]$
$\vec{v}_3 = [0, 0, 1, -i]$

Using the general solution of the form:
$\vec{x}(t) = c_1 e^{\lambda_1 t}\vec{v_1} + c_2 e^{\lambda_2 t}\vec{v_2} + c_3 e^{\lambda_3 t}\vec{v_3} + c_4 e^{\lambda_4 t}\vec{v_4}$

Next substitute the corresponding eigenvectors and eigenvalues:
$\vec{x}(t) = c_1 e^{2it}[1, i, 0, 0] + c_2 e^{-2it}[1, -i, 0, 0] + c_3 e^{200it}[0, 0, 1, i] + c_4 e^{-200it}[0, 0, 1, -i]$
$x_1(t) = c_1 e^{2it} + c_2 e^{-2it}$
$x_2(t) = ic_1 e^{2it} - ic_2 e^{-2it}$
$x_3(t) = c_3 e^{200it} + c_4 e^{-200it}$
$x_4(t) = ic_3 e^{200it} - ic_2 e^{-200it}$

Use Euler's Formula: $e^{i\omega t} = cos(\omega t) + isin(\omega t)$
$x_1(t) = A_1 cos(2t) + A_2 sin(2t)$
$x_2(t) = -A_1 sin(2t) + A_2 cos(2t)$
$x_3(t) = A_3 cos(200t) + A_4 sin(200t)$
$x_4(t) = -A_3 sin(200t) + A_4 cos(200t)$
where
$A_1 = c_1 + c_2$
$A_2 = i(c_1 - c_2)$
$A_3 = c_3 + c_4$
$A_4 = i(c_3 - c_4)$

Now to solve for the coefficient vector $\vec{A}$, evaluate at the initial condition $t = 0$.

$\vec{A} = [1, 0, 1, 0]$

This gives the analytical solution to the linear system of equations:

$\vec{x}(t) = [cos(2t), -sin(2t), cos(200t), -sin(200t)]$

Numerical Integration

My numerical integrations strategy was to use an already-built method of the Python library `scipy`. This allowed me to minimize scripting, debugging, and testing time.

The `scipy.integrate.solve_ivp()` function allows for many different numerical integration methods and tolerances to be used. I chose the Runge-Kutta45 method and specified desired relative and absolute tolerances of 10^-8. This combination ensured the solver met the specified accuracy from the problem statement while not being too computationally expensive.

Plotting and Comparing Results

Due to the linear system containing different orders of effects (x_1 & x_2 on different order of magnitude than x_3 & x_4), I chose to create two sets of plots with different "step sizes" for clarity.

Figure 1 shows four plots.

1. Analytical Solution evaluated from $t = [0, 10]$ with step size of 0.100.
2. Numerical Solution evaluated from $t = [0, 10]$ with step size of 0.100.
3. Absolute Error (defined as $x_{analytical}(t) - x_{numerical}(t)$) for the same range and step size.
4. The result of $x_1^2(t) + x_2^2(t) + x_3^2(t) + x_4^2(t) - 2$ for the same range and step size.

ASTE 586 Computer Project Part I

* This plot shows the whole range t=[0,10] with a relatively big step size for plot clarity. *

Figure 1 shows that the numerical solution is within the tolerances specified in the problem statement, but it does not show an accurate representation for the x_3 & x_4 terms due to the relatively large step size.

Figure 2 contains the same set of four plots, but evaluated from $t = [0, 10]$ with step size of 0.00001. The plots are then only shown from $t = [9.9, 10.0]$ as this is the most interesting region due to the highest amount of propagation error.

ASTE 586 Computer Project Part I

* This plot shows the end of the evaluated range with a much more discrete step size. *

These four plots show the behavior of x_3 & x_4 much more accurately and serve as a good final confirmation that the goal of the computer project has been satisfied.

The maximum absolute error (*note all maximum errors do not occur at the same value of* t*): $x_1: 7.550*10^{-15}x_2: 7.661*10^{-15}x_3: 1.193*10^{-05}x_4: 1.194*10^{-05}$*

The tolerance based on $x_1^2(t) + x_2^2(t) + x_3^2(t) + x_4^2(t) - 2$ was: $[-2.340 * 10^{-05},\ 2.346 * 10^{-09}]$

Conclusion

The initial value problem was solved analytically and numerically. The accuracy of the numerical solver met the requirements and is also easily tuned for different required accuracies within the method. For further details on the solver script and the analytical derivation, please see the attachments to this report.

Appendix A: Python Script

File - /Users/gertha/Library/CloudStorage/GoogleDrive-andygerth1@gmail.com/My Drive/USC_MSAA/ASTE586/Computer_Project/ASTE

```
## ASTE 586 Computer Project
##      Part 1
## Andrew Gerth
## 20250209

import numpy as np
import scipy as sp
from matplotlib import pyplot as plt

# Define Analytical Solution (see paper work)
def x_fun_analytical(t):
    ## Input: time "t"
    ## Output: 4 element tuple of solutions for x(t)
    x1t = np.cos(2*t)
    x2t = -np.sin(2*t)
    x3t = np.cos(200*t)
    x4t = -np.sin(200*t)
    x_t = [x1t, x2t, x3t, x4t]
```

Attachment B: Detailed Derivation of Analytical Solution##

ASTE 586 Computer Project Part I | Andrew Gerth

diff eqs:

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \\ \dot{x}_4(t) \end{bmatrix} = \begin{bmatrix} 0 & 2 & 0 & 0 \\ -2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 200 \\ 0 & 0 & -200 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix}$$