Name and section: ______________________________

ID number: ______________________________

E-mail: ______________________________

1. (30 points) Consider the **linear** ordinary differential equation (ODE)

$$u' = \lambda u, \quad u = u(t). \tag{1}$$

If $z = k\lambda$, where k is the time step, then for **one-step methods** we arrive at

$$U^{n+1} = R(z)U^n, \tag{2}$$

where $U^{n+1} \approx u(t_n)$, and $R(z)$ is a rational function. For an r-stage Runge-Kutta (RK) method, these polynomials in z have degree at most r. For an explicit method, $R(z)$ will be a polynomial of degree r and for an implicit method it will be a more general rational function.

Since $u(t_{n+1}) = e^z u(t_n)$ is the exact solution to Eq. (1), we expect that a pth-order accurate method will give a function $R(z)$ satisfying

$$R(z) = e^z + \mathcal{O}\left(z^{p+1}\right), \quad \text{as} \quad z \to 0.$$

For a given method at hand, we can thus determine the value of p by expanding e^z in a Taylor series about $z = 0$, writing the $\mathcal{O}\left(z^{p+1}\right)$ term as

$$Cz^{p+1} + \mathcal{O}\left(z^{p+2}\right),$$

multiplying through by the denominator of $R(z)$, and then collecting terms.

Then, **determine** $R(z)$ and p for the following methods:

(a) (10 points) The **trapezoidal** method

$$\frac{U^{n+1} - U^n}{k} = \frac{1}{2}\left[f(U^n) + f(U^{n+1})\right]. \tag{3}$$

(b) (10 points) The **backward Euler's** method

$$\frac{U^{n+1} - U^n}{k} = f(U^{n+1}). \tag{4}$$

(c) (10 points) The **TR-BDF2** (trapezoidal backward differentiation formula) or **DIRK** (diagonally implicit Runge-Kutta) method

$$\begin{aligned} Y_1 &= U^n, \\ Y_2 &= U^n + \frac{k}{4}\left[f\left(t_n, Y_1\right) + f\left(t_n + \frac{k}{2}, Y_2\right)\right], \\ Y_3 &= U^n + \frac{k}{3}\left[f\left(t_n, Y_1\right) + f\left(t_n + \frac{k}{2}, Y_2\right) + f\left(t_n + k, Y_3\right)\right], \\ U^{n+1} &= Y_3 = U^n + \frac{k}{3}\left[f\left(t_n, Y_1\right) + f\left(t_n + \frac{k}{2}, Y_2\right) + f\left(t_n + k, Y_3\right)\right]. \end{aligned} \tag{5}$$

2. (a) (7 points) Consider the **linear difference equation**

$$U^{n+2} = U^n, \tag{10}$$

together with given **starting values** U^0 and U^1. Upon using the roots of the associated characteristic polynomial, find the **particular solution** of Eq. (10).

(b) (13 points) A **Fibonacci sequence** is generated by starting with $F_0 = 0$ and $F_1 = 1$ and summing the last two terms to get the next term in the sequence, so

$$F_{n+1} = F_n + F_{n-1}. \tag{11}$$

Then, show that for large n the ratio F_n/F_{n-1} approaches the **golden ratio** $\phi = \left(1 + \sqrt{5}\right)/2 \approx 1.618034$.

3. (20 points) Which of the following Linear Multistep Methods (LMMs) are **convergent**? For the ones that **are not**, are they **inconsistent**, or **not zero-stable**, or **both**?

(a) (5 points) $U^{n+2} = \frac{1}{2}U^{n+1} + \frac{1}{2}U^n + 2kf(U^{n+1})$.

(b) (5 points) $U^{n+1} = U^n$.

(c) (5 points) $U^{n+4} = U^n + \frac{4}{3}k\left[f(U^{n+3}) + f(U^{n+2}) + f(U^{n+1})\right]$.

(d) (5 points) $U^{n+3} = -U^{n+2} + U^{n+1} + U^n + 2k\left[f(U^{n+2}) + f(U^{n+1})\right]$.

4. (a) (10 points) Consider the following **linear difference equations** together with **initial data**

$$U^{n+2} - U^{n+1} + 0.25U^n = 0, \quad U^0 = 2,\ U^1 = 3, \tag{17}$$

$$2U^{n+3} - 5U^{n+2} + 4U^{n+1} - U^n = 0, \quad U^0 = 11, U^1 = 5, U^2 = 1. \tag{18}$$

For each equation, **find** the **general solution**. Subsequently, determine the **particular solution** based on the initial data given. What is the value of U^{10}?

(b) (20 points) Consider the LMM

$$2U^{n+3} - 5U^{n+2} + 4U^{n+1} - U^n = k\left[\beta_0 f(U^n) + \beta_1 f(U^{n+1})\right]. \tag{19}$$

For **what values** of β_0 and β_1 is the local truncation error (LTE) $\mathcal{O}(k^2)$? Suppose you use the values of β_0 and β_1 just determined in this LMM. Is this a convergent method?

Hint: In class, we derived a **general formula** of the LTE for LMMs !

5. (30 points) Consider the **initial value problem** (IVP)

$$\begin{bmatrix} u_1' \\ u_2' \end{bmatrix} = \begin{bmatrix} -6 & 4 \\ 4 & -6 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}, \quad \mathbf{u}(0) = [2\ -8]^T, \quad t \in (0, 2], \tag{24}$$

whose **exact solution** is given by

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} -3e^{-2t} + 5e^{-10t} \\ -3e^{-2t} - 5e^{-10t} \end{bmatrix}. \tag{25}$$

Use both **forward** and **backward** Euler's methods with step sizes: $k = 0.2, 0.15, 0.1$, and $k = 0.05$ in order to approximate the solution for $t \in (0, 2]$. Note you implemented forward Euler's method in Homework Assignment 6 (see, Question 3 therein). As per the **backward Euler's** method, create a MATLAB function stored as `euler_back.m` whose first line should read

```
function [ uout, tout ] = euler_back( func, ti, tf, k, u0, tol, nmax )
```

where:

- `func`: is the name of your MATLAB function containing $f(t, u)$,
- `ti`: is the initial time t_i,
- `tf`: is the final time t_f,
- `k`: is the time step employed,
- `u0`: is the initial condition, i.e., $u_0 \doteq u(t_i)$,
- `tol`: is the **tolerance** in Newton's method,
- `nmax`: is the **maximum number** of Newton steps allowed,
- `uout`: is the output vector containing the solution at each time step, and
- `tout`: the output vector containing the corresponding times.

Then, make a figure containing the **first component** of the approximate solution to u_1, i.e., U_1 as a function of time for each case (hence, **8 plots in all** !). Do the methods behave similarly as the step size is varied? **Describe** the numerical results.

Attach **all** MATLAB codes and figures.

6. (30 points) Consider the so-called **kinetics problem**

$$\begin{aligned} u_1' &= -K_1 u_1 u_2 + K_2 u_3, \\ u_2' &= -K_1 u_1 u_2 + K_2 u_3, \\ u_3' &= K_1 u_1 u_2 - K_2 u_3, \end{aligned} \tag{26}$$

with $K_1 = 3$ and $K_2 = 1$ and initial data $\mathbf{u}(0) = [3\,4\,2]^T$.

(a) (10 points) Write a MATLAB code (or in **any** other programming language) to solve the above IVP using **forward Euler's method**. Choose a time step k based on the **stability analysis** that we discussed in class and determine whether the numerical solution **remains bounded** over, e.g., $0 \le t \le 8$ in this case. Make a plot of your approximate solutions each corresponding to the approximation of the components u_1, u_2 and u_3 as functions of time in **one single figure**. Please, state **which is which** !

(b) (5 points) How **large** can you choose k before you observe **instability** in your code?

(c) (15 points) Repeat parts (a) and (b) for $K_1 = 300$ and $K_2 = 1$.

Attach **all** MATLAB codes and figures.

7. (40 points) Consider the IVP

$$\theta'' = -a\theta - b\theta', \quad \theta = \theta(t),$$
$$\theta(0) = \theta'(0) = 1, \tag{29}$$

which is the **model for a swinging pendulum** where frictional forces are added. The **exact solution** to Eq. (29) is

$$\theta(t) = c_1 e^{\widetilde{\lambda}_1 t} + c_2 e^{\widetilde{\lambda}_2 t}, \tag{30}$$

where $\widetilde{\lambda}_{1,2}$ and $c_{1,2}$ are given by

$$\widetilde{\lambda}_1 = \frac{1}{2}\left(-b - \sqrt{b^2 - 4a}\right), \qquad \widetilde{\lambda}_2 = \frac{1}{2}\left(-b + \sqrt{b^2 - 4a}\right),$$
$$c_1 = \frac{1}{2} - \frac{b+2}{2\sqrt{b^2 - 4a}}, \qquad c_2 = \frac{1}{2} + \frac{b+2}{2\sqrt{b^2 - 4a}},$$

for given a and b.

(a) (15 points) Find the approximate solution of the IVP (29) for $t \in (0, 10]$ by employing the **two-step explicit Adams-Bashforth** method (AB2)

$$U^{n+2} = U^{n+1} + \frac{k}{2}\left[-f\left(t_n, U^n\right) + 3f\left(t_{n+1}, U^{n+1}\right)\right], \tag{31}$$

with $k = 0.05$, $a = 1$ and $b = 0$. To do so, create a MATLAB function stored as `ab2.m` whose first line should read

```
function [ uout, tout ] = ab2( func, ti, tf, k, u0 )
```

where the inputs and outputs are the same as those of Question 5 (except for `tol` and `nmax` !). Subsequently, make a plot of the exact [cf. Eq. (30)] and approximate solutions **on the same graph**. State **which is which** by including a legend (use e.g., a **dashed-dotted black line** for the exact solution and **red open circles** for the approximate one).

Hint: Recall that Eq. (31) requires **two starting values**, i.e., U^0 (the initial condition) and U^1. Initially, make use of your `rk.2.m` for **just one** step in order to find U^1, and then perform the time marching **forward in time** by using the AB2 method afterward.

(b) (25 points) Employ the **midpoint/leapfrog**

$$\frac{U^{n+1} - U^{n-1}}{2k} = f\left(t_n, U^n\right), \tag{32}$$

trapezoidal

$$\frac{U^{n+1} - U^n}{k} = \frac{1}{2}\left[f\left(t_n, U^n\right) + f\left(t_{n+1}, U^{n+1}\right)\right], \tag{33}$$

and **AB2** (see, part (a)) methods, respectively. To do so, **create two m-files** stored as `midpoint.m` and `trapezoidal.m` whose first lines should read

```
function [ uout, tout ] = midpoint( func, ti, tf, k, u0 )
function [ uout, tout ] = trapezoidal( func, ti, tf, k, u0, tol, nmax )
```

The inputs and outputs of the former are the same as those of part (a), whereas the respective ones of the latter are the same as those of Question 5. Note that all the above methods are **second-order accurate**.

Then, study the IVP (29) with $k = 0.05$ and find the approximate solution for $t \in (0, 5]$ for the following cases:

- $a = 100,\ b = 0$ (undamped),
- $a = 100,\ b = 3$ (damped),
- $a = 100,\ b = 10$ (more damped).

Compare the **exact** with the **approximate** solutions for each method, **comment on** and **explain** the behavior of each method. Make a plot for **each case** and **method** by presenting the exact and approximate solutions **on the same graph** (hence, **9 plots in all** !). As always, please, state **which is which** !

Attach **all** MATLAB codes and figures.

Noah Brayer

MATH 552 Scientific Computing II

Professor Charalampidis

7 May 2018

Homework 7

1. Determine $R(z)$ and p for the following methods:
   a. The trapezoidal method:

$$\frac{U^{n+1}-U^n}{k} = \frac{1}{2}[f(U^n) + f(U^{n+1})]$$

$$U^{n+1} - U^n = \frac{k}{2}[\lambda U^n + \lambda U^{n+1}] \to R(z)U^n - U^n = \frac{\lambda k}{2}[U^n + R(z)U^n] \to$$

$$R(z) - 1 = \frac{z}{2}[1 + R(z)] \to R(z)(2-z) = 2 + z \to R(z) = \frac{2+z}{2-z}$$

$$R(z) = \frac{2+z}{2-z} = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24} + \cdots + Cz^{p+1} + \mathcal{O}(z^{p+2})$$

   b. The backward Euler's method:

$$\frac{U^{n+1}-U^n}{k} = f(U^{n+1})$$

$$U^{n+1} - U^n = k\lambda U^{n+1} \to R(z)U^n - U^n = k\lambda R(z)U^n \to R(z)(1-z) = 1 \to$$

$$R(z) = \frac{1}{1-z} = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24} + \cdots + Cz^{p+1} + \mathcal{O}(z^{p+2})$$

   c. The TR-BDF2 or DIRK method:

$$Y_1 = U^n,$$

$$Y_2 = U^n + \frac{k}{4}\left[f(t_n, Y_1) + f\left(t_n + \frac{k}{2}, Y_2\right)\right],$$

$$Y_3 = U^n + \frac{k}{3}\left[f(t_n, Y_1) + f\left(t_n + \frac{k}{2}, Y_2\right) + f(t_n + k, Y_3)\right],$$

$$U^{n+1} = Y_3$$

$$Y_2 = U^n + \frac{k}{4}[\lambda U^n + \lambda Y_2] \to Y_2\left(1 - \frac{z}{4}\right) = U^n + \frac{z}{4}U^n \to Y_2 = \frac{4+z}{4-z}U^n$$

$$Y_3 = U^n + \frac{k}{3}[\lambda U^n + \lambda Y_2 + \lambda Y_3] \to Y_3\left(1 - \frac{z}{3}\right) = U^n\left(1 + \frac{z}{3}\right) + \frac{z}{3}\left(\frac{4+z}{4-z}\right)U^n \to$$

$$R(z)(3-z) = (3+z) + \left(\frac{4z + z^2}{4-z}\right) \to R(z) = \frac{12+5z}{(4-z)(3-z)}$$

$$R(z) = \frac{12+5z}{(4-z)(3-z)} = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{z^4}{24} + \cdots + Cz^{p+1} + \mathcal{O}(z^{p+2})$$

2.

a. Find the particular solution of:

$$U^{n+2} = U^n$$

b. Show that the ratio of sequential large Fibonacci numbers approaches the golden ratio

$$F_{n+1} = F_n + F_{n-1}, \qquad \frac{F_n}{F_{n-1}} = \phi$$

$$\frac{F_{n+1}}{F_n} = \frac{F_n}{F_{n-1}} \rightarrow \frac{F_n + F_{n-1}}{F_n} = \frac{F_n}{F_{n-1}} \rightarrow 1 + \frac{1}{\phi} = \phi \rightarrow \phi^2 - \phi - 1 = 0 \rightarrow \phi = \frac{1+\sqrt{5}}{2}$$

3. Which of the following LMMs are convergent?

a. $U^{n+2} = \frac{1}{2}U^{n+1} + \frac{1}{2}U^n + 2kf(U^{n+1})$

$$\alpha_0 = -\frac{1}{2}, \alpha_1 = -\frac{1}{2}, \alpha_2 = 1, \qquad \sum_{j=0}^{2} \alpha_j = 0, \qquad \sum_{j=0}^{2} j\alpha_j = 0 - \frac{1}{2} + 2 = \frac{3}{2}$$

$$\beta_0 = 0, \beta_1 = 2, \beta_2 = 0, \qquad \sum_{j=0}^{2} \beta_j = 2$$

Since $\sum_{j=0}^{2} \beta_j \neq \sum_{j=0}^{2} j\alpha_j$, it is inconsistent

$$\rho(\zeta) = \sum_{j=0}^{2} \alpha_j \zeta^j = -\frac{1}{2} - \frac{1}{2}\zeta + \zeta^2 = 0 \rightarrow \zeta = -\frac{1}{2}, 1$$

Since $\zeta = -\frac{1}{2}, 1$, it is zero-stable

b. $U^{n+1} = U^n$

$$\alpha_0 = -1, \alpha_1 = 1, \qquad \sum_{j=0}^{1} \alpha_j = 0, \qquad \sum_{j=0}^{1} j\alpha_j = 1$$

$$\beta_0 = 0, \beta_1 = 0, \qquad \sum_{j=0}^{1} \beta_j = 0$$

Since $\sum_{j=0}^{2} \beta_j \neq \sum_{j=0}^{2} j\alpha_j$, it is inconsistent

$$\rho(\zeta) = \sum_{j=0}^{1} \alpha_j \zeta^j = -1 + \zeta = 0 \rightarrow \zeta = 1$$

Since $\zeta = 1$, it is not zero-stable

c. $U^{n+4} = U^n + \frac{4}{3}k[f(U^{n+3}) + f(U^{n+2}) + f(U^{n+1})]$

$$\alpha_0 = -1, \alpha_1 = 0, \alpha_2 = 0, \alpha_3 = 0, \alpha_4 = 1, \qquad \sum_{j=0}^{4} \alpha_j = 0, \qquad \sum_{j=0}^{4} j\alpha_j = 4$$

$$\beta_0 = 0, \beta_1 = \frac{4}{3}, \beta_2 = \frac{4}{3}, \beta_3 = \frac{4}{3}, \beta_4 = 0, \qquad \sum_{j=0}^{4} \beta_j = 4$$

Since $\sum_{j=0}^{4} \beta_j = \sum_{j=0}^{4} j\alpha_j$, it is consistent

$$\rho(\zeta) = \sum_{j=0}^{4} \alpha_j \zeta^j = -1 + \zeta^4 = 0 \to \zeta = \pm i, \pm 1$$

Since $\zeta = \pm i, \pm 1$, it is not zero-stable

d. $U^{n+3} = -U^{n+2} + U^{n+1} + U^n = 2k[f(U^{n+2}) + f(U^{n+1})]$

$$\alpha_0 = -1, \alpha_1 = -1, \alpha_2 = 1, \alpha_3 = 1, \qquad \sum_{j=0}^{3} \alpha_j = 0, \qquad \sum_{j=0}^{3} j\alpha_j = -1 + 2 + 3 = 4$$

$$\beta_0 = 0, \beta_1 = 2, \beta_2 = 2, \beta_3 = 0, \qquad \sum_{j=0}^{3} \beta_j = 4$$

Since $\sum_{j=0}^{3} \beta_j = \sum_{j=0}^{3} j\alpha_j$, it is consistent

$$\rho(\zeta) = \sum_{j=0}^{3} \alpha_j \zeta^j = -1 - \zeta + \zeta^2 + \zeta^3 = 0 \to \zeta = \pm 1$$

Since $\zeta = \pm 1$, it is not zero-stable

4.

a. Find the general solution and particular solution for each equation:

i. $U^{n+2} - U^{n+1} + \frac{1}{4}U^n = 0, U^0 = 2, U^1 = 3$

ii. $2U^{n+3} - 5U^{n+2} + 4U^{n+1} - U^n = 0, U^0 = 11, U^1 = 5, U^2 = 1$

b. For what values of β_0 and β_1 is the LTE $\mathcal{O}(k^2)$? Is this a convergent method?

$$2U^{n+3} - 5U^{n+2} + 4U^{n+1} - U^n = k[\beta_0 f(U^n) + \beta_1 f(U^{n+1})]$$

$$\alpha_0 = -1, \alpha_1 = 4, \alpha_2 = -5, \alpha_3 = 2$$

$$\beta_2 = 0, \beta_3 = 0$$

$$\tau = \frac{1}{k}\{-u(t_n) + 4u(t_{n+1}) - 5u(t_{n+2}) + 2u(t_{n+3}) - k[\beta_0 u'(t_n) + \beta_1 u'(t_{n+1})]\}$$

$$= \frac{1}{k}\{-u(t_n) + 4u(t_n) + 4ku'(t_n) + 2k^2u''(t_n) + \frac{2}{3}k^3u'''(t_n) + \mathcal{O}(k^4) - 5u(t_n)$$

$$- 10ku'(t_n) - 10k^2u''(t_n) - \frac{20}{3}k^3u'''(t_n) + \mathcal{O}(k^4) + 2u(t_n)$$

$$+ 6ku'(t_n) + 9k^2u''(t_n) + 9k^3u'''(t_n) + \mathcal{O}(k^4)$$

$$- k[\beta_0 u'(t_n) + \beta_1 u'(t_{n+1})]\}$$

$$= \frac{1}{k}\{ku''(t_n) + 3k^3u'''(t_n) + \mathcal{O}(k^4) - k[\beta_0 u'(t_n) + \beta_1 u'(t_n) - \beta_1 ku''(t_n)$$

$$- \frac{1}{2}\beta_1 k^2 u'''(t_n) + \frac{1}{6}\beta_1 k^3 u''''(t_n) + \mathcal{O}(k^4)]$$

$$= -(\beta_0 + \beta_1)u'(t_n) + (1 - \beta_1)ku''(t_n) + \left(3 - \frac{1}{2}\beta_1\right)k^2u'''(t_n) + \mathcal{O}(k^3) \rightarrow$$

$$\beta_0 = -1, \beta_1 = 1, \tau = \frac{5}{2}u'''(t_n)k^2 + \mathcal{O}(k^3)$$

$$\sum_{j=0}^{3} j\alpha_j = 4 - 10 + 6 = 0$$

$$\sum_{j=0}^{3} \beta_j = -1 + 1 + 0 + 0 = 0$$

Since $\sum_{j=0}^{3} j\alpha_j = \sum_{j=0}^{3} \beta_j$, it is consistent

$$\rho(\zeta) = \sum_{j=0}^{3} \alpha_j \zeta^j = -1 + 4\zeta - 5\zeta^2 + 2\zeta^3 = 0 \rightarrow \zeta = \frac{1}{2}, 1$$

Since $\zeta = \frac{1}{2}, 1$, it is not zero-stable

5. Describe the numerical results:

   The backwards method did not work with larger values of k, and the forwards method oscillated for larger values of k.

6.

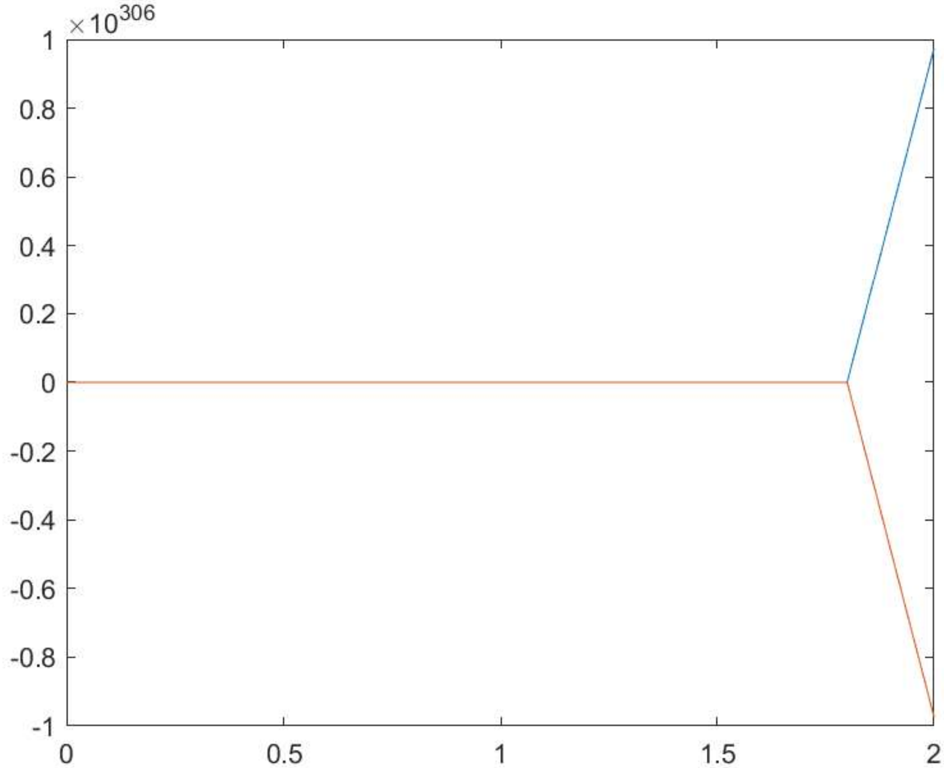
$$\lambda_1 = -K_1(u_1 + u_2) - K_2 = -3 * (3 + 4) - 1 = -22 \rightarrow k\lambda \approx -1 \rightarrow k \approx 0.05$$

$$\lambda_1 = -K_1(u_1 + u_2) - K_2 = -300 * (3 + 4) - 1 = -2101 \rightarrow k\lambda \approx -1 \rightarrow k \approx 5e - 4$$

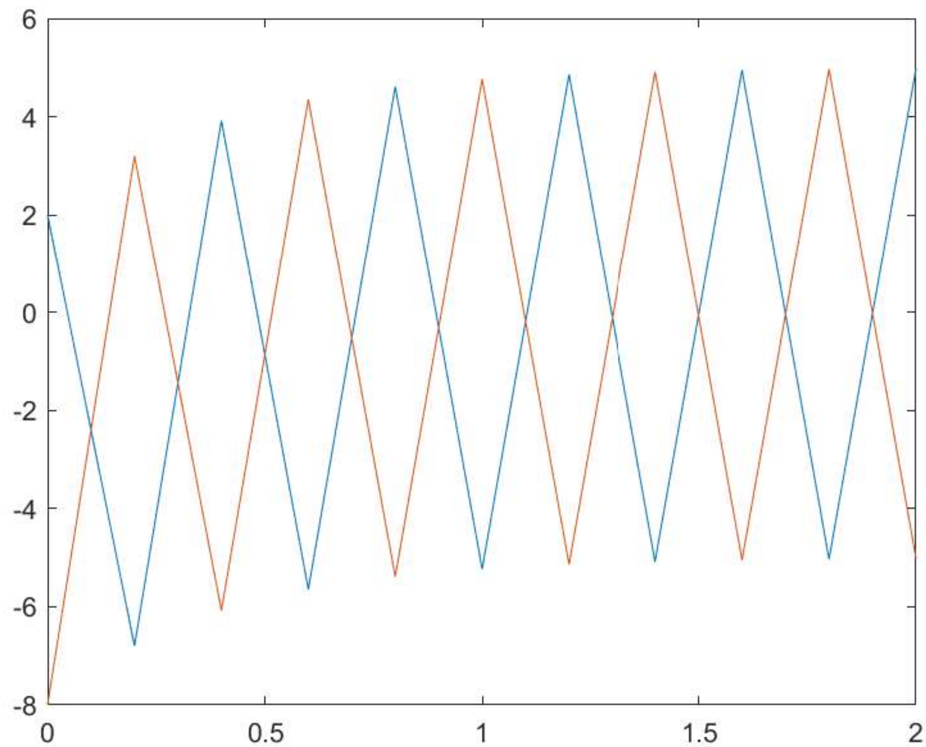
The graphs started showing instability around $k \approx 0.1$ and $k \approx 1e - 3$ respectively.

```
format long;
func=@(t,u)[-6 4; 4 -6]*u;
u0=[2;-8];
ti=0;
tf=2;
tol=1e-15;
nmax=100;

k=0.2;
[uout,tout]=euler_back(func,ti,tf,k,u0,tol,nmax);
uout
figure;
plot(tout,uout);
[uout,tout]=euler_fwd(func,ti,tf,k,u0);
uout
figure;
plot(tout,uout);

k=0.15;
[uout,tout]=euler_back(func,ti,tf,k,u0,tol,nmax);
uout
figure;
plot(tout,uout);
[uout,tout]=euler_fwd(func,ti,tf,k,u0);
uout
figure;
plot(tout,uout);

k=0.1;
[uout,tout]=euler_back(func,ti,tf,k,u0,tol,nmax);
uout
figure;
plot(tout,uout);
[uout,tout]=euler_fwd(func,ti,tf,k,u0);
uout
figure;
plot(tout,uout);

k=0.05;
[uout,tout]=euler_back(func,ti,tf,k,u0,tol,nmax);
uout
figure;
plot(tout,uout);
[uout,tout]=euler_fwd(func,ti,tf,k,u0);
uout
figure;
plot(tout,uout);
```

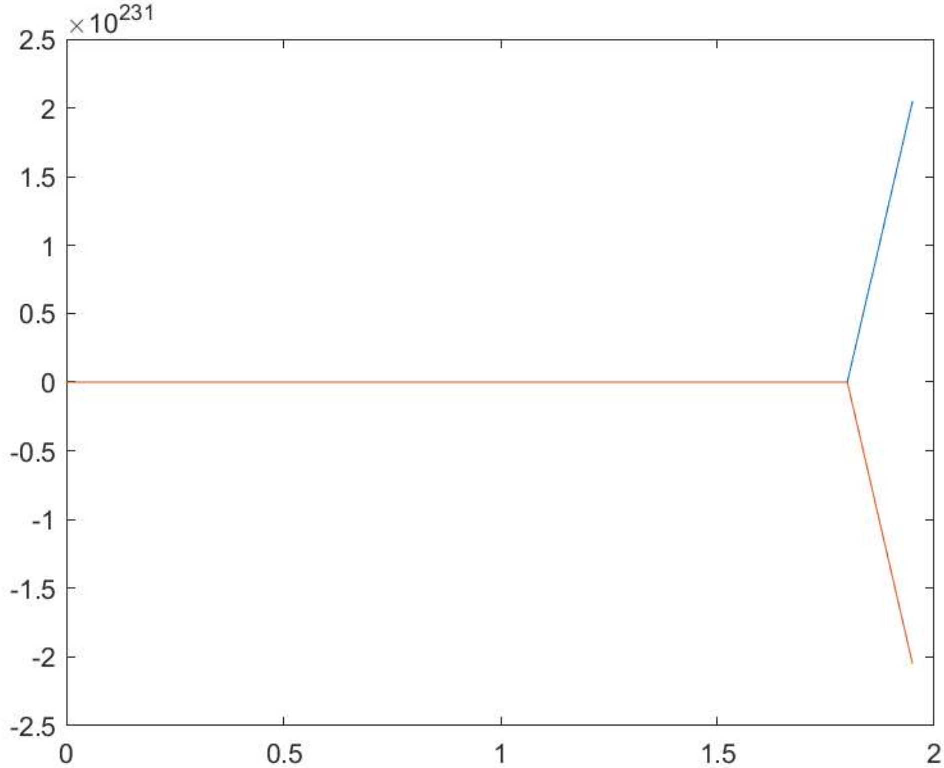
```
function [uout,tout]=euler_back(func,ti,tf,k,u0,tol,nmax)
% Determine the number of steps:
nt_steps = ( tf - ti ) / k;

% Executable statements:
uout(1,:) = u0;
tout(:,1) = ti;
        t = ti;

% For-loop over the number of time-steps:
for m = 0:nt_steps-1

    n=0;
    uprev=u0;
    e=[1;1];
    while(abs(e)>tol)
        unext=u0+k*func(t+k,uprev);
        e=unext-uprev;
        uprev=unext;
        if(n>nmax)
            break;
        end
        n=n+1;
    end
    u0=unext;

    t = ti + ( m + 1 ) * k;

     % Store things:
     uout(m+2,:) = u0';
     tout(m+2) = t;

end

end
```



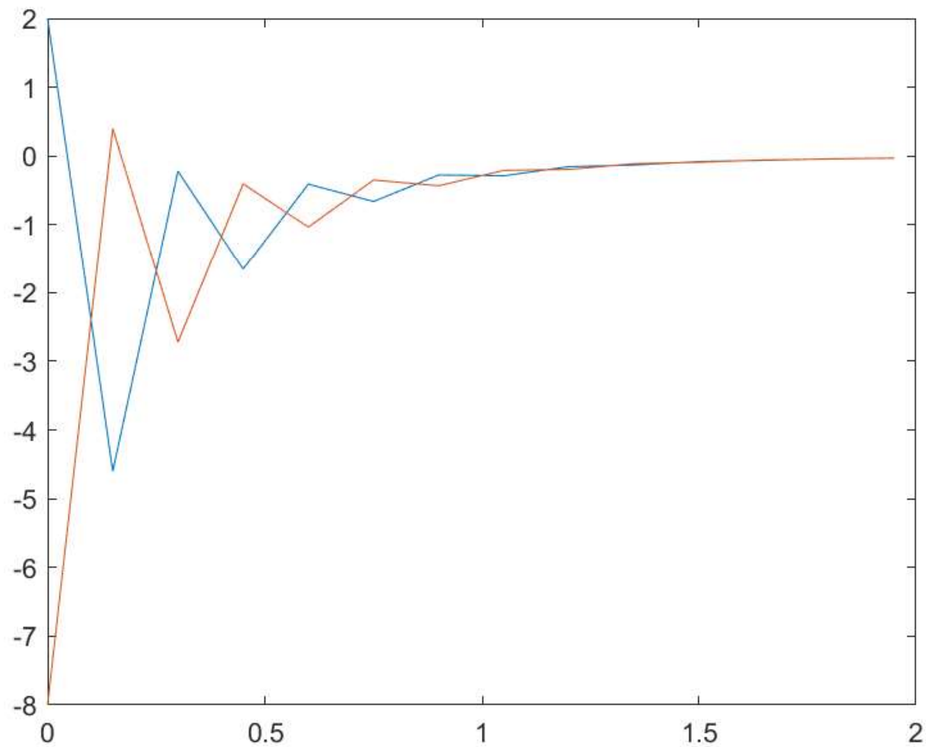
$\times 10^{306}$
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.5
1
1.5
2

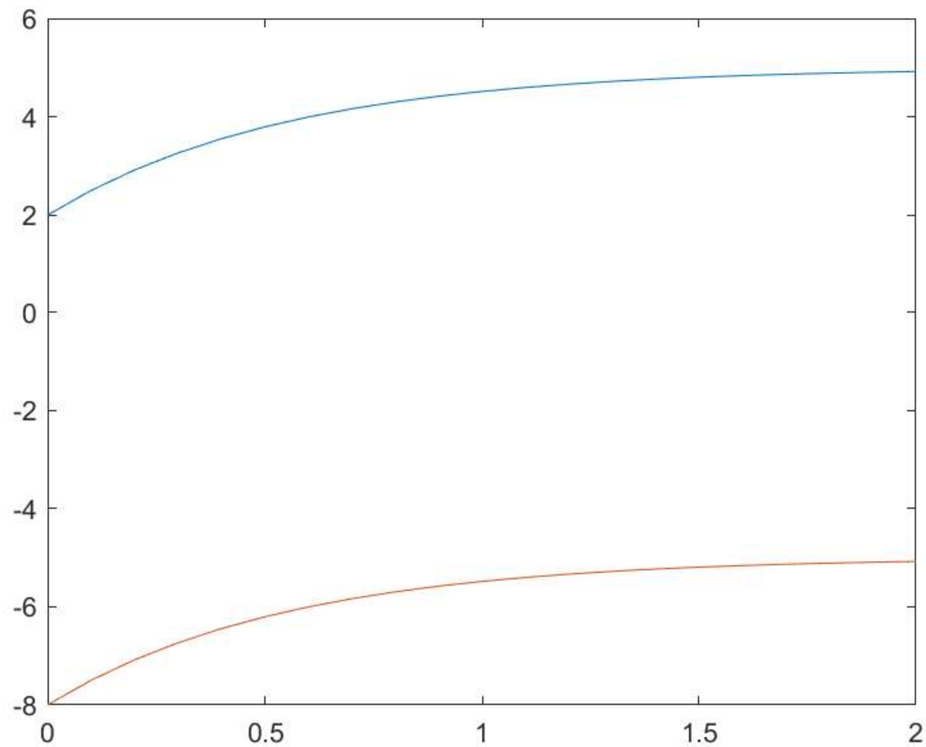


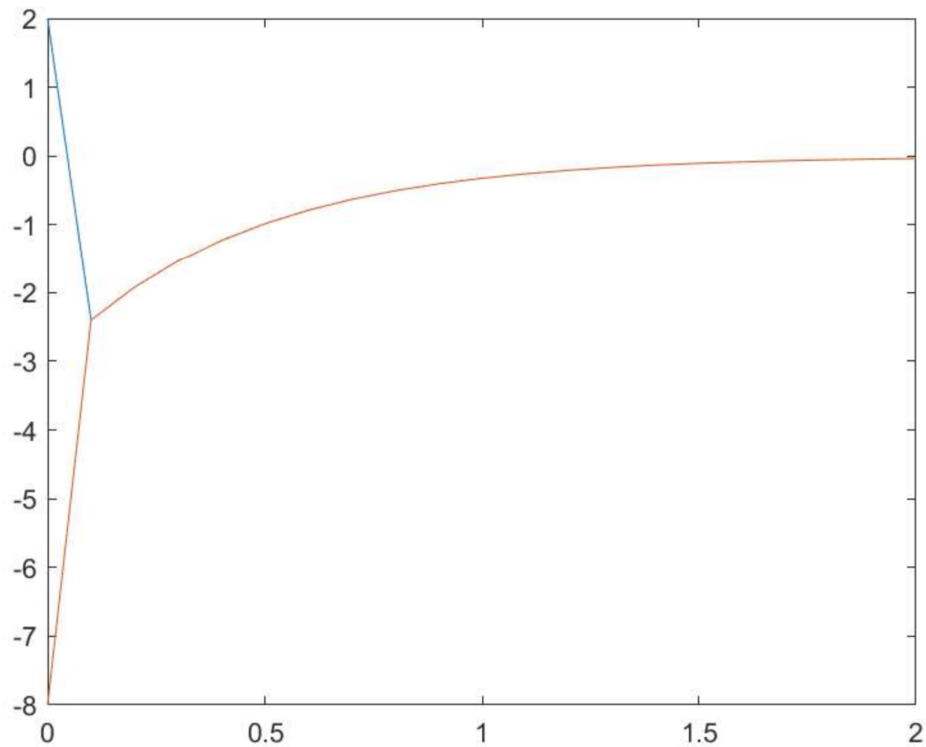
6
4
2
0
-2
-4
-6
-8
0
0.5
1
1.5
2

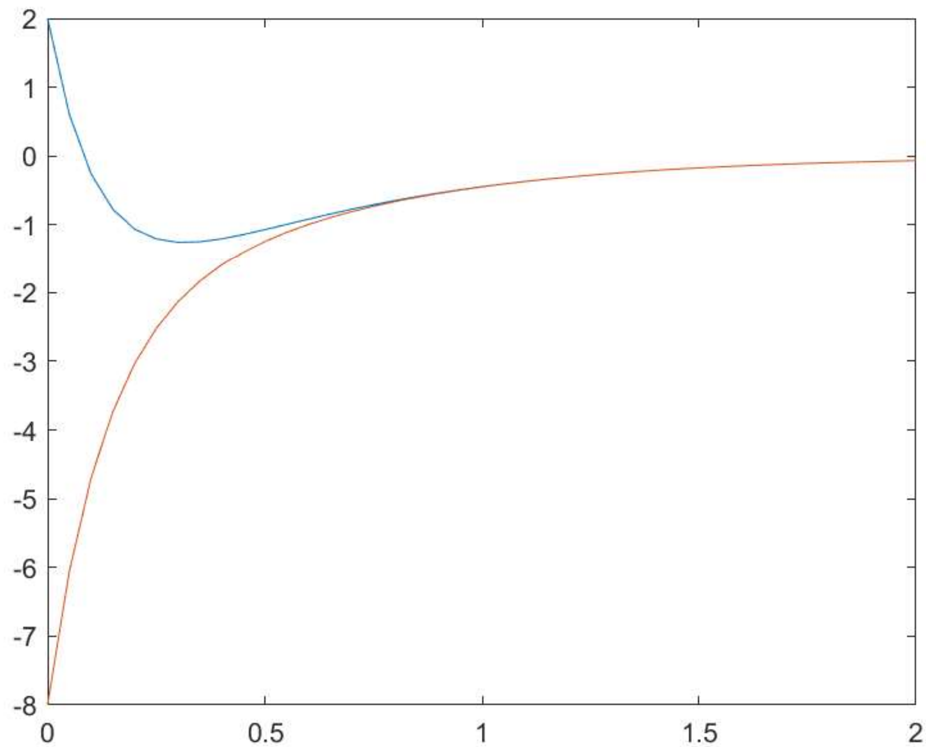


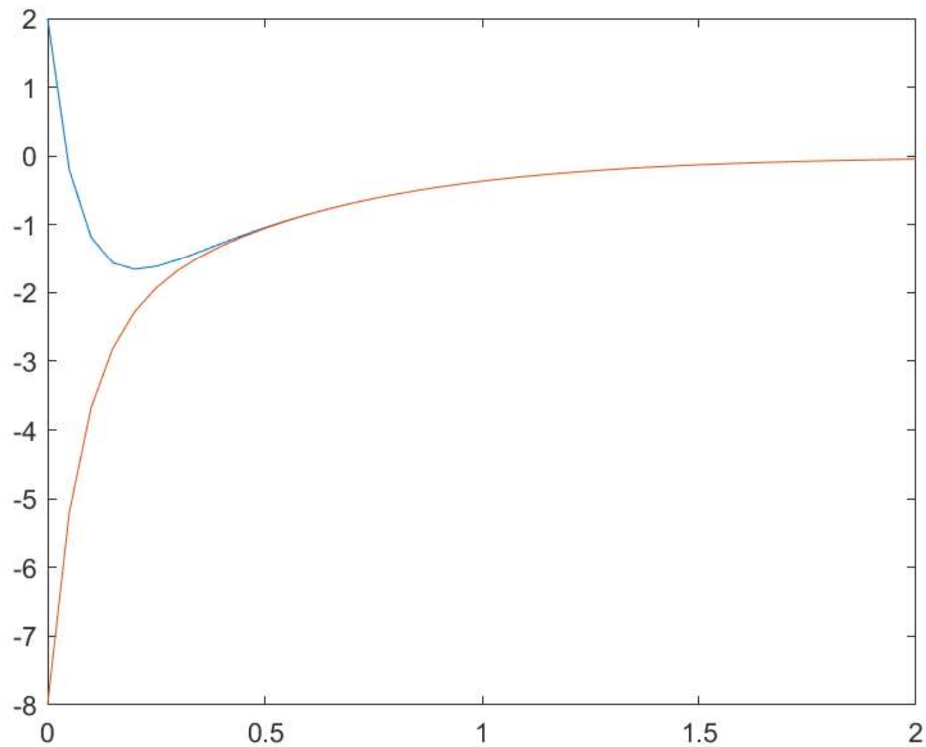
×10^231
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
0.5
1
1.5
2


2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
0
0.5
1
1.5
2

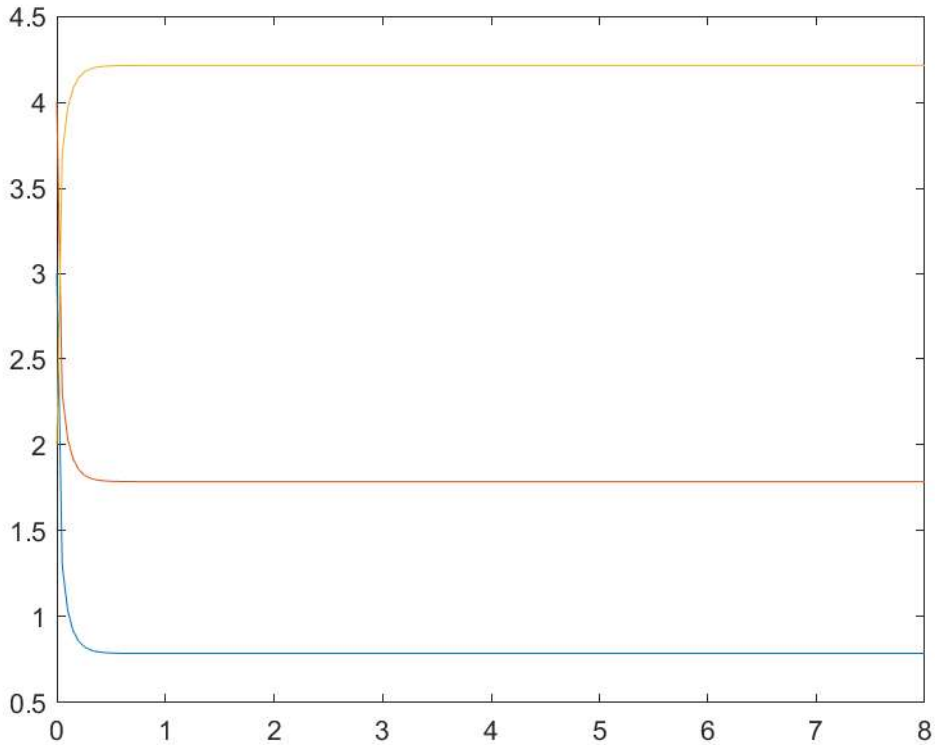

6
4
2
0
-2
-4
-6
-8
0
0.5
1
1.5
2


2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
0
0.5
1
1.5
2

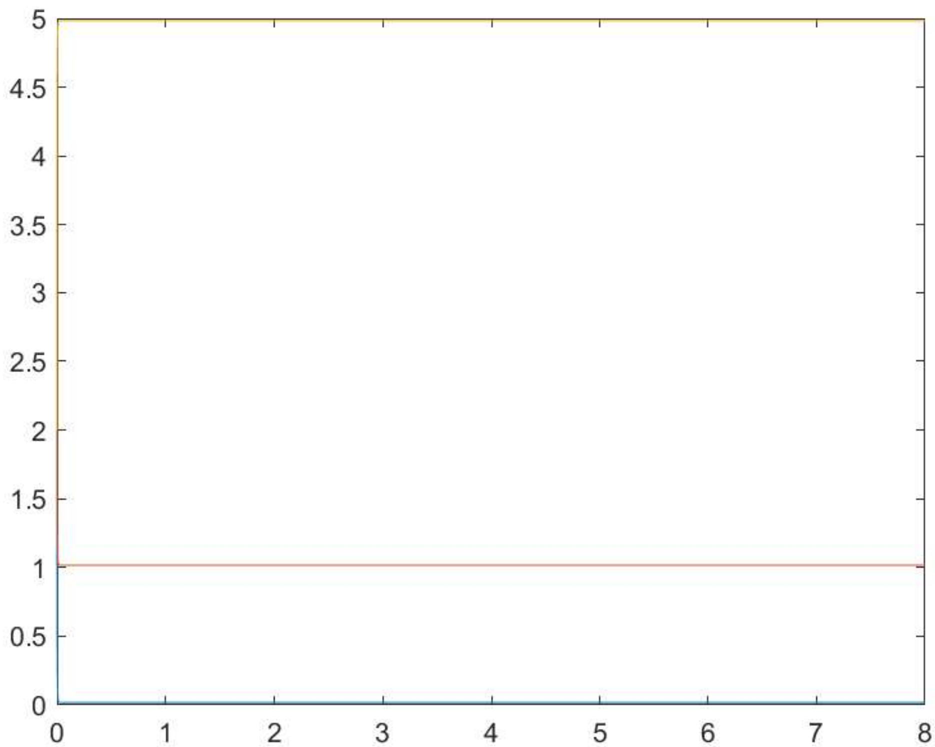

2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
0
0.5
1
1.5
2

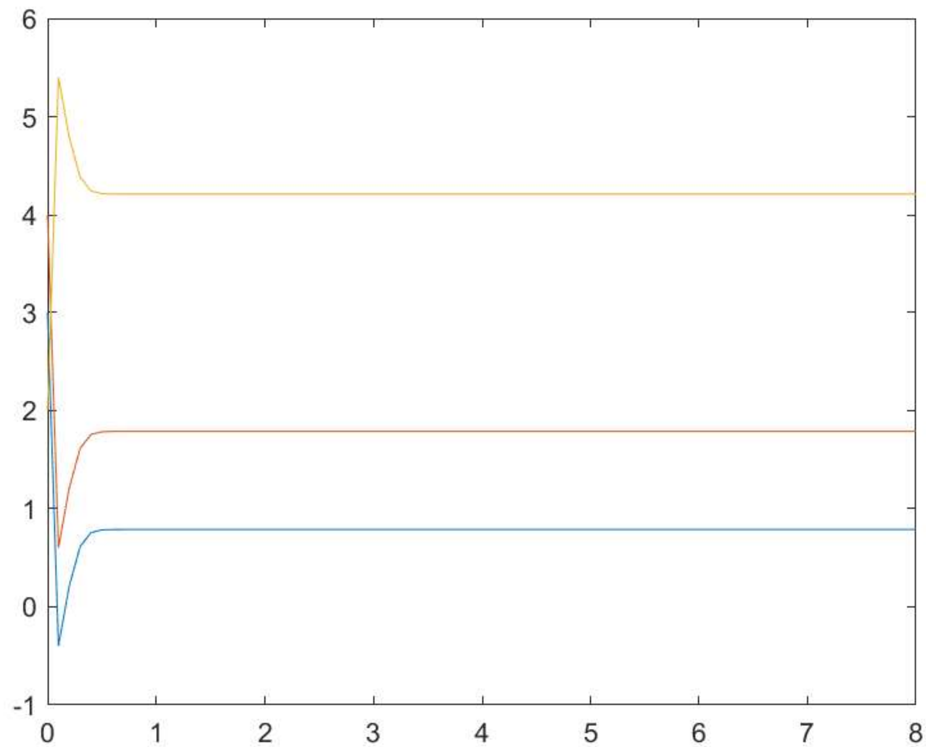

2
1
0
-1
-2
-3
-4
-5
-6
-7
-8
0
0.5
1
1.5
2

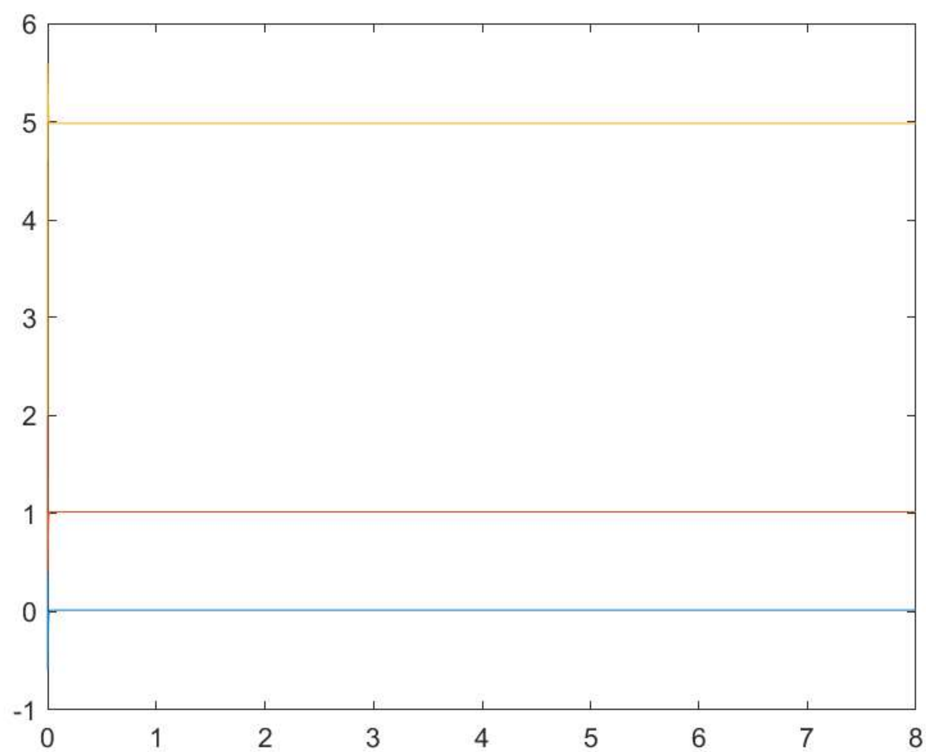
```
format long;

u0=[3;4;2];
ti=0;
tf=8;

func=@(t,u)[-3*u(1)*u(2)+1*u(3);-3*u(1)*u(2)+1*u(3);3*u(1)*u(2)-1*u(3)];
k=0.1;
[uout,tout]=euler_fwd(func,ti,tf,k,u0);
figure;
plot(tout,uout);

func=@(t,u)[-300*u(1)*u(2)+1*u(3);-300*u(1)*u(2)+1*u(3);300*u(1)*u(2)-1*u(3)];
k=1e-3;
[uout,tout]=euler_fwd(func,ti,tf,k,u0);
figure;
plot(tout,uout);
```



4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7
8


5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0
1
2
3
4
5
6
7
8


6
5
4
3
2
1
0
-1
0
1
2
3
4
5
6
7
8



6
5
4
3
2
1
0
-1
0
1
2
3
4
5
6
7
8