AMATH 586 SPRING 2020
HOMEWORK 1 — DUE APRIL 10 ON GITHUB BY 11PM

Be sure to do a `git pull` to update your local version of the `amath-586-2020` repository.
Shannon Dow

Problem 1: Using the Taylor series representation of the matrix exponential:

(a) Verify the identities

$$\frac{\mathrm{d}}{\mathrm{d}\,t}\,\mathrm{e}^{tA} = A\,\mathrm{e}^{tA} = \mathrm{e}^{tA}\,A$$

for an $n \times n$ matrix A.

(b) Verify that $u(t) = \mathrm{e}^{tA}\,\eta$ is indeed the solution of the IVP

$$\begin{cases} u'(t) = Au(t), \\ u(0) = \eta. \end{cases}$$

Solution:

(a) The matrix exponential is:

$$e^{tA} = 1 + tA + \frac{1}{2}t^2A^2 + \frac{1}{6}t^3A^3 + \ldots$$

$$\frac{d}{dt}[e^{tA}] = \frac{d}{dt}[1+tA+\frac{1}{2}t^2A^2+\frac{1}{6}t^3A^3+\ldots] = A+tA^2+\frac{1}{2}t^2A^3+\ldots = A(1+tA+\frac{1}{2}t^2A^2+\ldots) = Ae^{tA}$$

$$\frac{d}{dt}[e^{tA}] = \frac{d}{dt}[1+tA+\frac{1}{2}t^2A^2+\frac{1}{6}t^3A^3+\ldots] = A+tA^2+\frac{1}{2}t^2A^3+\ldots = (1+tA+\frac{1}{2}t^2A^2+\ldots)A = e^{tA}A$$

(b)

$$u'(t) = \frac{d}{dt}[e^{tA}\eta] = Ae^{tA}\eta = Au(t)$$

Problem 2: Construct a system (i.e., needs to be not scalar valued)

$$\begin{cases} u'(t) = f(u(t)), \end{cases}$$

and two choices of initial data $u_0 \neq v_0$ so that two solutions

$$\begin{cases} u'(t) = f(u(t)), \\ u(0) = u_0, \end{cases} \quad \begin{cases} v'(t) = f(v(t)), \\ v(0) = v_0, \end{cases}$$

satisfy

$$\|u(t) - v(t)\|_2 = \|u(0) - v(0)\|_2\,\mathrm{e}^{Lt} \tag{1}$$

where L a Lipschitz constant for $f(u)$. Recall that we have shown that for any solution

$$\|u(t) - v(t)\|_2 \leq \|u(0) - v(0)\|_2 \, \mathrm{e}^{Lt}.$$

So, you are tasked with showing that this is sharp. Then show that equality (1) fails to hold for $u'(t) = -f(u(t)), v'(t) = -f(v(t))$ with the same initial conditions.

Solution

$$\begin{cases} u'(t) = Lu(t) \\ v'(t) = Lv(t) \end{cases}$$

With initial data: $u(0) = u_0, v(0) = v_0$ where $u_0 \neq v_0$ Solving both of these, we get that:

$$\begin{cases} u(t) = u_0 e^{Lt} \\ v(t) = v_0 e^{Lt} \end{cases}$$

Thus:

$$\|u(t) - v(t)\|_2 = \|u_0 e^{Lt} - v_0 e^{Lt}\|_2 = \|(u_0 - v_0)e^{Lt}\|_2 = \|u_0 - v_0\|_2 e^{Lt} = \|u(0) - v(0)\|_2 e^{Lt}$$

Now, lets consider the system:

$$\begin{cases} u'(t) = -Lu(t) \\ v'(t) = -Lv(t) \end{cases}$$

with the same initial data. The solution becomes:

$$\begin{cases} u(t) = u_0 e^{-Lt} \\ v(t) = v_0 e^{-Lt} \end{cases}$$

$$\|u(t) - v(t)\|_2 = \|u_0 e^{-Lt} - v_0 e^{-Lt}\|_2 = \|(u_0 - v_0)e^{-Lt}\|_2 = \|u_0 - v_0\|_2 e^{-Lt} \neq \|u(0) - v(0)\|_2 e^{Lt}$$

Problem 3: Consider the IVP

$$\begin{cases} u_1'(t) = 2u_1(t), \\ u_2'(t) = 3u_1(t) - u_2(t), \end{cases}$$

with initial conditions specified at time $t = 0$. Solve this problem in two different ways:

(a) Solve the first equation, which only involves u_1, and then insert this function into the second equation to obtain a nonhomogeneous linear equation for u_2. Solve this using (5.8). Check that your solution satisfies the initial conditions and the ODE.

(b) Write the system as $u' = Au$ and compute the matrix exponential using (D.30) to obtain the solution.

Solution:

(a) Let the initial conditions be: $u_1(0) = \alpha$ and $u_2(0) = \beta$

$$\int \frac{u_1'(t)}{u_1(t)} dt = \int 2dt$$

$$\ln[u_1(t)] = 2t + C$$

$$u_1(t) = Ce^{2t}$$

$$u_1(0) = Ce^0 = \alpha \implies C = \alpha$$
$$u_1(t) = \alpha e^{2t}$$
Now, substitute into the second one:
$$u_2'(t) = -u_2(t) + 3\alpha e^{2t}$$
$$u_2(0) = \beta$$
Using Duhamel's Formula where $A = -1, f(t) = 3\alpha e^{2t}, t_0 = 0, \eta = \beta$
$$u_2(t) = e^{-t}u_2(0) + \int_0^t e^{s-t}3\alpha e^{2s}ds$$
$$u_2(t) = e^{-t}\beta + 3\alpha \int_0^t e^{3s-t}ds$$
$$u_2(t) = e^{-t}\beta + \alpha(e^{3s-t})|_0^t ds$$
$$u_2(t) = e^{-t}\beta + \alpha(e^{2t} - e^{-t})$$
Verify that the solution satisfies the differential equation:
$$u_1(t) = \alpha e^{2t}$$
$$u_1'(t) = 2\alpha e^{2t} = 2u_1(t)$$

$$u_2(t) = e^{-t}\beta + \alpha(e^{2t} - e^{-t})$$
$$u_2'(t) = -\beta e^{-t} + \alpha(2e^{2t} + e^{-t})$$
$$3u_1(t) - u_2(t) = 3\alpha e^{2t} - e^{-t}\beta - \alpha(e^{2t} - e^{-t})$$
$$3u_1(t) - u_2(t) = -\beta e^{-t} + 2\alpha e^{2t} + \alpha e^{-t} = u_2'(t)$$
Verify that both solutions satisfy the initial conditions:
$$u_1(0) = \alpha e^0 = \alpha$$
$$u_2(0) = e^0\beta + \alpha(e^0 - e^0) = \beta$$
(b)
$$u'(t) = Au(t), \text{where } u'(t) = \begin{bmatrix} u_1'(t) \\ u_2'(t) \end{bmatrix}, u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}, \text{ and } A = \begin{bmatrix} 2 & 0 \\ 3 & -1 \end{bmatrix}$$
Additionally, the initial conditions can be defined as $u(o) = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$
Since A is diagonalizable, we can rewrite $A = R\Lambda R^{-1}$ or $\Lambda = R^{-1}AR$
$$u'(t) = R\Lambda R^{-1}u(t)$$
$$R^{-1}u'(t) = R^{-1}R\Lambda R^{-1}u(t)$$
Let $v(t) = R^{-1}u(t)$
$$v'(t) = \Lambda v(t)$$
Solving this, you get,
$$v(t) = e^{\Lambda t}v(0)$$
Replacing v's with u's, we get:
$$u(t) = Re^{\Lambda t}R^{-1}u(0)$$
Diagonalizing A, we get:
$$R = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} R^{-1} = \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix}, \Lambda = \begin{bmatrix} -1 & 0 \\ 0 & 2 \end{bmatrix}$$

$$e^{\Lambda t} = \begin{bmatrix} e^{-t} & 0 \\ 0 & e^{2t} \end{bmatrix}$$

Thus,

$$u(t) = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} e^{-t} & 0 \\ 0 & e^{2t} \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

Multiplying this out, we get:

$$u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} = \begin{bmatrix} \alpha e^{2t} \\ \alpha(e^{2t} - e^{-t}) + \beta e^{-t} \end{bmatrix}$$

Problem 4: Consider the IVP

$$\begin{cases} u_1'(t) = 2u_1(t), \\ u_2'(t) = 3u_1 + 2u_2(t), \end{cases}$$

with initial conditions specified at time $t = 0$. Solve this problem. **Solution** Using the same initial conditions as problem 3. Similar to problem 3 above:

$$u_1(t) = \alpha e^{2t}$$

Substituting into $u_2(t)$ yields:

$$u_2'(t) = 2u_2(t) + 3\alpha e^{2t}$$

Using Duhamels formula:

$$u_2(t) = \beta e^{2t} + 3\alpha \int_0^t e^{2t-2s} e^{2s} ds$$

$$u_2(t) = \beta e^{2t} + 3\alpha \int_0^t e^{2t} ds$$

$$u_2(t) = \beta e^{2t} + 3\alpha t e^{2t}$$

$$u_2(t) = (\beta + 3\alpha t)e^{2t}$$

Problem 5: Consider the Lotka–Volterra system[1]

$$\begin{cases} u_1'(t) = \alpha u_1(t) - \beta u_1(t)u_2(t), \\ u_2'(t) = \delta u_1(t)u_2(t) - \gamma u_2(t). \end{cases}$$

For $\alpha = \delta = \gamma = \beta = 1$ and $u_1(0) = 5, u_2(0) = 0.8$ use the forward Euler method to approximate the solution with $k = 0.001$ for $t = 0, 0.001, \ldots, 50$. Plot your approximate solution as a curve in the (u_1, u_2)-plane and plot your approximations of $u_1(t)$ and $u_2(t)$ on the same axes as a function of t. Repeat this with backward Euler. What do you notice about the behavior of the numerical solutions? The most obvious feature is most apparent in the (u_1, u_2)-plane.

Problem 6: Determine the coefficients $\beta_0,\ \beta_1,\ \beta_2$ for the third order, 2-step Adams-Moulton method. Do this in two different ways:

[1]This is a famous model of predator-prey dynamics.

(a) Using the expression for the local truncation error in Section 5.9.1,

(b) Using the relation

$$u(t_{n+2}) = u(t_{n+1}) + \int_{t_{n+1}}^{t_{n+2}} f(u(s))\, ds.$$

Interpolate a quadratic polynomial $p(t)$ through the three values $f(U^n),\ f(U^{n+1})$ and $f(U^{n+2})$ and then integrate this polynomial exactly to obtain the formula. The coefficients of the polynomial will depend on the three values $f(U^{n+j})$. It's easiest to use the "Newton form" of the interpolating polynomial and consider the three times $t_n = -k$, $t_{n+1} = 0$, and $t_{n+2} = k$ so that $p(t)$ has the form

$$p(t) = A + B(t+k) + C(t+k)t$$

where $A,\ B$, and C are the appropriate divided differences based on the data. Then integrate from 0 to k. (The method has the same coefficients at any time, so this is valid.)

Solution

(a) The LTE formula from 5.9.1 is:

$$\tau(t_{n+r}) = \frac{1}{k}\Big(\sum_{j=0}^{2} \alpha_j u(t_{n+j}) - k\sum_{j=0}^{2} \beta_j u'(t_{n+j})\Big)$$

Recall that to be consistent, $\sum_{j=0}^{2} \alpha_j = 0$, so we can rewrite as:

$$\frac{u(t+2k) - u(t+k)}{k} = \sum_{j=0}^{2} \beta_j u'(t_{n+j})$$

$$\frac{u(t+2k) - u(t+k)}{k} = \beta_0 u'(t) + \beta_1 u'(t+k) + \beta_2 u'(t+2k)$$

Expanding the left hand side using Taylor Series:

$$\frac{u(t+2k)}{k} = \frac{1}{k}[u(t) + 2ku'(t) + 2k^2u''(t) + \frac{4}{3}k^3u'''(t) + O(k^4)]$$

$$-\frac{u(t+k)}{k} = \frac{-1}{k}[u(t) + ku'(t) + \frac{1}{2}k^2u''(t) + \frac{1}{6}k^3u'''(t) + O(k^4)]$$

Expanding the right hand side using Taylor Series:

$$\beta_0 u'(t) = \beta_0 u'(t)$$

$$\beta_1 u'(t+k) = \beta_1[u'(t) + ku''(t) + \frac{1}{2}k^2u'''(t) + O(k^3)]$$

$$\beta_2 u'(t+2k) = \beta_2[u'(t) + 2ku''(t) + 2k^2u'''(t) + O(k^3)]$$

Now, we have a system of 3 equations and 3 unknowns:

$$\beta_0 + \beta_1 + \beta_2 = 1$$

$$\frac{3}{2} = \beta_1 + 2\beta_2$$

$$\frac{7}{6} = \frac{1}{2}\beta_1 + 2\beta_2$$

$$\frac{7}{6} = \frac{1}{2}(\frac{3}{2} - 2\beta_2) + 2\beta_2$$

$$\beta_2 = \frac{5}{12}$$

$$\beta_1 = \frac{8}{12}$$
$$\beta_0 = -\frac{1}{12}$$

With these three coefficients, we get the 3rd order 2-step Adams-Moulton Method:

$$U^{n+2} = U^{n+1} + \frac{k}{12}(-f(U^n) + 8f(U^{n+1} + 5f(U^{n+2}))$$

(b) Using the Newtonian interpolating polynomial through the following points:

- $x_0 = t_n = -k, y_0 = f(U^n)$
- $x_1 = t_{n+1} = 0, y_1 = f(U^{n+1})$
- $x_2 = t_{n+2} = k, y_2 = f(U^{n+2})$

$$A = y_0 = f(U^n)$$
$$B = \frac{y_1 - y_0}{x_1 - x_0} = \frac{f(U^{n+1}) - f(U^n)}{k}$$
$$C = \frac{\frac{y_2-y_0}{x_2-x_0} - \frac{y_1-y_0}{x_1-x_0}}{x_2 - x_1} = \frac{f(U^{n+2}) - 2f(U^{n+1}) + f(U^n)}{2k^2}$$
$$\int_0^k f(U^n) + \frac{f(U^{n+1}) - f(U^n)}{k}(t+k) + \frac{f(U^{n+2}) - 2f(U^{n+1}) + f(U^n)}{2k^2}(t^2 + tk)dt$$
$$f(U^n)t + \frac{f(U^{n+1}) - f(U^n)}{k}(\frac{1}{2}t^2 + kt) + \frac{f(U^{n+2}) - 2f(U^{n+1}) + f(U^n)}{2k^2}(\frac{1}{3}t^3 + \frac{1}{2}t^2k)|_0^k$$
$$f(U^n)k + \frac{3k}{2}(f(U^{n+1}) - f(U^n)) + \frac{5k}{12}(f(U^{n+2}) - 2f(U^{n+1}) + f(U^n))$$

Then combining like terms and substituting into the relation:

$$U^{n+2} = U^{n+1} + \frac{k}{12}(-f(U^n) + 8f(U^{n+1} + 5f(U^{n+2}))$$

Homework One

Shannon Dow

Problem 5:

Consider the Lotka--Volterra system:

$$\begin{cases} u_1'(t) = \alpha u_1(t) - \beta u_1(t)u_2(t), \\ u_2'(t) = \delta u_1(t)u_2(t) - \gamma u_2(t). \end{cases}$$

For $\alpha = \delta = \gamma = \beta = 1$ and $u_1(0) = 5, u_2(0) = 0.8$ use the forward Euler method to approximate the solution with $k = 0.001$ for $t = 0, 0.001, \dots, 50$. Plot your approximate solution as a curve in the (u_1, u_2)-plane and plot your approximations of $u_1(t)$ and $u_2(t)$ on the same axes as a function of t. Repeat this with backward Euler. What do you notice about the behavior of the numerical solutions? The most obvious feature is most apparent in the (u_1, u_2)-plane.

In [1]:

```
#Assign Values to Parameters:
α = 1
δ = 1
γ = 1
β = 1
```

Out[1]:

```
1
```

In [2]:

```
#Define f(u)
f = u -> [α*u[1]-β*u[1]*u[2], δ*u[1]*u[2]-γ*u[2]]
```

Out[2]:

```
#3 (generic function with 1 method)
```

In [3]:

```
#Stepsize
k = 0.001
#Max Time
T = 50
```

Out[3]:

```
50
```

Forward Euler

In [4]:

```
# Forward Euler
n = convert(Int64,T/k)# Number of time steps, converted to Int64
U = zeros(2,n+1) # To save the solution values
U[:,1] = [5,0.8]
for i = 2:n+1
    U[:,i] = U[:,i-1] + k*f(U[:,i-1])
end
```

In [5]:

```
#Print U
U
```

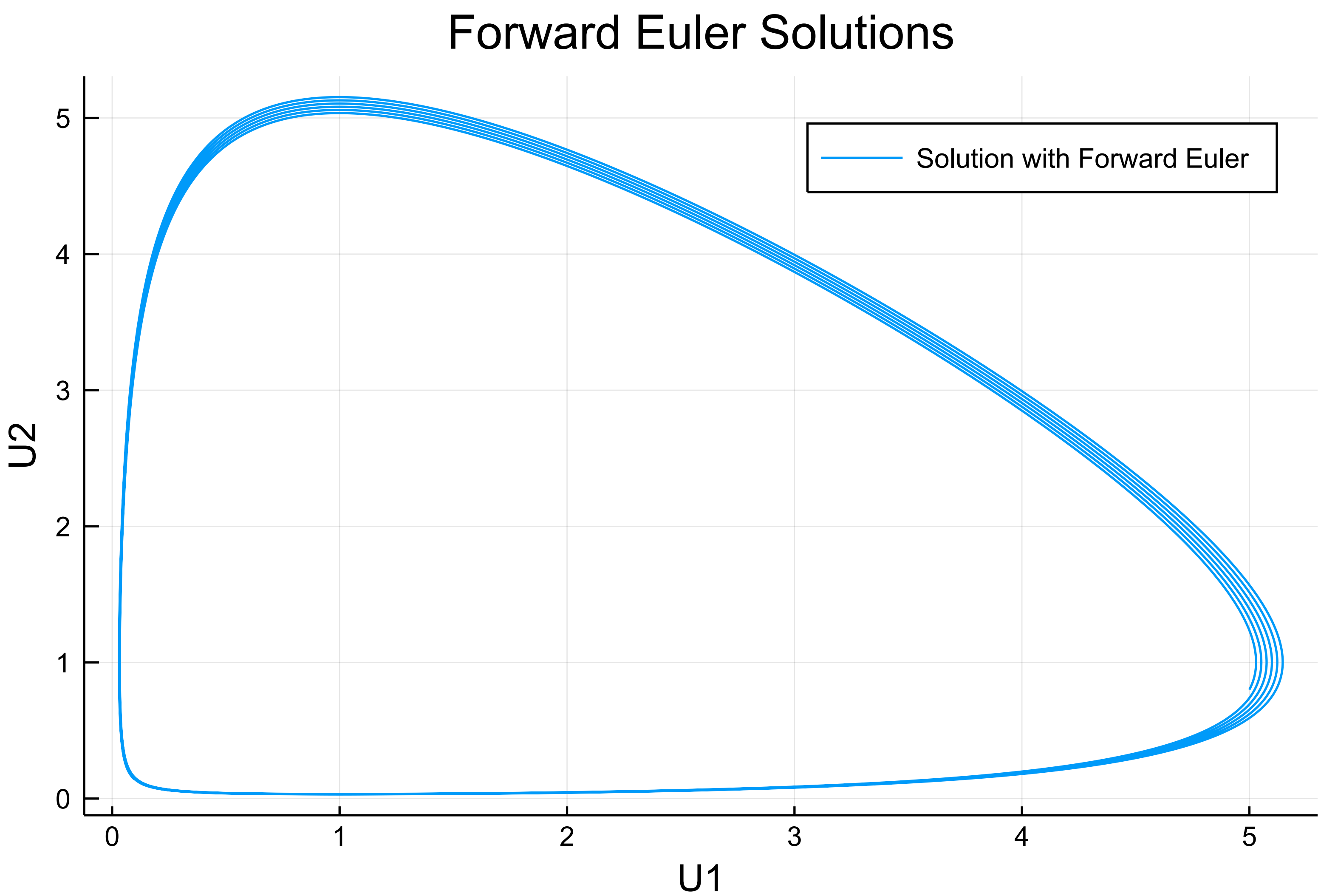
Out[5]:

```
2×50001 Array{Float64,2}:
 5.0  5.001   5.00198   5.00295   …  0.0967196  0.0968025  0.0968854
 0.8  0.8032  0.806414  0.809641     0.14305    0.142921   0.142792
```

In [6]:

```
using Plots
plot(U[1,:],U[2,:],label="Solution with Forward Euler",title = "Forward Euler Sc
xlabel!("U1")
ylabel!("U2")
```

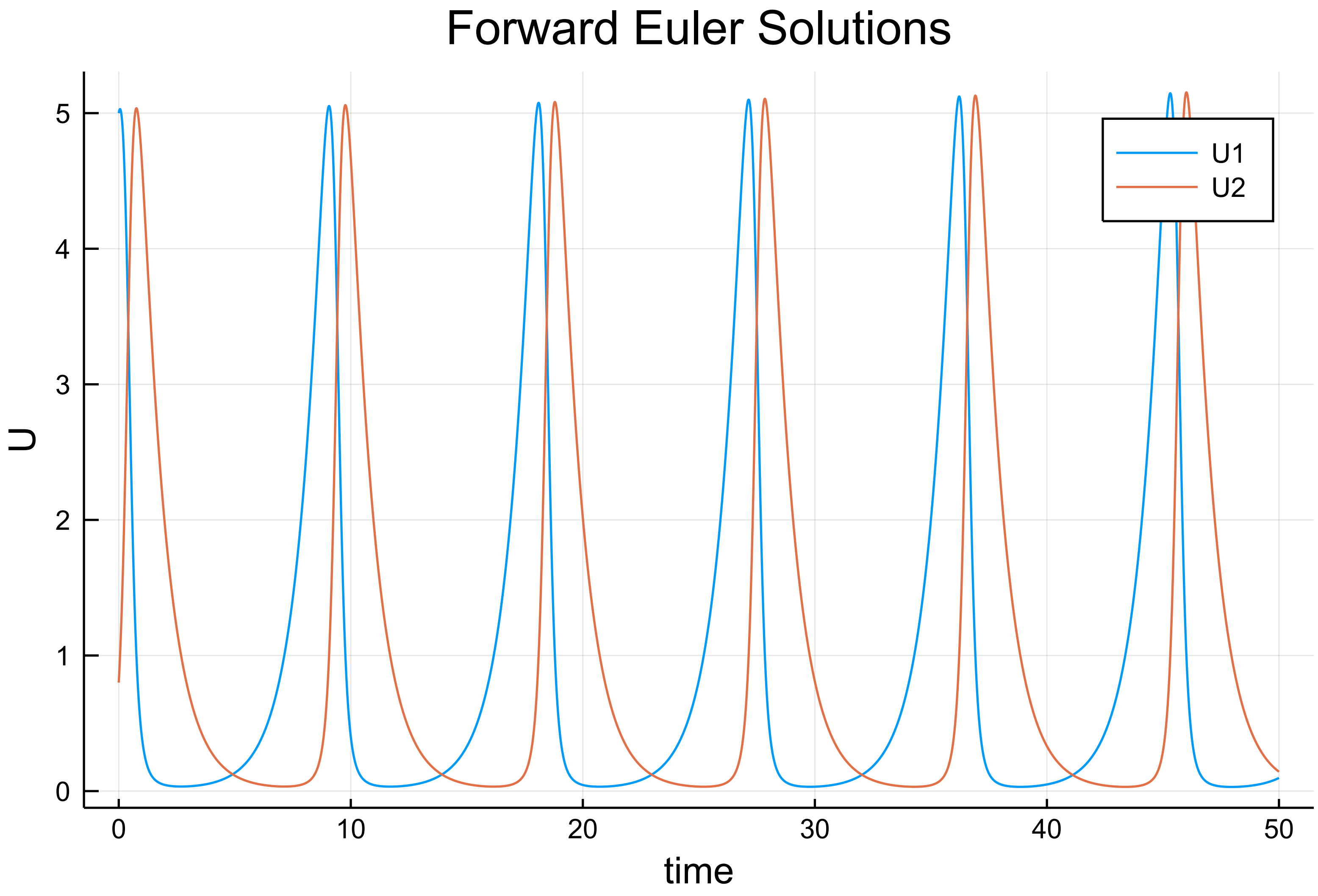
Out[6]:



In [7]:

```
t= range(0,stop = T,step = k)
plot(t,U[1,:],label ="U1",title = "Forward Euler Solutions")
plot!(t,U[2,:], label = "U2")
xlabel!("time")
ylabel!("U")
```

Out[7]:



Backward Euler

In [8]:

```
g = (U,Un) -> U - Un - k*f(U)
Dg = (U) -> [1-k*α+k*β*U[2] k*β*U[1];
     -k*δ*U[2] 1-k*δ*U[1]+k*γ]
```

Out[8]:

```
#7 (generic function with 1 method)
```

In [9]:

```
# Backward Euler
n = convert(Int64,T/k) # Number of time steps, converted to Int64
Ub = zeros(2,n+1) # To save the solution values
Ub[:,1] = [5,0.8]
max_iter = 10
for i = 2:n+1
    Unew = Ub[:,i-1]
    Uold = Ub[:,i-1]
    for j = 1:max_iter
        Uold = Unew
        Unew = Uold - (Dg(Uold)\g(Uold,Ub[:,i-1]))
        #println(maximum(abs.(Unew-Uold)))
        if maximum(abs.(Unew-Uold)) < k/10 # Newton's method until error tol.
            break
        end
        if j == max_iter
            println("Newton didn't terminate")
        end
    end
    Ub[:,i] = Unew
end
```

In [10]:

```
Ub
```

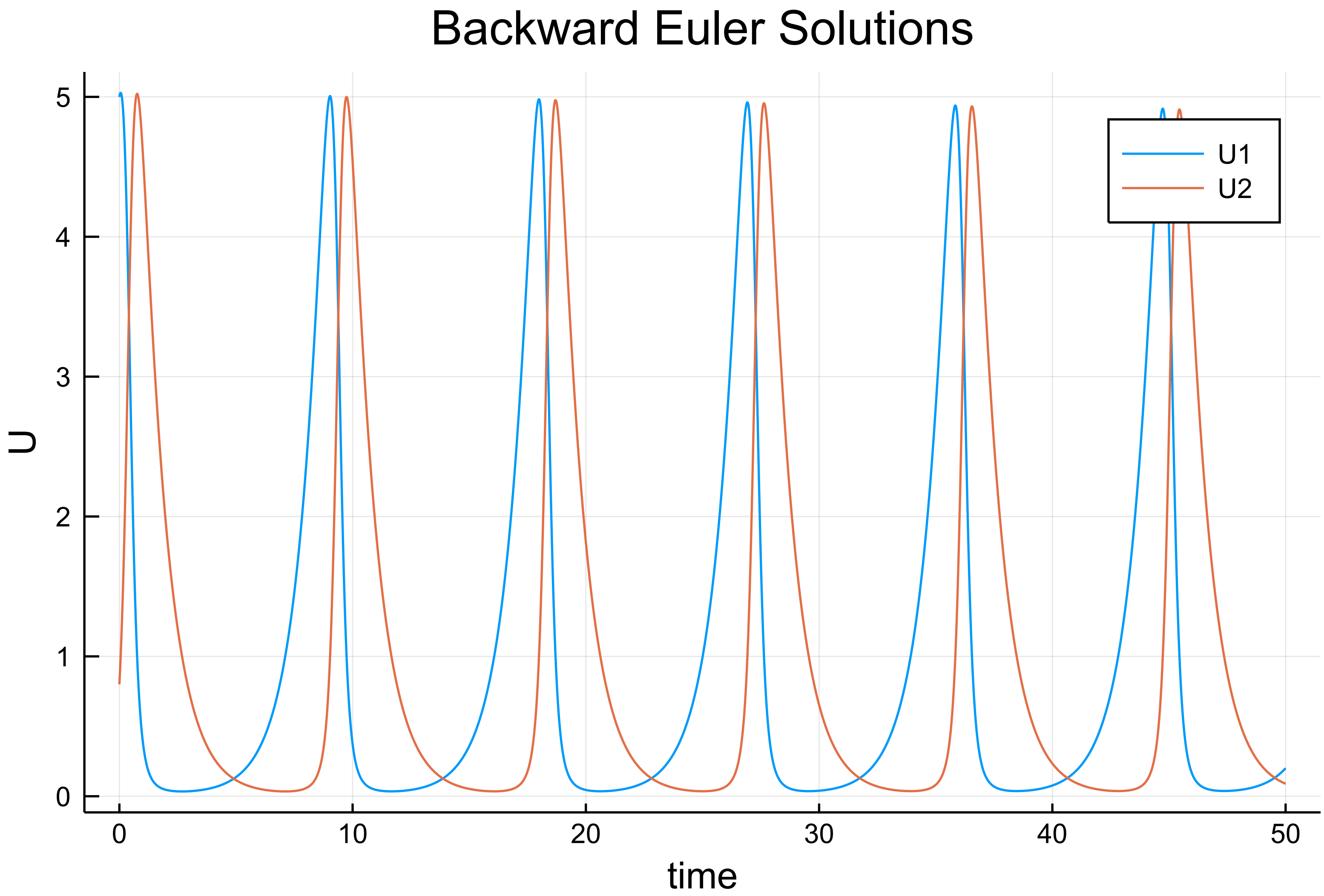
Out[10]:

```
2×50001 Array{Float64,2}:
 5.0  5.00098   5.00195   5.0029    …  0.200303   0.200485   0.200668
 0.8  0.803214  0.806441  0.809682     0.0894329  0.0893614  0.08929
```

In [11]:

```
t= range(0,stop = T,step = k)
plot(t,Ub[1,:],label ="U1",title = "Backward Euler Solutions")
plot!(t,Ub[2,:], label = "U2")
xlabel!("time")
ylabel!("U")
```

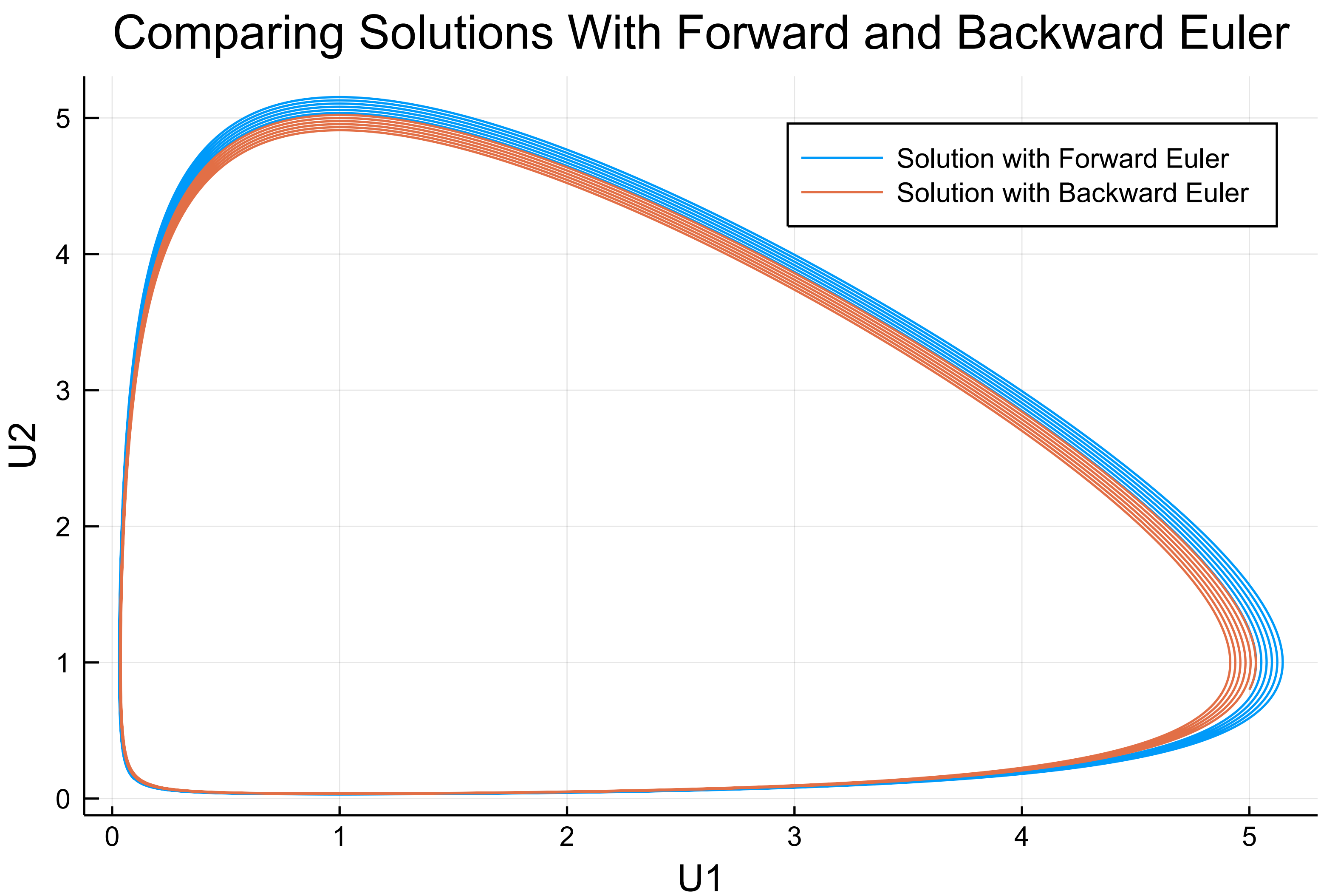
Out[11]:



In [14]:

```
plot(U[1,:],U[2,:],label="Solution with Forward Euler",title="Comparing Solutio
plot!(Ub[1,:],Ub[2,:],label="Solution with Backward Euler")
xlabel!("U1")
ylabel!("U2")
```

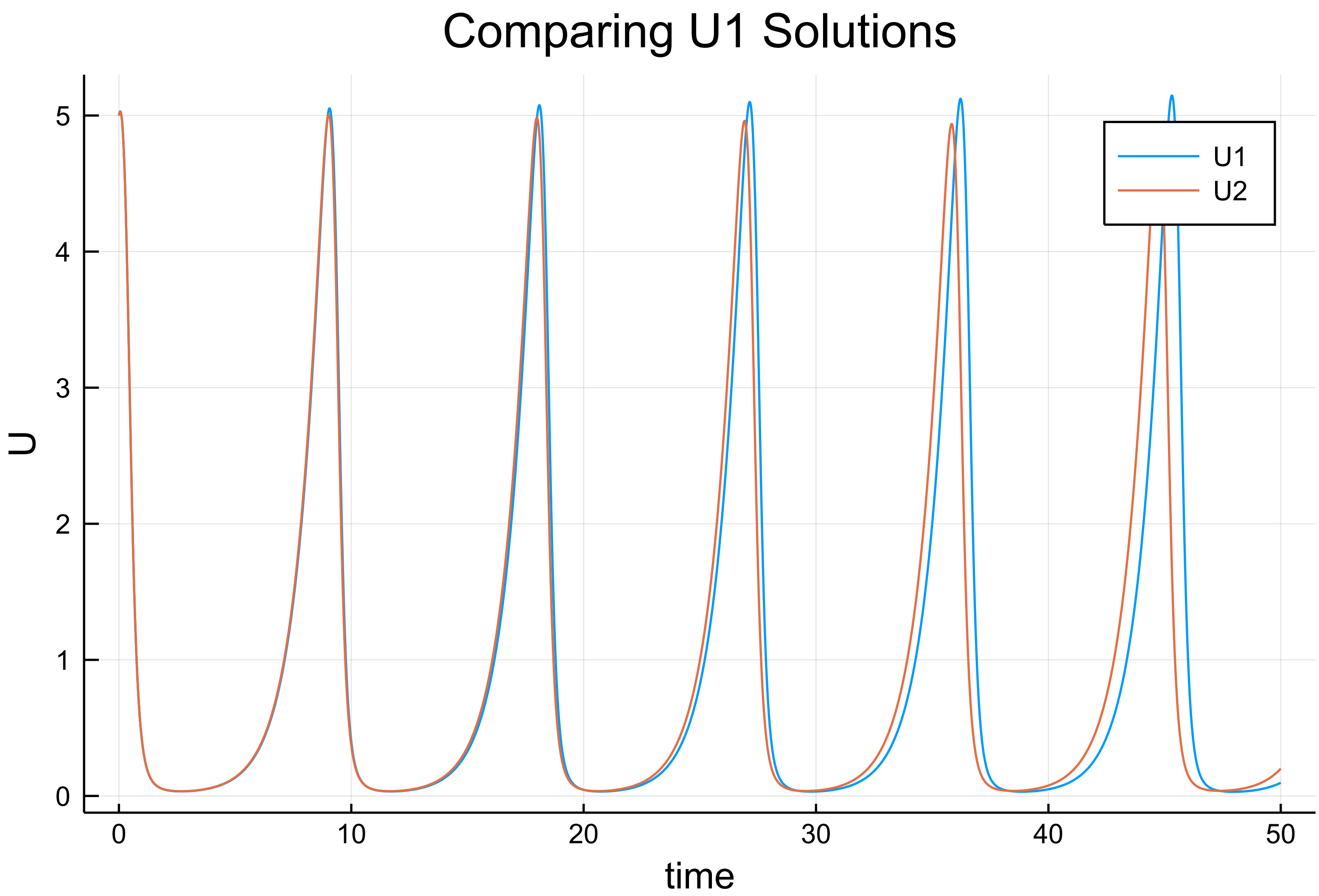
Out[14]:



In [13]:

```
t= range(0,stop = T,step = k)
plot(t,U[1,:],label ="U1",title = "Comparing U1 Solutions")
plot!(t,Ub[1,:], label = "U2")
xlabel!("time")
ylabel!("U")
```

Out[13]:



As shown in the graph above, the solution with Forward Euler grows faster over time so it has a larger amplitude and period, whereas the Backward Euler solution decays over time. This can also be seen in the U1-U2 plane where the Forward Euler solution goes further than the Backward Euler solution.

In [ ]:

```

```