hw5

April 30, 2022

0.0.1 1)

$$u' = \lambda u$$

$$U^{n+1} = U^{n-1} + 2kf(U^n)$$

$$U^{n+1} = U^{n-1} + 2k\lambda U^n$$

$$U^0 = \eta$$

Forward Euler to generate U^1:

$$U^1 = U^0 + kf(U^0)$$

$$U^1 = U^0 + k\lambda U^0 = (1 + k\lambda)\eta$$

The characteristic polynomial for the above difference equation:

$$\rho(\zeta) = \zeta^2 - 2k\lambda\zeta - 1$$

$$\zeta_{1,2} = \frac{2k\lambda \pm \sqrt{4k^2\lambda^2 + 4}}{2}$$

$$\zeta_{1,2} = k\lambda \pm \sqrt{k^2\lambda^2 + 1}$$

The general solution has the form:

$$U^n = c_1 \cdot (k\lambda - \sqrt{k^2\lambda^2 + 1})^n + c_2 \cdot (k\lambda + \sqrt{k^2\lambda^2 + 1})^n$$

Initial conditions:

$$U^0 = c_1 + c_2$$

$$\eta = c_1 + c_2$$

$$U^1 = c_1 \cdot (k\lambda - \sqrt{k^2\lambda^2 + 1}) + c_2 \cdot (k\lambda + \sqrt{k^2\lambda^2 + 1})$$

$$(1+k\lambda)\eta = (c_1+c_2)k\lambda + (c_2-c_1)\sqrt{k^2\lambda^2+1}$$

$$(1+k\lambda)\eta = \eta k\lambda + (\eta - 2c_1)\sqrt{k^2\lambda^2+1}$$

$$c_1 = \eta\frac{\sqrt{k^2\lambda^2+1}-1}{2\sqrt{k^2\lambda^2+1}}$$

$$c_2 = \eta\frac{\sqrt{k^2\lambda^2+1}+1}{2\sqrt{k^2\lambda^2+1}}$$

Hence the solution:

$$U^n = \eta\frac{\sqrt{k^2\lambda^2+1}-1}{2\sqrt{k^2\lambda^2+1}}\cdot(k\lambda-\sqrt{k^2\lambda^2+1})^n + \eta\frac{\sqrt{k^2\lambda^2+1}+1}{2\sqrt{k^2\lambda^2+1}}\cdot(k\lambda+\sqrt{k^2\lambda^2+1})^n$$

Notice that as n approaches infinity in the limit $(k\lambda-\sqrt{k^2\lambda^2+1})^n$ behaves like $e^{-\lambda}$ and $(k\lambda+\sqrt{k^2\lambda^2+1})^n$ behaves like e^{λ}. Also, since k depends on n and $k\to 0$ as $n\to\infty$, we have $\sqrt{k^2\lambda^2+1}-1\to 0$ and $\sqrt{k^2\lambda^2+1}+1\to 2$.

Therefore the limit of the numerical solution is

$$\lim_{n\to\infty} U^n \approx \eta e^{\lambda}$$

which is exponentially decaying for $\lambda < 0$. The paradox is resolved because the c_1 coefficient goes to 0 as n approaches ∞.

0.0.2 2)

$$u_1' = -K_1u_1u_2 + K_2u_3$$

$$u_2' = -K_1u_1u_2 + K_2u_3$$

$$u_3' = K_1u_1u_2 - K_2u_3$$

with $K_1 = 3, K_2 = 1$ and $u_1(0) = 3, u_2(0) = 4, u_3(0) = 2$.
You can find the implementation in ex2.m.

1. According to the textbook the Jacobian matrix is:

$$A = \begin{bmatrix} -K_1u_2 & -K_1u_1 & K_2 \\ -K_1u_2 & -K_1u_1 & K_2 \\ K_1u_2 & K_1u_1 & -K_2 \end{bmatrix}$$

with eigenvalues:

$$\lambda_1 = -K_1(u_1+u_2) - K_2 = -3(u_1+u_2) - 1$$

$$\lambda_2 = \lambda_3 = 0$$

We have:

$$0 \leq u_1(t) + u_2(t) \leq u_1(0) + u_2(0) + 2u_3(0) = 3 + 4 + 2 \cdot 2 = 11$$

Hence:

$$-1 \geq -3(u_1(t) + u_2(t)) - 1 \geq -3 \cdot 11 - 1 = -34$$

$$-k \geq k\lambda_1 \geq -3 \cdot 11 - 1 = -34k$$

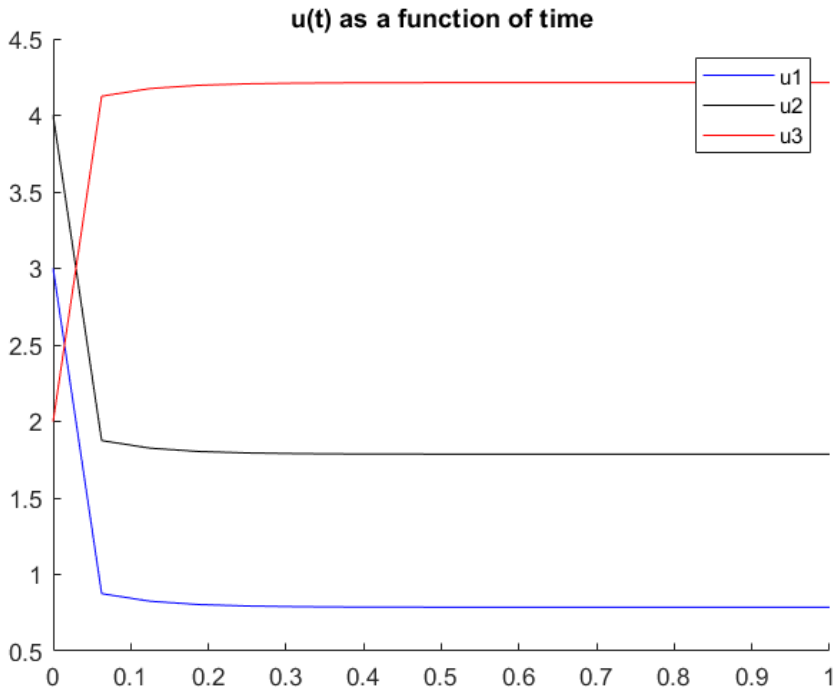
Also from the textbook it's known that for the forward Euler method:

$$0 \geq k\lambda_1 \geq -2$$

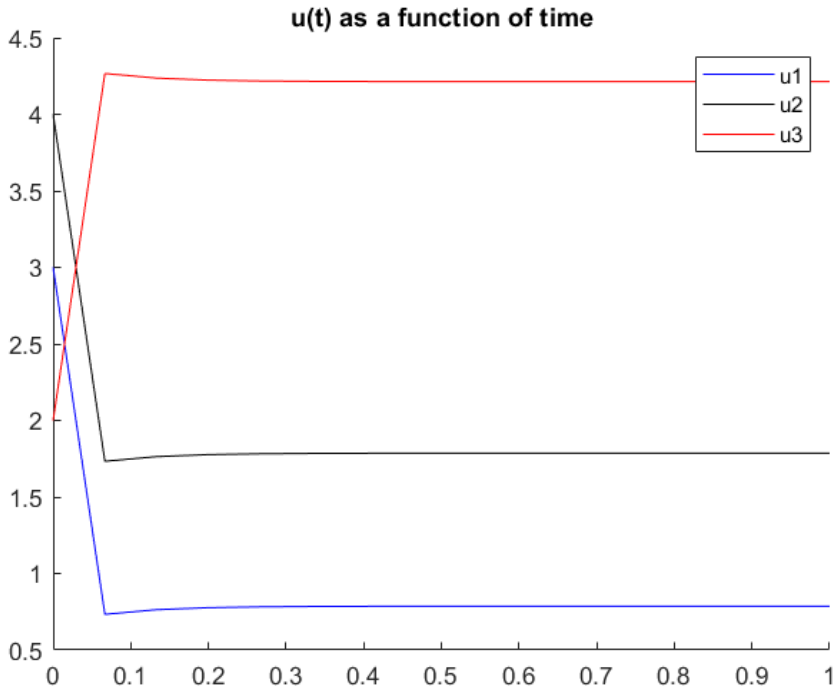
Then from both of the inequalities above:

$$0 \leq k \leq \frac{1}{17}$$

2. From the plots it can be noticed that for $m = 17$ or $k = (1-0)/16$ the program is still stable but for $m = 16$ or $k = 1/15$ we can observe instability (for example u(1) starts increasing instead of decreasing).



Plot for $k = 1/16$:



Plot for $k = 1/15$:

3. For $K_1 = 300$ and $K_2 = 1$:
Eigenvalues:

$$\lambda_1 = -K_1(u_1 + u_2) - K_2 = -300(u_1 + u_2) - 1$$

$$\lambda_2 = \lambda_3 = 0$$

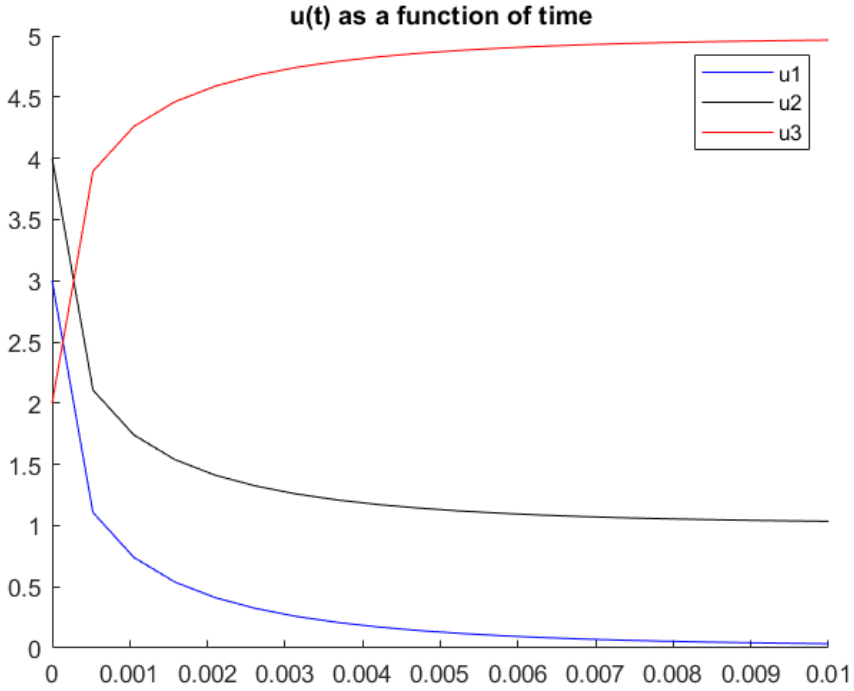
Hence:

$$-1 \geq -300(u_1(t) + u_2(t)) - 1 \geq -300 \cdot 11 - 1 = -3301$$

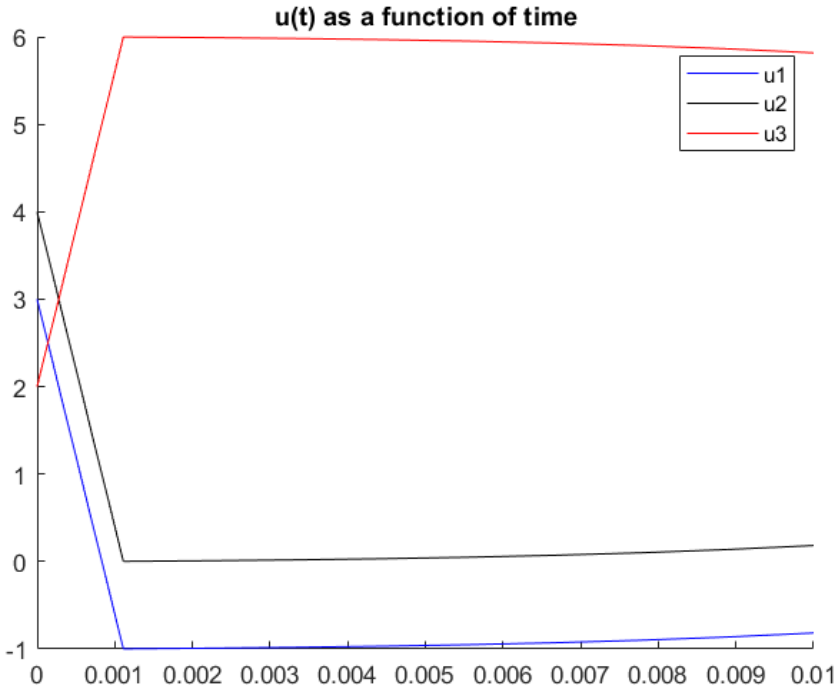
$$-k \geq k\lambda_1 \geq -3 \cdot 11 - 1 = -3301k$$

$$0 \leq k \leq \frac{2}{3301} \approx 0.0006$$

From the plots it can be noticed that for $m = 20$ or $k = (0.01 - 0)/19 \approx 0.0005$ the program is still stable but for $m = 10$ or $k \approx 0.001$ we can observe instability (for example u(1) starts increasing instead of decreasing).



Plot for $k = 0.0005$:



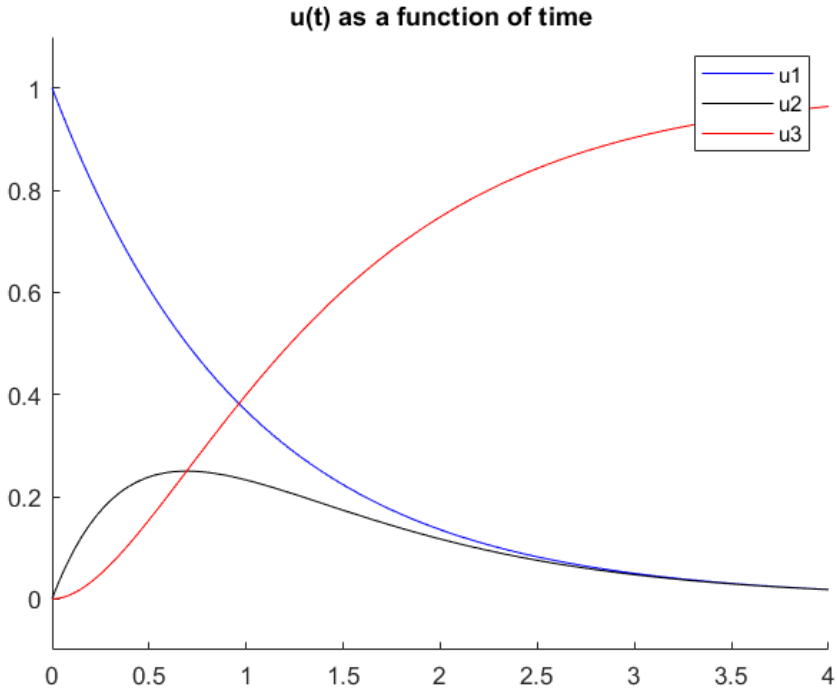
Plot for $k = 0.001$:

0.0.3 3)

1. decaytest with decay1:

```
tol          f evaluations
```

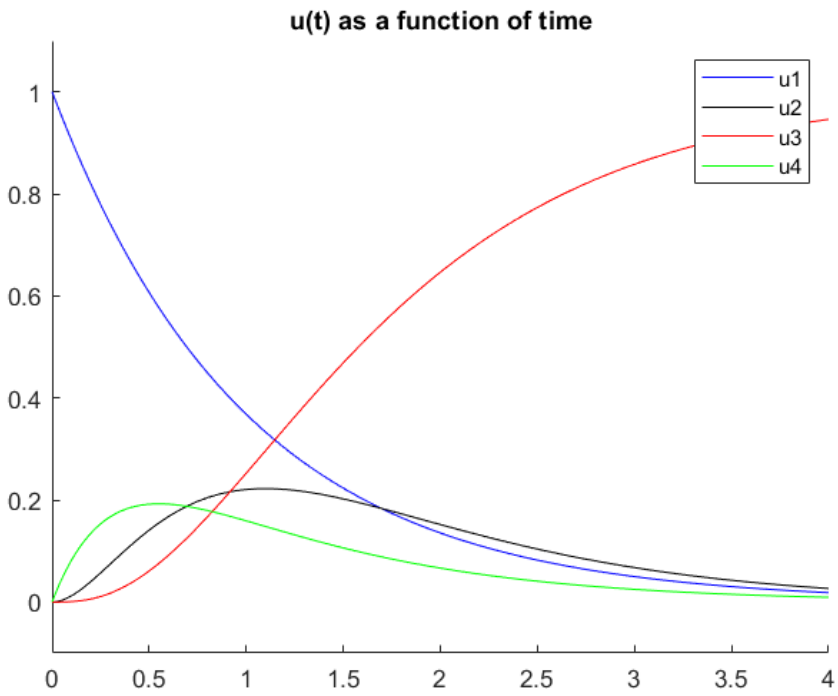
```
1.000e-02        27
1.000e-04        47
1.000e-06        77
1.000e-08       107
```



Corresponding plot:

2. decaytest with modified decay1_2:

```
   tol         f evaluations

1.000e-02        32
1.000e-04        57
1.000e-06        89
1.000e-08       127
```

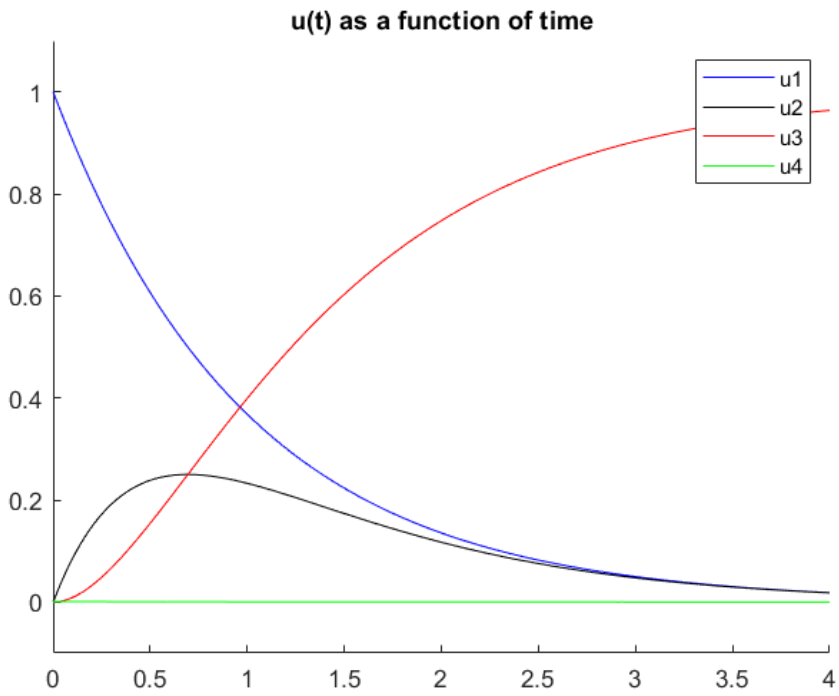


Corresponding plot:

3. decaytest with modified decay1_3:

```
tol          f evaluations

1.000e-06       5154
```

Corresponding plot (indeed the plot is nearly identical to the one from (1)):

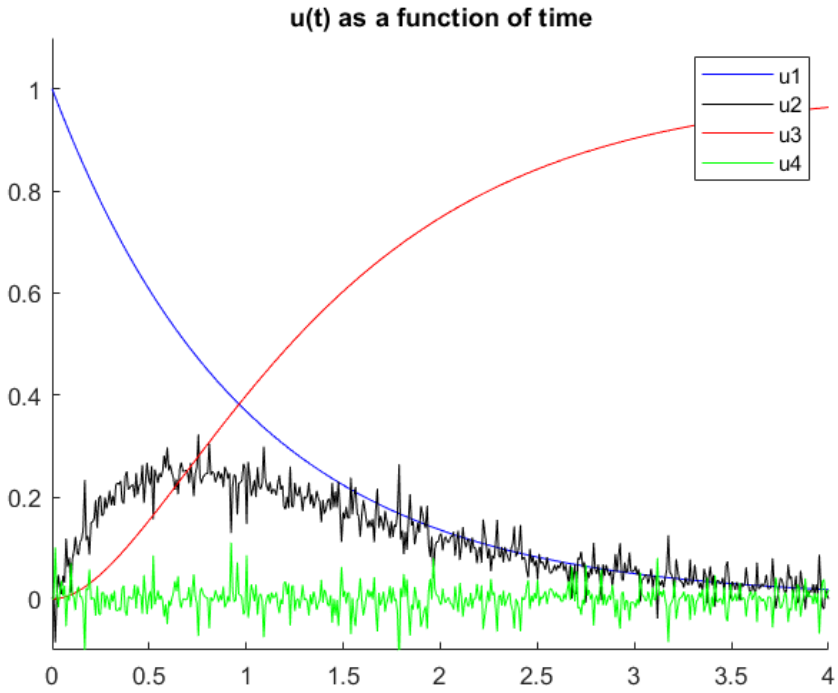


4. decaytest with modified decay1_3:

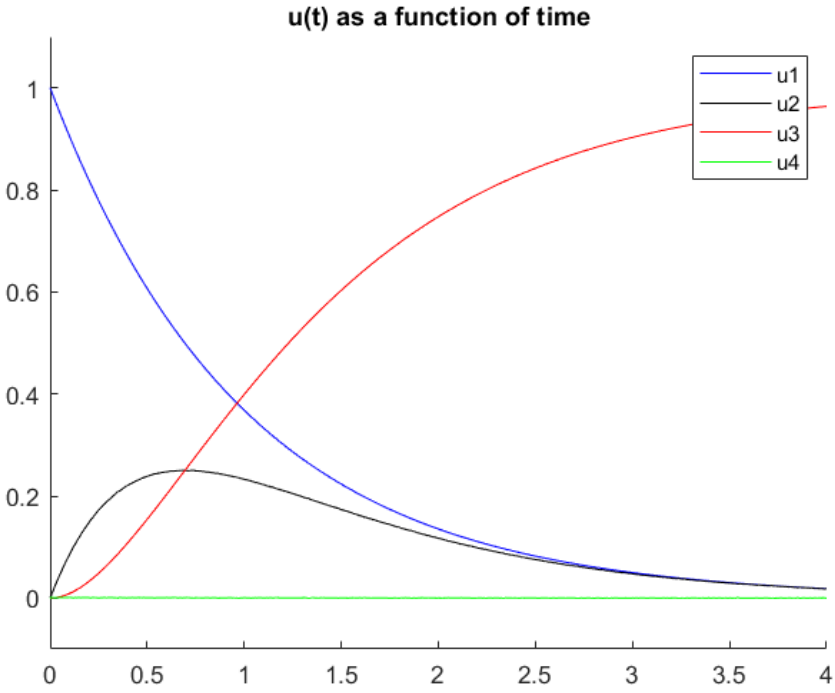
```
   tol          f evaluations

 1.000e-02       5255
 1.000e-04       5295
 1.000e-06       5154
 1.000e-08       5515
```

(a) Indeed, the number of function evaluations requires is much larger than when solving (1), even though the solution is essentially the same. It is due to the fact that the ode in (3) is a stiff one, while the ode in (1) is not. It's stiff because $f'(u)$ is much larger than $u'(t)$ because K_3 makes it large. Since $f'(u)$ measures how rapidly f varies as we move away from this particular solution, it takes a lot of time to converge to the specific solution.

(b) The number of function evaluations doesn't change much as the tolerance is reduced. Because due to the stiffness as soon as the function converged and we move towards the particular solution it starts to move rapidly and converge faster.



5. Plot for tol=1e-02:



Plot for tol1e-04:

We can see the obvious distortion happening in the first plot. For the larger tolerance due to the stiffness the solution exhibits rapid transient behavior. To make it smoother tol should be at most inverse proportional to λ_{max} which is related to K_{max}, which is 1000 in our case.

6. For $K_3 = 500$: 2802
For $K_3 = 1000$: 5154

For $K_3 = 1500$: 10634

We can observe linear dependence because the number of function evaluations depend on "stiffness ratio" of the system, which is $|\lambda_{max}/\lambda_{min}|$ where λ_{max} depends on K_3 since it's way larger than the other K_i's. When we fit these points to the linear function, we will get:

$$N = 5.2583 \cdot K_3 + 62$$

Hence for $K_3 = 10^7$ it is about $5.2583 \cdot 10^7 + 62 = \mathbf{52583062}$

7. For $K_3 = 500$: 135

For $K_3 = 1000$: 130

For $K_3 = 1500$: 140

We can observe that the number of function evaluations is much smaller and now roughly constant for large K3. It is because ode15s is a stiff ODE solver compared to ode113. It uses variable steps, which almost removes the dependence on K_3.

For $K_3 = 10^7$: **153**

0.0.4 4)

The implicit Runge-Kutta method:

$$U^* = U^n + \frac{k}{2} f(U^*, t_n + k/2),$$

$$U^{n+1} = U^n + k f(U^*, t_n + k/2).$$

Test problem $u' = \lambda u$

1.

$$U^* = U^n + \frac{k}{2}\lambda U^*,$$

$$U^* = \frac{2}{2 - \lambda k} U^n.$$

$$U^{n+1} = U^n + k\lambda U^*,$$

$$U^{n+1} = U^n + k\lambda \frac{2}{2 - \lambda k} U^n,$$

$$U^{n+1} = \frac{2 + \lambda k}{2 - \lambda k} U^n.$$

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

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

$$1 + z/2 = 1 + \frac{z}{2} - \frac{z^3}{12} - \frac{z^4}{12} + \cdots + Cz^{p+1} + O(z^{p+2})$$

and so

$$Cz^{p+1} = \frac{1}{12}z^3 + \dots$$

from which we conclude that $p = 2$, which means it is the **second** order method.

2. The characteristic polynomial:

$$\pi(\zeta; z) = \zeta - \frac{1 + z/2}{1 - z/2}]$$

Hence:

$$\zeta_1 = \frac{1 + z/2}{1 - z/2}.$$

From the book this is a linear fractional transformation and it can be shown that

$$|\zeta_1| \leq 1 \leftrightarrow Re(z) \leq 0$$

where $Re(z)$ is the real part. So the stability region is the **left half** $-$ **plane**.

3. Since its stability region contains the left half-plance, by definition, it is **A** $-$ **stable**. Also, since:

$$\lim_{z\to\infty} |R(z)| = \lim_{z\to\infty} |\frac{1 + z/2}{1 - z/2}| \neq 0$$

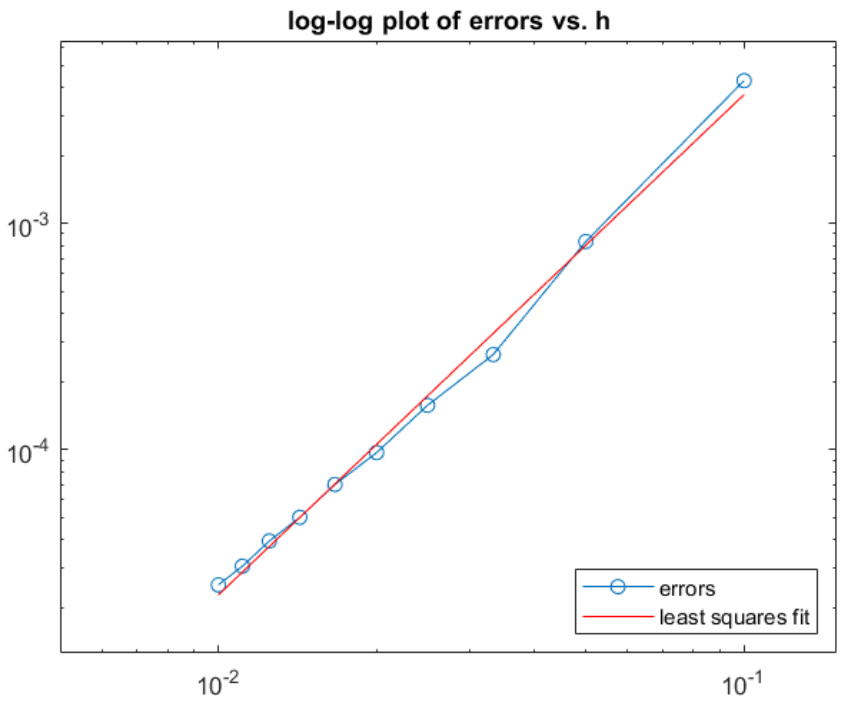
It is **not L** $-$ **stable**.

0.0.5 5)

1. The results of error_table(hvals, E):

h	error	ratio	observed order
0.10000	4.27612e-03	NaN	NaN
0.05000	8.31632e-04	5.14184	2.36229
0.03333	2.63271e-04	3.15884	2.83676
0.02500	1.57109e-04	1.67572	1.79450
0.02000	9.70909e-05	1.61816	2.15687
0.01667	7.02022e-05	1.38302	1.77855
0.01429	5.02269e-05	1.39770	2.17208
0.01250	3.95015e-05	1.27152	1.79892
0.01111	3.05708e-05	1.29213	2.17598
0.01000	2.52662e-05	1.20995	1.80881

The results of error_loglog(hvals, E):
Least squares fit gives E(h) = 0.60301 * h^2.21119

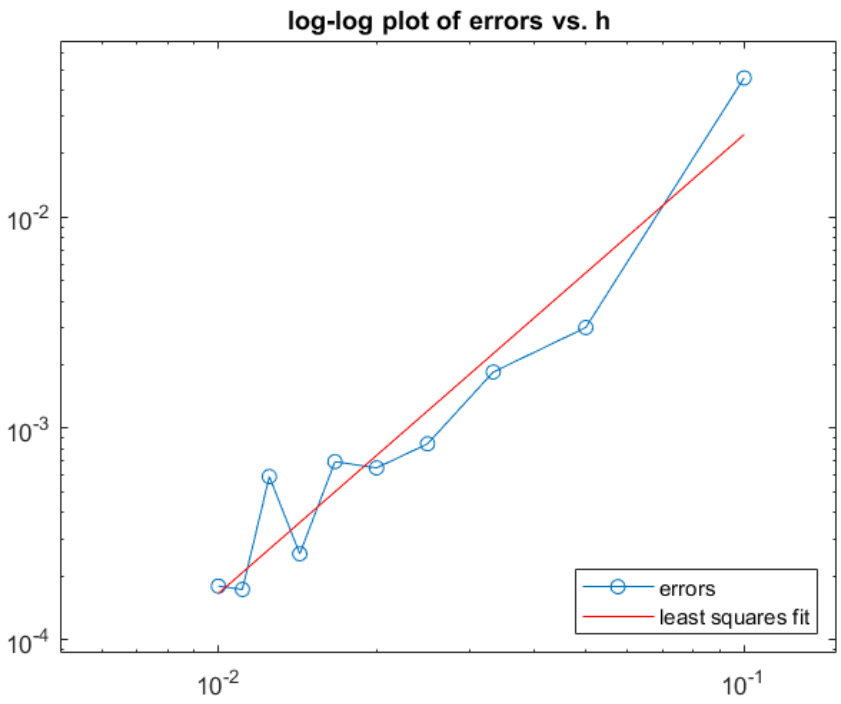


Corresponding plot:
Indeed since 2.21119 ~ 2, second-order accuracy is observed.

2. The results of error_table(hvals, E):

h	error	ratio	observed order
0.10000	4.56019e-02	NaN	NaN
0.05000	3.00158e-03	15.19262	3.92530
0.03333	1.84676e-03	1.62533	1.19790
0.02500	8.44197e-04	2.18759	2.72106
0.02000	6.49201e-04	1.30036	1.17701
0.01667	6.95041e-04	0.93405	-0.37422
0.01429	2.54517e-04	2.73082	6.51702
0.01250	5.90733e-04	0.43085	-6.30560
0.01111	1.72528e-04	3.42399	10.44977
0.01000	1.78903e-04	0.96437	-0.34439

The results of error_loglog(hvals, E):
Least squares fit gives E(h) = 3.66502 * h^2.17338



Corresponding plot:

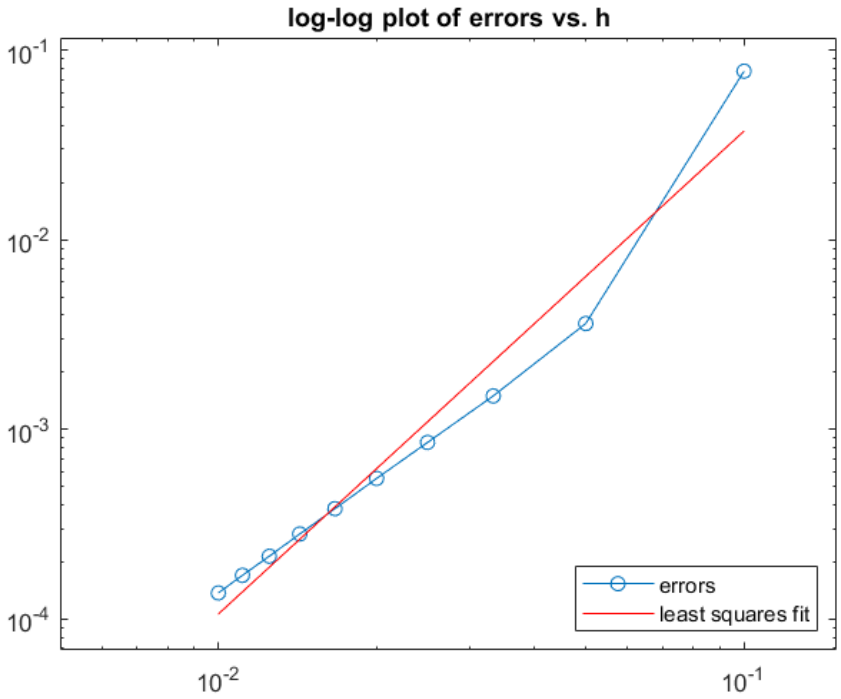
Indeed since 2.17338 ~ 2, second-order accuracy is observed. For n and n+1, I have just used the true boundaries, while for n+1/2 I have taken the average of the true boundaries for n and n+1. Also, time step is 2 times larger than given above, because I have iterated for smaller time step just for convenience to get n+1/2 iteration.

3. The results of error_table(hvals, E):

h	error	ratio	observed order
0.10000	7.73955e-02	NaN	NaN
0.05000	3.62014e-03	21.37912	4.41813
0.03333	1.50287e-03	2.40882	2.16822
0.02500	8.54974e-04	1.75779	1.96070
0.02000	5.52076e-04	1.54865	1.96011
0.01667	3.82576e-04	1.44305	2.01159
0.01429	2.81389e-04	1.35960	1.99279
0.01250	2.14846e-04	1.30973	2.02063
0.01111	1.70378e-04	1.26100	1.96889
0.01000	1.37586e-04	1.23834	2.02894

The results of error_loglog(hvals, E):

Least squares fit gives E(h) = 13.156 * h^2.54586



Corresponding plot:

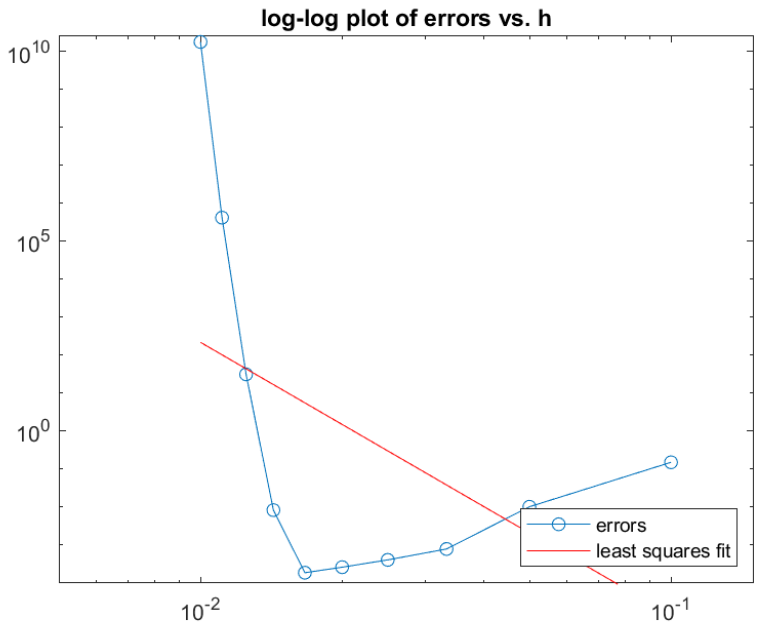
Indeed since 2.54586 ~ 2, second-order accuracy is observed. Actually for jtest=1:50 power of h in the fit equation is 2.1342, from which second-order accuracy is more obvious. CN achieves a 1e-05 tolerance for the 5th h already, while FE does not achieve it at all in our case.

4. The results of error_table(hvals, E):

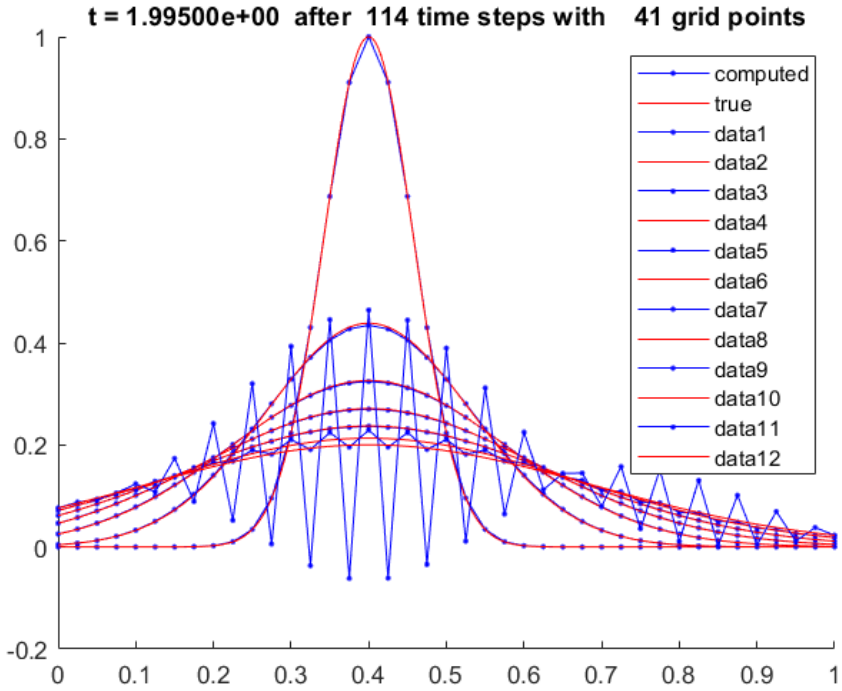
h	error	ratio	observed order
0.10000	1.44957e-01	NaN	NaN
0.05000	9.75028e-03	14.86694	3.89404
0.03333	7.49787e-04	13.00408	6.32672
0.02500	3.87530e-04	1.93478	2.29418
0.02000	2.48414e-04	1.56002	1.99287
0.01667	1.77703e-04	1.39792	1.83733
0.01429	7.94987e-03	0.02235	-24.65639
0.01250	3.02507e+01	0.00026	-61.73920
0.01111	4.08875e+05	0.00007	-80.75563
0.01000	1.75805e+10	0.00002	-101.26084

The results of error_loglog(hvals, E):

Least squares fit gives E(h) = 8.67325e-13 * h^-7.19299



Corresponding plot:



Plot for $\kappa = 0.02, m = 39, \beta = 150$:

Indeed, at about time 1.6 the instability becomes apparent and appears as a sawtooth oscillation. It's due to the fact that for our given parameters $|1 + k\lambda_p| - 1 = \epsilon > 0$ for some $|p|$ and at 100 time steps or time $100 \cdot k = 100 \cdot 26h^2 = 100 \cdot 26(1/(m+1))^2 = 100 \cdot 26(1/40)^2 \approx 1.6$ and $\epsilon^{100} >> 0$ and the instability becomes apparent. It is a sawtooth oscillation because for the most unstable eigenmode $1 + k\lambda_p < -1$ and hence for even number of time steps it will exceed the true values and for odd it will be less than the true values causing the oscillations. Proof of

$|1 + k\lambda_p| - 1 > 0$: From the book for forward Euler:

$$\lambda_p = \frac{2\kappa}{h^2}(\cos(\xi_p h) - 1) = \frac{2 \cdot 0.02}{h^2}(\cos(\xi_p 1/40) - 1)$$

$$|1 + k\lambda_p| = |1 + 26h^2\frac{2 \cdot 0.02}{h^2}(\cos(\xi_p 1/40) - 1)| = |1 + 1.04(\cos(\xi_p 1/40) - 1)| = |1.04\cos(\xi_p 1/40) - 0.04|$$

Then from above for $\xi_p > 2.76$ the following is true $|1 + k\lambda_p| - 1 > 0$.