ASTE 586 Computer Project Part I | Andrew Gerth

diff eq's:

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

initial cond. $\vec{x}(0) = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$

given in form $\frac{d}{dt}\vec{x}(t) = [A]\vec{x}(t)$

use matrix exponential: $\vec{x}(t) = e^{At}\vec{x}(0)$

find e^{At} by finding eig vals of A. $\det(A - \lambda I) = 0$

$$\det \begin{bmatrix} -\lambda & 2 & 0 & 0 \\ -2 & -\lambda & 0 & 0 \\ 0 & 0 & -\lambda & 200 \\ 0 & 0 & -200 & -\lambda \end{bmatrix} = 0 \Rightarrow$$

$$\Rightarrow -\lambda \begin{vmatrix} -\lambda & 0 & 0 \\ 0 & -\lambda & 200 \\ 0 & -200 & -\lambda \end{vmatrix} + 2 \begin{vmatrix} 2 & 0 & 0 \\ 0 & -\lambda & 200 \\ 0 & -200 & -\lambda \end{vmatrix}$$

$$-\lambda\left[-\lambda\left[\lambda^2 + 40000\right]\right] + 2\left[2\left[\lambda^2 + 40000\right]\right] = \lambda^4 + 40000\lambda^2 + 4\lambda^2 + 160000 = 0$$

$$\lambda^4 + 40004\lambda^2 + 160000 = 0$$

$\lambda_{1,2} = \pm(2i)$

$\lambda_{3,4} = \pm(200i)$

for $\lambda_1 = 2i$ Solve $(A - 2iI)\vec{x} = \vec{0} \Rightarrow$

$$\begin{bmatrix} -2i & 2 & 0 & 0 \\ -2 & -2i & 0 & 0 \\ 0 & 0 & -2i & 200 \\ 0 & 0 & -200 & -2i \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$-2ix_1 + 2x_2 = 0 \Rightarrow x_2 = ix_1$ choose $x_1 = 1$

$-2ix_3 + 200x_4 = 0 \Rightarrow x_4 = \frac{ix_3}{100}$ $x_3 = 0$

$$\vec{v}_1 = \begin{bmatrix} 1 \\ i \\ 0 \\ 0 \end{bmatrix}$$

for $\lambda_2 = -2i$ $\quad (A+2iI)\vec{v}_2 = \vec{0}$

$$\begin{bmatrix} 2i & 2 & 0 & 0 \\ -2 & 2i & 0 & 0 \\ 0 & 0 & 2i & 200 \\ 0 & 0 & -200 & 2i \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$x_2 = -ix_1$

$x_4 = -\frac{ix_3}{100}$

$$\vec{v}_2 = \begin{bmatrix} 1 \\ -i \\ 0 \\ 0 \end{bmatrix}$$

for $\lambda_3 = +200i$

$$\begin{bmatrix} -200i & 2 & 0 & 0 \\ -2 & -200i & 0 & 0 \\ 0 & 0 & -200i & 200 \\ 0 & 0 & -200 & -200i \end{bmatrix}$$

$$\vec{v}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ i \end{bmatrix}, \quad \vec{v}_4 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ -i \end{bmatrix}$$

General Soln:

$$x(t) = c_1 e^{\lambda_1 t}\vec{v}_1 + c_2 e^{\lambda_2 t}\vec{v}_2 + c_3 e^{\lambda_3 t}\vec{v}_3 + c_4 e^{\lambda_4 t}\vec{v}_4$$

Sub $\vec{\lambda}$ & $\vec{v}$

$$x(t) = c_1 e^{2it}\begin{bmatrix} 1 \\ i \\ 0 \\ 0 \end{bmatrix} + c_2 e^{-2it}\begin{bmatrix} 1 \\ -i \\ 0 \\ 0 \end{bmatrix} + c_3 e^{200it}\begin{bmatrix} 0 \\ 0 \\ 1 \\ i \end{bmatrix} + c_4 e^{-200it}\begin{bmatrix} 0 \\ 0 \\ 1 \\ -i \end{bmatrix}$$

$$\vec{x}(t) = \begin{bmatrix} c_1 e^{2it} \\ ic_1 e^{2it} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} c_2 e^{-2it} \\ -ic_2 e^{-2it} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ c_3 e^{200it} \\ ic_3 e^{200it} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ c_4 e^{-200it} \\ -ic_4 e^{-200it} \end{bmatrix}$$

$$\vec{x}(t) = \begin{bmatrix} c_1 e^{2it} + c_2 e^{-2it} \\ ic_1 e^{2it} - ic_2 e^{-2it} \\ c_3 e^{200it} + c_4 e^{-200it} \\ ic_3 e^{200it} - ic_4 e^{-200it} \end{bmatrix}$$

use euler's formula $e^{i\omega t} = \cos(\omega t) + i\sin(\omega t)$

$$x_1(t) = c_1[\cos(2t) + i\sin(2t)] + c_2[\cos(-2t) + i\sin(-2t)]$$

$$x_2(t) = ic_1[\cos(2t) + i\sin(2t)] - ic_2[\cos(-2t) + i\sin(-2t)]$$

$$x_2(t) = c_1\cos(2t) + c_1 i\sin(2t) + c_2\cos(2t) - c_2 i\sin(2t)$$

$$= \cos(2t)[c_1 + c_2] + i\sin(2t)[c_1 - c_2]$$

call $c_1 + c_2 \equiv A_1$ & $i[c_1 - c_2] \equiv A_2$

$$x_2(t) = ic_1\cos(2t) + ic_1\sin(2t) - ic_2\cos(2t) + c_2\sin(2t)$$

$$x_2(t) = \cos(2t)[ic_1 - ic_2] + \sin(2t)[c_1 + c_2]$$

same A_1 & A_2 → continue on next page.

In terms of A_1 & A_2

$x_1(t) = A_1\cos(2t) + A_2\sin(2t)$ ... same pattern for λ_3 & λ_4

$x_2(t) = -A_1\sin(2t) + A_2\cos(2t)$

$x_3(t) = A_3\cos(200t) + A_4\sin(200t)$

$x_4(t) = -A_3(\sin(200t)) + A_4\cos(200t)$

bring back initial cond. $\vec{x}(0) = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$

$t = 0$

$1 = A_1(1) + 0$

$0 = A_2(1)$

$1 = A_3(1) + 0$

$0 = A_4(1)$

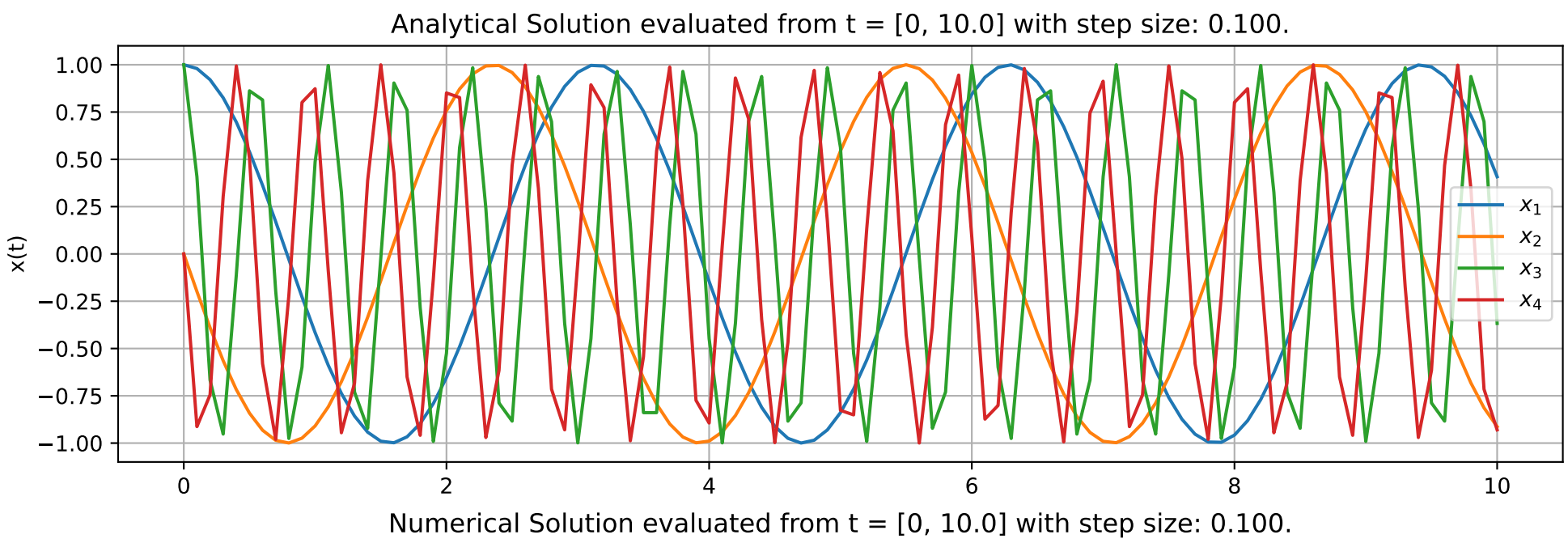
$\rightarrow \vec{A} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$

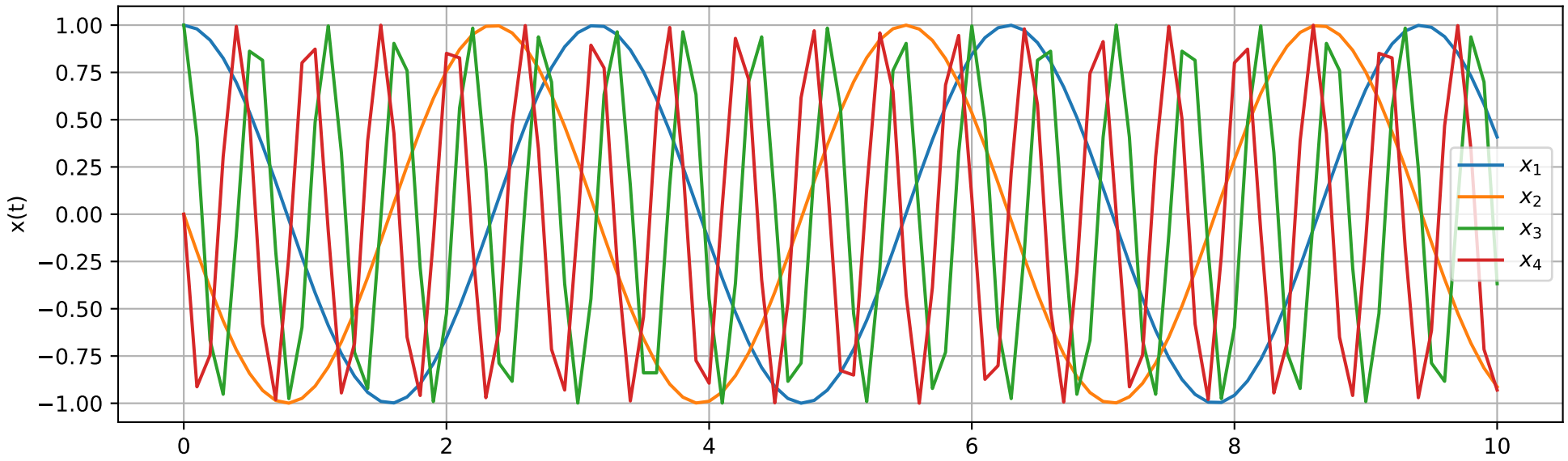
now plug back into $\vec{x}(t)$

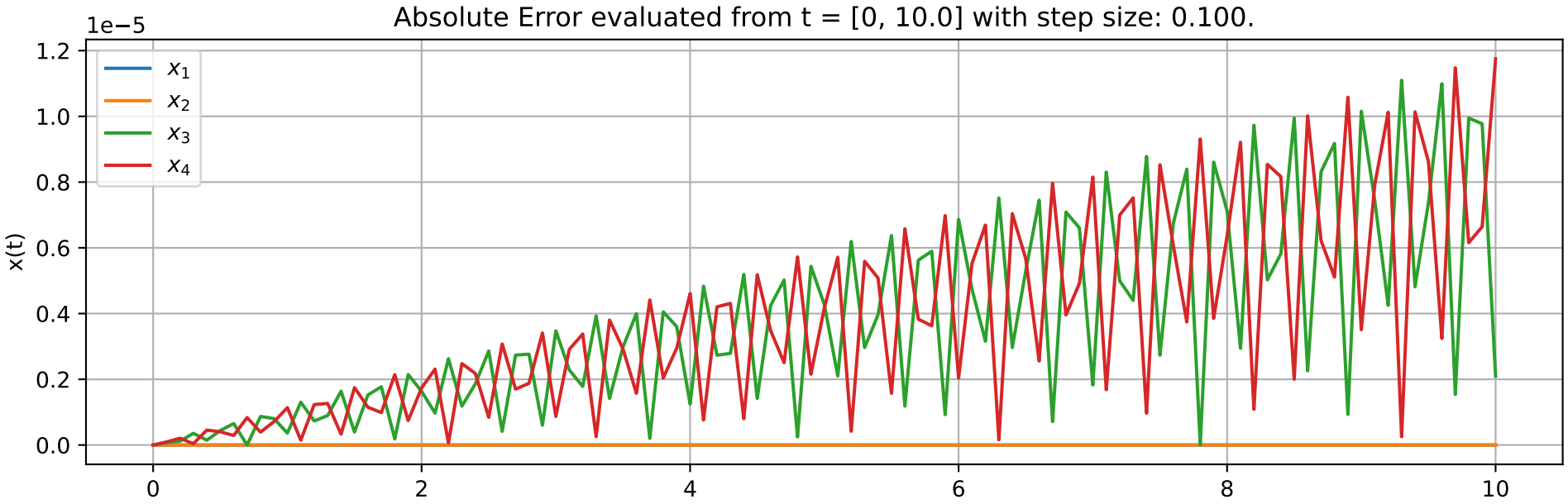
$$\vec{x}(t) = \begin{bmatrix} \cos(2t) \\ -\sin(2t) \\ \cos(200t) \\ -\sin(200t) \end{bmatrix}$$

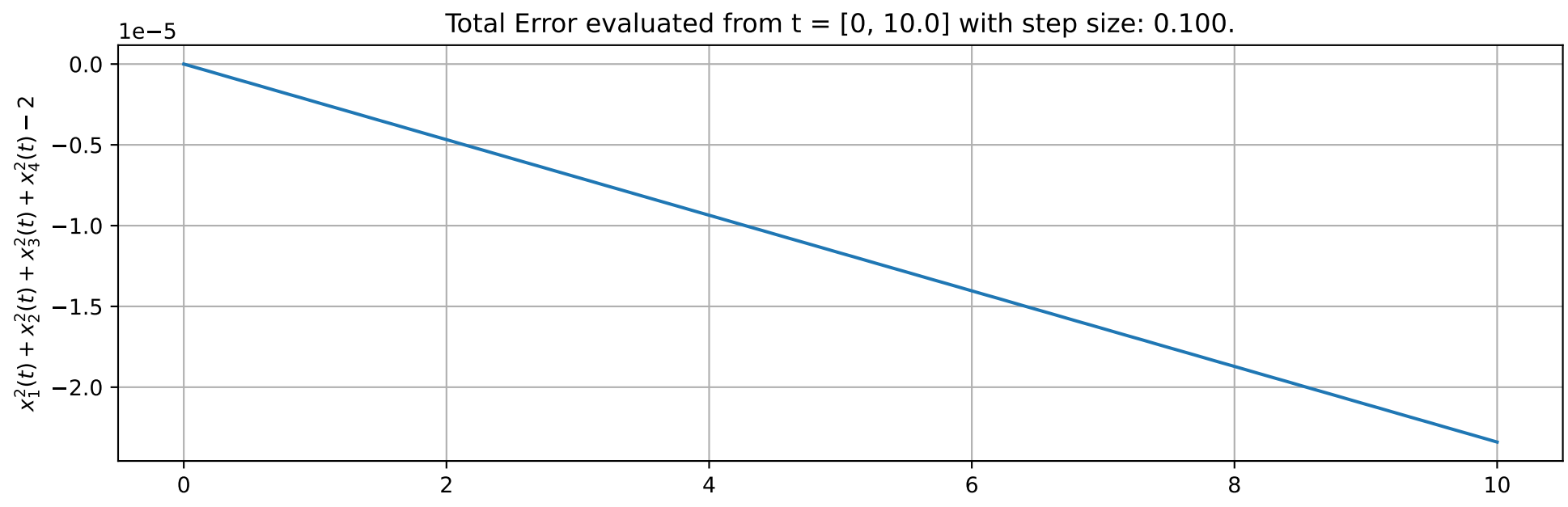
ASTE 586 Computer Project Part I

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







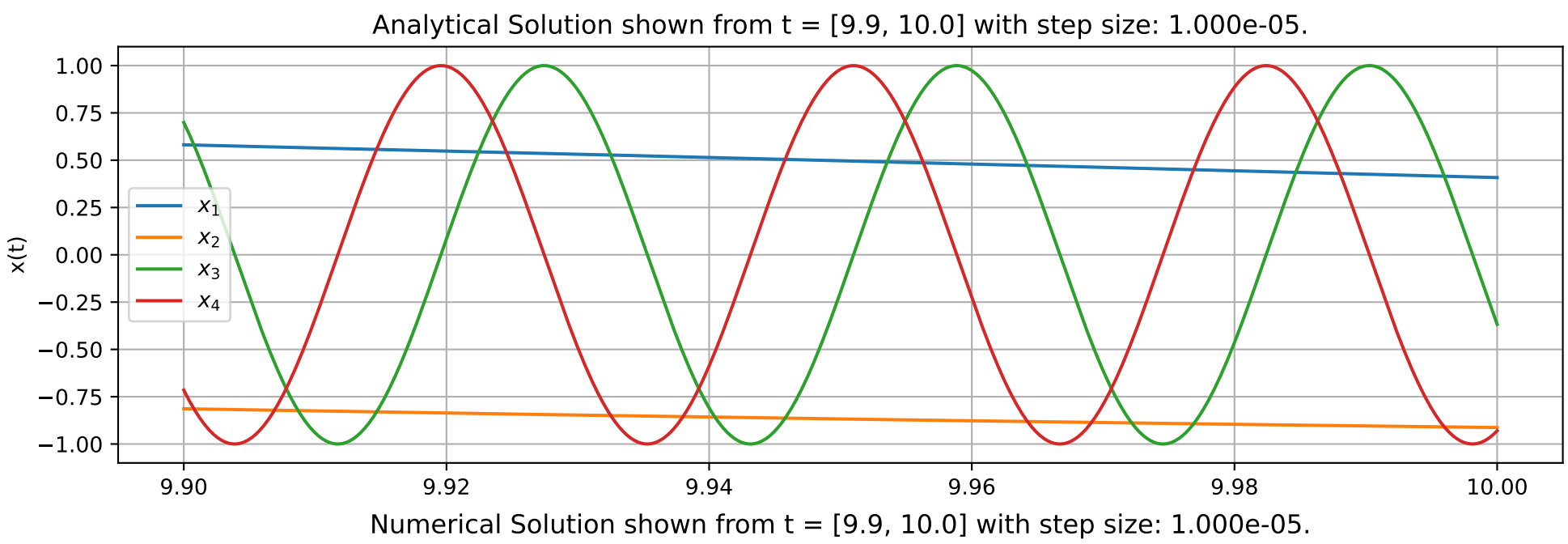


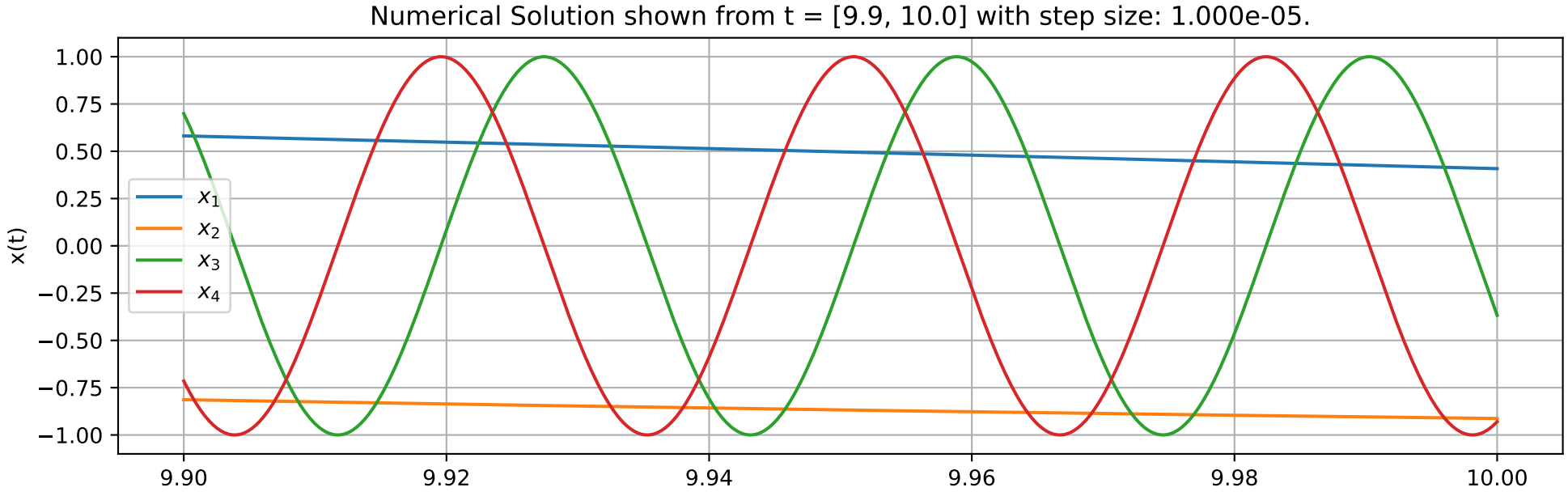
Maximum Error for x1: 7.327e-15
Maximum Error for x2: 7.438e-15
Maximum Error for x3: 1.110e-05
Maximum Error for x4: 1.175e-05

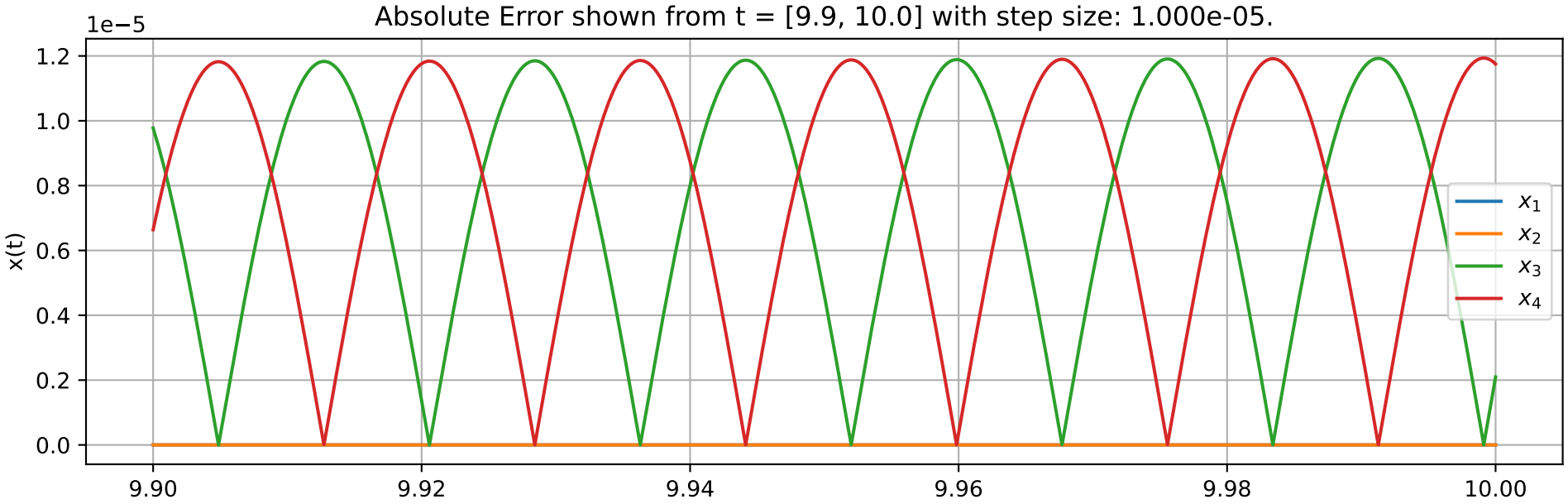
Tolerance based on $x_1^2(t) + x_2^2(t) + x_3^2(t) + x_4^2(t) - 2$: [-2.340e-05, 0.000e+00]

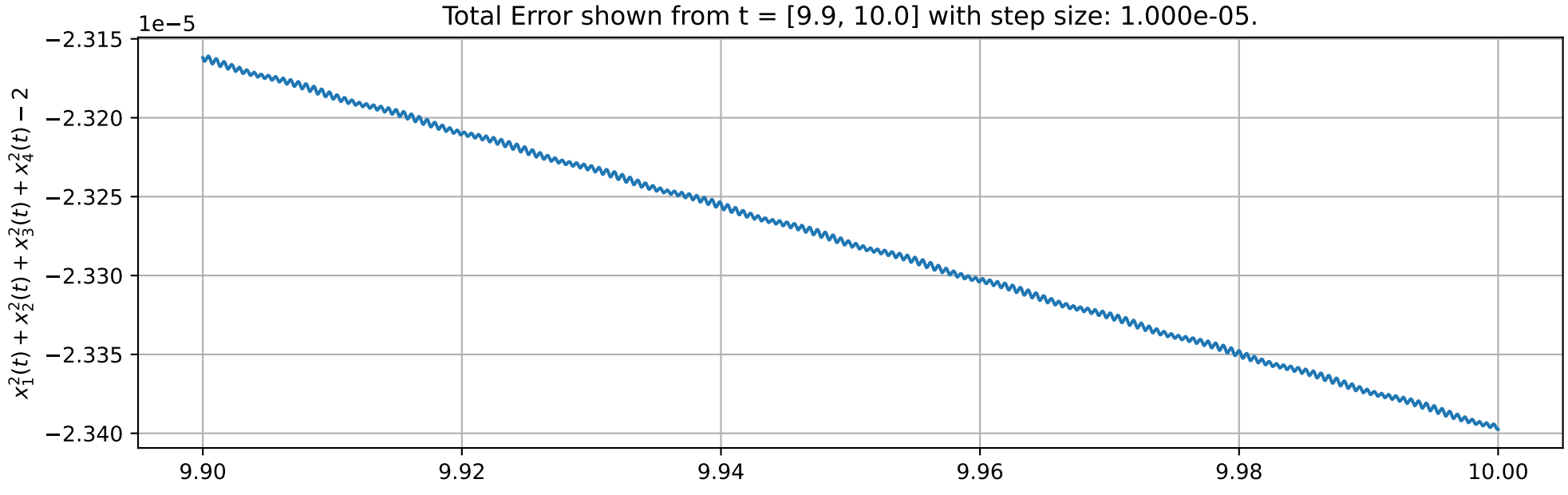
ASTE 586 Computer Project Part I

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









Maximum Error for x1: 7.550e-15
Maximum Error for x2: 7.661e-15
Maximum Error for x3: 1.193e-05
Maximum Error for x4: 1.194e-05

Tolerance based on $x_1^2(t) + x_2^2(t) + x_3^2(t) + x_4^2(t) - 2$: [-2.340e-05, 2.346e-09]

```python
## 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]

    return x1t, x2t, x3t, x4t

# Define diff eq system for Numerical Solver in scipy
def dx_fun(t, x):
    A = np.array([[0, 2, 0, 0],
                  [-2, 0, 0, 0],
                  [0, 0, 0, 200],
                  [0, 0, -200, 0]])
    return A @ x

# Define initial values
initial_values = np.array([1, 0, 1, 0])

t = np.linspace(0,10,101) # Define t as a vector
x_analytical = np.zeros((len(t), 4)) # Define empty matrix for x values to go into

# For loop to evaluate x(t) and then assign into x_analytical matrix
for i in range(0, len(t)):
    (x_analytical[i, 0], x_analytical[i, 1], x_analytical[i, 2], x_analytical[i, 3]) =
x_fun_analytical(t[i])

# Numerical Solving-time
result = sp.integrate.solve_ivp(dx_fun,
                                    t_span=[t[0], t[len(t)-1]],
                                    y0=initial_values,
                                    t_eval=t,
                                    method='RK45',
                                    rtol=1E-8,
                                    atol=1E-8,
                                    vectorized=True)
#print(x_numerical.y[:, 0])
#print(x_numerical.y)
x_numerical = np.transpose(result.y)

## Calculate Error
# Absolute Error
err_absolute = abs(x_analytical - x_numerical)
#print(err_absolute)
max_err_absolute = np.max(err_absolute, axis=0)
#print(max_err_absolute) # Find max of each column (x1, x2, x3, x4)
polynomial_check = x_numerical[:, 0]**2 + x_numerical[:, 1]**2 + x_numerical[:, 2]**2 +
x_numerical[:, 3]**2 - 2
#print(polynomial_check)
```

```
61
62
63 ## Report Results
64 text_results1 = ('Maximum Error for x1: {:.3e}\n'
65                 'Maximum Error for x2: {:.3e}\n'
66                 'Maximum Error for x3: {:.3e}\n'
67                 'Maximum Error for x4: {:.3e}'.format(max_err_absolute[0],
   max_err_absolute[1], max_err_absolute[2], max_err_absolute[3]))
68
69 text_results2 = r'Tolerance based on $x_1^2(t) + x_2^2(t) + x_3^2(t) + x_4^2(t) - 2$
   :   ' + '[{:.3e}, {:.3e}]\n'.format(min(polynomial_check), max(polynomial_check))
70
71 print(text_results1)
72 print(text_results2)
73
74
75 ## Plotting zone
76 fig, ax = plt.subplots(4, 1, figsize=(12,16))
77 fig.suptitle('ASTE 586 Computer Project Part I\n\n* This plot shows the whole range t=[0
   ,10] with a relatively big step size for plot clarity. *')
78 fig.canvas.manager.set_window_title('Plot 1')
79
80 ax[0].plot(t, x_analytical)
81 ax[1].plot(t, x_numerical)
82 ax[2].plot(t, err_absolute)
83 ax[3].plot(t, polynomial_check)
84
85 ax[0].set_title('Analytical Solution evaluated from t = [0, {}] with step size: {:.3f}.'
   .
86             format(t[len(t)-1], t[1]))
87 ax[1].set_title('Numerical Solution evaluated from t = [0, {}] with step size: {:.3f}.'.
88             format(t[len(t)-1], t[1]))
89 ax[2].set_title('Absolute Error evaluated from t = [0, {}] with step size: {:.3f}.'.
90             format(t[len(t)-1], t[1]))
91 ax[3].set_title('Total Error evaluated from t = [0, {}] with step size: {:.3f}.'.
92             format(t[len(t)-1], t[1]))
93
94 for i in range(0, len(ax)-1):
95     ax[i].legend(['$x_1$', '$x_2$', '$x_3$', '$x_4$'])
96     ax[i].set_ylabel('x(t)')
97 for i in range(0, len(ax)):
98     #ax[i].set_xlabel('t')
99     ax[i].grid(True)
100 ax[3].set_ylabel(r'$x_1^2(t) + x_2^2(t) + x_3^2(t) + x_4^2(t) - 2$')
101
102 ax[3].text(0.025,-0.45, text_results1, transform=ax[3].transAxes, bbox=dict(facecolor='
    gray', alpha=0.25))
103 ax[3].text(0.4,-0.35, text_results2, transform=ax[3].transAxes, bbox=dict(facecolor='
    gray', alpha=0.25))
104
105
106 ## Plotting zone (zoomed in with smaller range, smaller step size
107 del t, x_analytical, x_numerical, result, err_absolute, max_err_absolute,
    polynomial_check
108
109 t = np.linspace(0,10,1000001) # Define t as a vector
110 x_analytical = np.zeros((len(t), 4)) # Define empty matrix for x values to go into
111
112 # For loop to evaluate x(t) and then assign into x_analytical matrix
113 for i in range(0, len(t)):
114     (x_analytical[i, 0], x_analytical[i, 1], x_analytical[i, 2], x_analytical[i, 3]) =
    x_fun_analytical(t[i])
```

```
115
116 # Numerical Solving-time
117 result = sp.integrate.solve_ivp(dx_fun,
118                                  t_span=[t[0], t[len(t)-1]],
119                                  y0=initial_values,
120                                  t_eval=t,
121                                  method='RK45',
122                                  rtol=1E-8,
123                                  atol=1E-8,
124                                  vectorized=True)
125 #print(x_numerical.y[:, 0])
126 #print(x_numerical.y)
127 x_numerical = np.transpose(result.y)
128
129 ## Calculate Error
130 # Absolute Error
131 err_absolute = abs(x_analytical - x_numerical)
132 #print(err_absolute)
133 max_err_absolute = np.max(err_absolute, axis=0)
134 #print(max_err_absolute) # Find max of each column (x1, x2, x3, x4)
135 polynomial_check = x_numerical[:, 0]**2 + x_numerical[:, 1]**2 + x_numerical[:, 2]**2 +
    x_numerical[:, 3]**2 - 2
136 #print(polynomial_check)
137
138
139 ## Report Results
140 text_results1 = ('Maximum Error for x1: {:.3e}\n'
141                  'Maximum Error for x2: {:.3e}\n'
142                  'Maximum Error for x3: {:.3e}\n'
143                  'Maximum Error for x4: {:.3e}'.format(max_err_absolute[0],
    max_err_absolute[1], max_err_absolute[2], max_err_absolute[3]))
144
145 text_results2 = r'Tolerance based on $x_1^2(t) + x_2^2(t) + x_3^2(t) + x_4^2(t) - 2$
    :   ' + '[{:.3e}, {:.3e}]\n'.format(min(polynomial_check), max(polynomial_check))
146
147 print(text_results1)
148 print(text_results2)
149
150
151 index9 = np.where(t == 9.900)[0][0]
152 #print(index9)
153 #print(type(index9))
154
155 fig2, ax2 = plt.subplots(4, 1, figsize=(12,16))
156 fig2.suptitle('ASTE 586 Computer Project Part I\n\n* This plot shows the end of the
    evaluated range with a much more discrete step size. *')
157 fig2.canvas.manager.set_window_title('Plot 2 (Zoomed In)')
158
159 ax2[0].plot(t[index9:], x_analytical[index9:, :])
160 ax2[1].plot(t[index9:], x_numerical[index9:, :])
161 ax2[2].plot(t[index9:], err_absolute[index9:, :])
162 ax2[3].plot(t[index9:], polynomial_check[index9:])
163
164 ax2[0].set_title('Analytical Solution shown from t = [{}, {}] with step size: {:.3e}.'.
165             format(t[index9], t[len(t)-1], t[1]))
166 ax2[1].set_title('Numerical Solution shown from t = [{}, {}] with step size: {:.3e}.'.
167             format(t[index9], t[len(t)-1], t[1]))
168 ax2[2].set_title('Absolute Error shown from t = [{}, {}] with step size: {:.3e}.'.
169             format(t[index9], t[len(t)-1], t[1]))
170 ax2[3].set_title('Total Error shown from t = [{}, {}] with step size: {:.3e}.'.
171             format(t[index9], t[len(t)-1], t[1]))
172
```

```
173 for i in range(0, len(ax2)-1):
174     ax2[i].legend(['$x_1$', '$x_2$', '$x_3$', '$x_4$'])
175     ax2[i].set_ylabel('x(t)')
176 for i in range(0, len(ax)):
177     #ax2[i].set_xlabel('t')
178     ax2[i].grid(True)
179 ax2[3].set_ylabel(r'$x_1^2(t) + x_2^2(t) + x_3^2(t) + x_4^2(t) - 2$')
180 
181 ax2[3].text(0.025,-0.45, text_results1, transform=ax2[3].transAxes, bbox=dict(facecolor=
    'gray', alpha=0.25))
182 ax2[3].text(0.4,-0.35, text_results2, transform=ax2[3].transAxes, bbox=dict(facecolor='
    gray', alpha=0.25))
183 
184 fig.savefig("ASTE586_ComputerProject_Part1_Plot1.pdf")
185 fig2.savefig("ASTE586_ComputerProject_Part1_Plot2.pdf")
186 plt.show()
187 
188 
```

```
/usr/local/bin/python3.13 /Users/gertha/Library/CloudStorage/GoogleDrive-andygerth1@gmail.com/My Drive/USC_MSAA/ASTE586/Computer_Project/ASTE586_ComputerProject_Part1.py
Maximum Error for x1: 7.327e-15
Maximum Error for x2: 7.438e-15
Maximum Error for x3: 1.110e-05
Maximum Error for x4: 1.175e-05
Tolerance based on $x_1^2(t) + x_2^2(t) + x_3^2(t) + x_4^2(t) - 2$:   [-2.340e-05, 0.000e+00]

Maximum Error for x1: 7.550e-15
Maximum Error for x2: 7.661e-15
Maximum Error for x3: 1.193e-05
Maximum Error for x4: 1.194e-05
Tolerance based on $x_1^2(t) + x_2^2(t) + x_3^2(t) + x_4^2(t) - 2$:   [-2.340e-05, 2.346e-09]

2025-02-09 13:37:33.085 Python[38090:3323804] +[IMKClient subclass]: chose IMKClient_Modern
2025-02-09 13:37:33.085 Python[38090:3323804] +[IMKInputSession subclass]: chose IMKInputSession_Modern

Process finished with exit code 0

```