Mario L. Gutierrez Abed
364009832
mlg3843@rit.edu

Problem Set 1
Numerical Analysis II

08-13-2021

Problem 1. *Use the existence theorem on initial-value problems to predict in what interval a solution of the following initial value problems exist and find the largest interval:*

a) $x' = x\tan(t+3)$; $\quad x(-3) = 1$.

b) $x' = 1 + x^2$; $\quad x(0) = 0$.

Existence Theorem for IVPs

For an Initial Value Problem (IVP)

$$\frac{\mathrm{d}x}{\mathrm{d}t} = f(t,x); \qquad x(t_0) = x_0, \tag{1}$$

if the function f is continuous in a rectangle

$$R = \{(t,x) \colon |t - t_0| \le \alpha,\ \ |x - x_0| \le \beta\}, \tag{2}$$

then the IVP (1) has a solution $x(t)$ for

$$|t - t_0| \le \min\left\{\alpha, \frac{\beta}{M}\right\}, \tag{3}$$

where M is the maximum of $f(t,x)$ in the rectangle R.

Solution to a). The tangent function can only take input values in $(-\pi/2, \pi/2)$; thus from $\tan(t+3)$ we deduce that $t + 3 \in (-\pi/2, \pi/2)$, which implies that

$$t \in \left(-\frac{\pi+6}{2}, \frac{\pi-6}{2}\right).$$

We have the Cauchy data $(t_0, x_0) = (-3, 1)$. Let us find the maximum value M, attained by f in the rectangle

$$R = \{(t,x) \colon |t+3| \le \alpha,\ \ |x-1| \le \beta\}.$$

The value M is then determined from

$$|f(t,x)| = |x\tan(t+3)| \le |x|\ |\tan(t+3)| \le (\beta+1)(\tan\alpha) \equiv M.$$

Now, we have

$$|t_{\max} - t_0| = \left|\frac{\pi-6}{2} - (-3)\right| = \frac{\pi}{2},$$

so the inequality (3) must satisfy

$$\frac{\pi}{2} \le \min\left\{\alpha, \frac{\beta}{M}\right\}.$$

The quantity α cannot possibly satisfy this inequality, since setting $\alpha = \pi/2$ would yield $\tan\alpha = \infty$. Thus we must choose

$$\frac{\pi}{2} \le \frac{\beta}{M}.$$

Setting this inequality to equality yields

$$\frac{\pi}{2} = \frac{\beta}{(\beta+1)(\tan\alpha)} \qquad \Longrightarrow \qquad \alpha = \arctan\frac{2\beta}{\pi(\beta+1)} \in (-\pi/2, \pi/2).$$

But since $\beta \geq 0$, we have $\alpha \in [0, \pi/2)$. In other words,

$$0 \leq t + 3 < \frac{\pi}{2} \implies -3 \leq t < \frac{\pi - 6}{2}.$$

Hence the largest interval for which the IVP is defined is $t \in [-3, \pi/2 - 3)$. ◻

Solution to b). The equation is autonomous and at first sight it seems to be defined for all $t \in \mathbb{R}$. However, if we solve the IVP (with the given Cauchy data $(t_0, x_0) = (0, 0)$), we have $x(t) = \tan t$, or, in terms of t,

$$t = \arctan x.$$

Thus, the IVP has solution only in $t \in (-\pi/2, \pi/2)$. ◻

Problem 2. *Find the solution of the initial-value problem $x' = ax + b$, $x(a_0) = x_0$ $(a \neq 0)$ and verify the inequality*

$$|x_1(t) - x_2(t)| \leq e^{L(t-a_0)} |x_1(a_0) - x_2(a_0)|.$$

Solution. We have

$$\frac{\mathrm{d}x}{\mathrm{d}t} = ax + b \implies \int \frac{\mathrm{d}x}{ax + b} = \int \mathrm{d}t \implies \frac{1}{a} \log |ax + b| = t + C \implies x(t) = Ce^{at} - \frac{b}{a}.$$

Moreover, from $x(a_0) = x_0$ we get

$$x_0 = Ce^{aa_0} - \frac{b}{a} \implies C = \left(x_0 + \frac{b}{a}\right) e^{-aa_0}.$$

Thus, the final solution to the IVP is

$$\boxed{x(t) = \left(x_0 + \tfrac{b}{a}\right) e^{a(t-a_0)} - \tfrac{b}{a}.}$$

Now, assume we have two different solutions x_1, x_2. Then,

$$\begin{aligned}
|x_1(t) - x_2(t)| &= \left|\left(x_1(a_0) + \frac{b}{a}\right) e^{a(t-a_0)} - \frac{b}{a} - \left[\left(x_2(a_0) + \frac{b}{a}\right) e^{a(t-a_0)} - \frac{b}{a}\right]\right| \\
&= \left|(x_1(a_0) - x_2(a_0))\, e^{a(t-a_0)}\right| \\
&\leq \left|e^{a(t-a_0)}\right| \, |(x_1(a_0) - x_2(a_0))| && \text{(subadditivity of the norm)} \\
&= e^{a(t-a_0)} \, |(x_1(a_0) - x_2(a_0))| \, .
\end{aligned}$$

Letting $a = L$, we have proven the desired inequality. Of course, we could have also noticed that the line $ax + b$ is Lipschitz, and thus Theorem 6.3 from Sauer's applies and the inequality is guaranteed. ◻

Problem 3. *Derive the third-order Runge-Kutta formula*

$$u(t+k) = u(t) + \frac{k}{6}\left(F_1 + 4F_2 + F_3\right), \tag{4a}$$

where

$$F_1 = f\left(t, u\right) \tag{4b}$$
$$F_2 = f\left(t + \frac{k}{2}, u + \frac{k}{2}F_1\right) \tag{4c}$$
$$F_3 = f\left(t + k, u - kF_1 + 2kF_2\right). \tag{4d}$$

Solution. We tackle this problem starting from a general three-stage evaluation:

$$U^{n+1} = U^n + k\left(\alpha_1 F_1 + \alpha_2 F_2 + \alpha_3 F_3\right) \tag{5a}$$

with

$$F_1 = f\left(t_n, U^n\right) \tag{5b}$$
$$F_2 = f\left(t_n + p_1 k, U^n + q_{11} k F_1\right) \tag{5c}$$
$$F_3 = f\left(t_n + p_2 k, U^n + q_{21} k F_1 + q_{22} k F_2\right). \tag{5d}$$

Here $\alpha_i, p_i, q_{ij} \in \mathbb{R}\ \forall i, j$ are coefficients that need to be determined.

Now, an expansion of $u(t)$ about some time step t_n, using $u_t = f$, yields

$$U^{n+1} = U^n + kf + \frac{k^2}{2}\left[f_t + f f_u\right] + \frac{k^3}{6}\left[f_{tt} + 2f_{tu}f + f_t f_u + f_{uu}f^2 + f_u^2 f\right] + O(k^4). \tag{6}$$

Unraveling this expansion in Eqs. (5), we get

$$\begin{aligned}
U^{n+1} &= U^n + k\left(\alpha_1 + \alpha_2 + \alpha_3\right) f + k^2\left[\alpha_2(p_1 f_t + q_{11} f f_u) + \alpha_3(p_2 f_t + q_{21} f f_u + q_{22} f f_u)\right] \\
&\quad + k^3\{\frac{\alpha_2}{2}\left(p_1^2 f_{tt} + q_{11}^2 f^2 f_{uu} + 2p_1 q_{11} f f_{tu}\right) \\
&\qquad + \alpha_3\left[q_{22}(p_1 f_t f_u + q_{11} f f_u^2) + \frac{1}{2}\left(p_2^2 f_{tt} + f^2(q_{21}+q_{22})^2 f_{uu} + 2(q_{21}+q_{22}) f p_2 f_{tu}\right)\right]\} + O(k^4).
\end{aligned} \tag{7}$$

Comparing Eqs. (6) and (7), we get the following messy linear system:

$$\begin{aligned}
\alpha_1 + \alpha_2 + \alpha_3 &= 1 \\
p_1\alpha_2 + p_2\alpha_3 &= \frac{1}{2} \\
q_{11}\alpha_2 + (q_{21}+q_{22})\alpha_3 &= \frac{1}{2} \\
p_1^2\alpha_2 + p_2^2\alpha_3 &= \frac{1}{3} \\
p_1 q_{11}\alpha_2 + p_2(q_{21}+q_{22})\alpha_3 &= \frac{1}{3} \\
p_1 q_{22}\alpha_3 &= \frac{1}{6} \\
q_{11}^2\alpha_2 + (q_{21}+q_{22})^2\alpha_3 &= \frac{1}{3} \\
q_{11} q_{22}\alpha_3 &= \frac{1}{6}.
\end{aligned}$$

Thus we have eight equations and eight unknowns; however, only six of these equations are independent. As a consequence, there is not ***one*** RK3 formulation, but various. To get the one presented in this problem we may set $p_2 = 1$ and $q_{11} = 1/2$. Plugging these values in Mathematica we get the system (4), as desired. □

Problem 4. *Show that when the fourth-order Runge-Kutta method is applied to the problem $x' = \lambda x$, the formula for advancing the solution will be*

$$x(t+k) = \left[1 + k\lambda + \frac{1}{2}k^2\lambda^2 + \frac{1}{6}k^3\lambda^3 + \frac{1}{24}k^4\lambda^4\right] x(t).$$

Solution. We set $f(t,x) := \lambda x$; then, applying RK4,

$$\begin{aligned}
k_1 &= f(t,x) = \lambda x \\
k_2 &= f\left(t + \frac{k}{2}, x + \frac{k}{2}k_1\right) \\
&= \lambda\left(x + \frac{k}{2}\lambda x\right) \\
&= \lambda x\left(1 + \frac{k}{2}\lambda\right) \\
k_3 &= f\left(t + \frac{k}{2}, x + \frac{k}{2}k_2\right) \\
&= \lambda\left(x + \frac{k}{2}\lambda x\left(1 + \frac{k}{2}\lambda\right)\right) \\
&= \lambda x\left(1 + \frac{k}{2}\lambda + \frac{k^2}{4}\lambda^2\right) \\
k_4 &= f\left(t + k, x + kk_3\right) \\
&= \lambda\left(x + k\lambda x\left(1 + \frac{k}{2}\lambda + \frac{k^2}{4}\lambda^2\right)\right) \\
&= \lambda x\left(1 + k\lambda + \frac{k^2}{2}\lambda^2 + \frac{k^3}{4}\lambda^3\right).
\end{aligned}$$

Plugging the k_i into the RK4 formula, we get

$$\begin{aligned}
x(t+k) &= x(t) + \frac{k}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \\
&= x + \frac{k}{6}\{\lambda x + 2\lambda x\left(1 + \frac{k}{2}\lambda\right) + 2\lambda x\left(1 + \frac{k}{2}\lambda + \frac{k^2}{4}\lambda^2\right) \\
&\quad + \lambda x\left(1 + k\lambda + \frac{k^2}{2}\lambda^2 + \frac{k^3}{4}\lambda^3\right)\} \\
&= x\left(1 + k\lambda + \frac{1}{2}k^2\lambda^2 + \frac{1}{6}k^3\lambda^3 + \frac{1}{24}k^4\lambda^4\right). \quad \surd
\end{aligned}$$

□

Problem 5. *Use the method of undetermined coefficients to derive the fourth-order Adams-Bashforth formula*

$$x_{n+1} = x_n + \frac{k}{24}\left[55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-3}\right]. \tag{8}$$

Solution. The general fourth-order Adams-Bashforth formula is of the form

$$x_{n+1} = x_n + k\left[\alpha_3 f_n + \alpha_2 f_{n-1} + \alpha_1 f_{n-2} + \alpha_0 f_{n-3}\right]. \tag{9}$$

We now need to determine the proper coefficients α_i so that Eq. (9) is exact when the exact solution x is a polynomial of maximal degree. Thus in our case we must have

$$x_n = x(t_n) \quad \text{and} \quad f_n = x'(t_n)$$

for all x which are polynomials of degree up to 4. We set up the polynomials

$$x(t) = (t - t_n)^j \qquad \text{for } j = 1, 2, 3, 4. \tag{10}$$

We then have

$$x'(t) = j(t - t_n)^{j-1} \qquad \text{and} \qquad x_{n+1} - x_n = (t_{n+1} - t_n)^j = k^j.$$

Moreover, since we must have $f_n = x'(t_n)$,

$$\begin{aligned}
f_n &= x'(t_n) = \begin{cases} 1 & \text{if } j = 0 \\ 0 & \text{if } j > 0 \end{cases} \\
f_{n-1} &= x'(t_{n-1}) = j(-k)^{j-1} \\
f_{n-2} &= x'(t_{n-2}) = j(-2k)^{j-1} \\
f_{n-3} &= x'(t_{n-3}) = j(-3k)^{j-1}.
\end{aligned}$$

Putting all this together and plugging back into Eqs. (9)-(10), we get the equations

$$\begin{aligned}
x(t) = (t - t_n) \quad &\implies \quad k = k\left[\alpha_3 + \alpha_2 + \alpha_1 + \alpha_0\right] \\
x(t) = (t - t_n)^2 \quad &\implies \quad k^2 = 2k\left[0\alpha_3 - k\alpha_2 - 2k\alpha_1 - 3k\alpha_0\right] \\
x(t) = (t - t_n)^3 \quad &\implies \quad k^3 = 3k\left[0\alpha_3 + k^2\alpha_2 + (2k)^2\alpha_1 + (3k)^2\alpha_0\right] \\
x(t) = (t - t_n)^4 \quad &\implies \quad k^4 = 4k\left[0\alpha_3 - k^3\alpha_2 - (2k)^3\alpha_1 - (3k)^3\alpha_0\right].
\end{aligned}$$

This leads to the system

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ -3 & -2 & -1 & 0 \\ 9 & 4 & 1 & 0 \\ -27 & -8 & -1 & 0 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1/2 \\ 1/3 \\ 1/4 \end{bmatrix},$$

which has solution

$$\begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} -3/8 \\ 37/24 \\ -59/24 \\ 55/24 \end{bmatrix}.$$

Plugging these values for the α_i in Eq. (9) we recover (8), as desired. □

Problem 6. *Find a two-step, third-order explicit method. Is the method stable?*

Solution. The general two-step explicit method is of the form

$$x_{n+1} = \alpha_1 x_n + \alpha_2 x_{n-1} + k[\beta_1 f_n + \beta_2 f_{n-1}]. \tag{11}$$

Taylor-expanding the RHS up to order k^3, and using $x' = f$,

$$\begin{aligned}
x_{n+1} = x(t+k) &= \alpha_1 x + \alpha_2\left[x - kx' + \frac{k^2}{2}x'' - \frac{k^3}{6}x''' + O(k^4)\right] + \beta_1 kx' + \beta_2\left[kx' - k^2x'' + \frac{k^3}{2}x''' + O(k^4)\right] \\
&= (\alpha_1 + \alpha_2)x + (\beta_1 + \beta_2 - \alpha_2)kx' + (\alpha_2 - 2\beta_2)\frac{k^2}{2}x'' + (3\beta_2 - \alpha_2)\frac{k^3}{6}x''' + O(k^4).
\end{aligned}$$

Comparing this with the Taylor expansion of the LHS of (11), namely

$$x_{n+1} = x(t+k) = x + kx' + \frac{k^2}{2}x'' + \frac{k^3}{6}x''' + O(k^4),$$

we get the system

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 1 \\ 0 & 1 & 0 & -2 \\ 0 & -1 & 0 & 3 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix},$$

which has solution

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} -4 \\ 5 \\ 4 \\ 2 \end{bmatrix}.$$

Plugging these coefficients back into Eq. (11), we get

$$x_{n+1} = -4x_n + 5x_{n-1} + k[4f_n + 2f_{n-1}],$$

which has characteristic polynomial $p(x) = x^2 + 4x - 5$. The roots are $r_1 = -5$, $r_2 = 1$; thus, since $|r_1| = 5 > 1$, the method is unstable. □

Problem 7. *Find a second-order, two-step implicit method that is weakly stable.*

Solution. The general two-step implicit method is of the form

$$x_{n+1} = \alpha_1 x_n + \alpha_2 x_{n-1} + k[\beta_0 f_{n+1} + \beta_1 f_n + \beta_2 f_{n-1}]. \tag{12}$$

Taylor-expanding the RHS up to order k^2, and using $x' = f$,

$$\begin{aligned} x_{n+1} = x(t+k) &= \alpha_1 x + \alpha_2 \left[x - kx' + \frac{k^2}{2}x'' + O(k^3)\right] + \beta_0 \left[kx' + k^2x'' + O(k^3)\right] + \beta_1 kx' + \beta_2 \left[kx' - k^2x'' + O(k^3)\right] \\ &= (\alpha_1 + \alpha_2)x + (\beta_0 + \beta_1 + \beta_2 - \alpha_2)kx' + (\alpha_2 + 2\beta_0 - 2\beta_2)\frac{k^2}{2}x'' + O(k^3). \end{aligned}$$

Comparing this with the Taylor expansion of the LHS of (12), namely

$$x_{n+1} = x(t+k) = x + kx' + \frac{k^2}{2}x'' + O(k^3),$$

we get the system

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & -2 & 1 & 1 & 1 \\ 0 & 1 & 2 & 0 & -2 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Unlike in the previous problem, this system is underdetermined, so it has infinitely many solutions. We may then choose values that yield a weakly stable system. Consider then, the characteristic polynomial $p(x) = x^2 - \alpha_1 x - \alpha_2$; seting $\alpha_1 = 0$, $\alpha_2 = 1$ we get roots $r_1 = 1$, $r_2 = -1$, thus yielding a weakly stable system. With these choices for α_1, α_2, we end up with the square system

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 0 & -2 \\ 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 2 \end{bmatrix},$$

which has solution

$$\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} 2 \\ -2 \\ 2 \end{bmatrix}.$$

Plugging these coefficients back into Eq. (12), we get

$$x_{n+1} = x_{n-1} + k[2f_{n+1} - 2f_n + 2f_{n-1}]. \qquad \square$$

Problem 8. *Apply the Euler method, the explicit Trapezoid method, and the fourth-order Runge-Kutta method on a grid/mesh of step-size* h = 0.1 *in* [0, 1] *for the initial value problem*

$$x' = \frac{t^3}{x^2}; \quad x(0) = 1.$$

Print a table of the t values, approximations, and global error at each step. $\boxed{C}$

Solution. First we compute the analytical solution, so that we can show the global error at each time-step. We have

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \frac{t^3}{x^2} \implies \int_0^1 x^2 \,\mathrm{d}x = \int_0^1 t^3 \,\mathrm{d}t$$
$$\implies x = \sqrt[3]{\frac{3}{4}t^4 + 1}.$$

Now the three algorithms are written in the following snippet:

```julia
# Euler's method
function euler(f::Function, init::Float64, a::Float64, b::Float64, k::Float64)::Array
    mesh = collect(a:k:b)
    n = length(mesh)
    xval = zeros(n)
    xval[1] = init
    for i = 2:n
        xval[i] = xval[i-1] + k*f(mesh[i-1],xval[i-1])
    end
    return xval
end


# Explicit Trapezoid method
function trap(f::Function, init::Float64, a::Float64, b::Float64, k::Float64)::Array
    mesh = collect(a:k:b)
    n = length(mesh)
    xval = zeros(n)
    xval[1] = init
    for i = 2:n
        xval[i] = xval[i-1] + (k/2)*(f(mesh[i-1],xval[i-1]) + f(mesh[i-1] + k, xval[i-1] + k*f(mesh[i-1],xval[i-1]))
    end
    return xval
end


# RK4
function rk4(f::Function, init::Float64, a::Float64, b::Float64, k::Float64)::Array
    mesh = collect(a:k:b)
    n = length(mesh)
    xval = zeros(n)
    xval[1] = init
    for i = 2:n
        k1 = f(mesh[i-1], xval[i-1])
        k2 = f(mesh[i-1] + (k/2), xval[i-1] + (k/2)*k1)
        k3 = f(mesh[i-1] + (k/2), xval[i-1] + (k/2)*k2)
        k4 = f(mesh[i-1] + k, xval[i-1] + k*k3)
        xval[i] = xval[i-1] + (k/6)*(s1 + 2*k2 + 2*k3 + k4)
    end
    return xval
end
```

We report t values, approximations, and global errors at each step in the following tables:

Time step	Approximation	Global Error
0.0	1.0	0.0
0.1	1.0	2.4999375026091286e-5
0.2	1.0001	0.00029984010658146154
0.3	1.0008998400239968	0.0011210731348108016
0.4	1.0035949874386747	0.0027644839531697496
0.5	1.0099492185549162	0.0054378065565037215
0.6	1.0222041514646962	0.009199345735735776
0.7	1.0428759598319033	0.01386823312634888 3
0.8	1.0744135719473906	0.018990481908894852
0.9	1.1187669793450852	0.02392773343351462
1.0	1.1770105845130252	0.028060547574589867

Time step	Approximation	Global Error
0.0	1.0	0.0
0.1	1.00005	2.500062497401423e-5
0.2	1.0004998750353655	0.00010003492878407627
0.3	1.0022459747531869	0.0002250615943792944
0.4	1.0067585862791142	0.0003991148872697359
0.5	1.0160055035569127	0.0006184784454927872
0.6	1.0322775413747798	0.0008740441743477767
0.7	1.0578930427791458	0.0011488498208935827
0.8	1.094822122437412	0.0014180685811264748
0.9	1.1443484019301737	0.0016536891515739338
1.0	1.2069039795933072	0.0018328475056921256

Time step	Approximation	Global Error
0.0	1.0	0.0
0.1	1.0000249995312636	1.562374674080047e-10
0.2	1.0003998416710735	1.564492091432612 1e-9
0.3	1.002020918810963	5.6521554103738936e-9
0.4	1.0063594870104478	1.5618603299571987e-8
0.5	1.015387067172 2036	4.206078374480171e-8
0.6	1.0314036096077206	1.1240728858830096e-7
0.7	1.056744466006829	2.7304857685983563e-7
0.8	1.0934046196732778	5.658169923705714e-7
0.9	1.1426956974172813	9.846386814782448e-7
1.0	1.20507259023186 3	1.458144248012161e-6

Figure 1: From left to right, reported values for Euler's Method, Trapezoid, and RK4, respectively. □

Problem 10. *Apply Backward Euler, using Newton's method as a solver, for the initial-value problem*

$$x' = x^2 - x^3, \quad x(0) = \frac{1}{2}, \quad t \in [0, 20]. \tag{13}$$

Which of the equilibrium solutions are approached by the approximate solutions. Then apply Euler's method. For what approximate range of k (= Δt) can Euler be used successfully to converge to the equilibrium? Plot approximate solutions given by Backward Euler, and the Euler with an excessive step size. $\boxed{C}$

Solution. Equilibrium solutions are given by $x = 0$ and $x = 1$. The Backward Euler method applied to Eq. (13) is given by

$$x_{n+1} = x_n + kf(t_{n+1}, x_{n+1}) = x_n + k\left(x_{n+1}^2 - x_{n+1}^3\right).$$

We set $g(x_{n+1}) := x_{n+1} - x_n - k\left(x_{n+1}^2 - x_{n+1}^3\right) = 0$, and apply Newton's method:

$$x_{n+1} = x_n - \frac{g(x_{n+1})}{g'(x_{n+1})} = x_n - \frac{x_{n+1} - x_n - k\left(x_{n+1}^2 - x_{n+1}^3\right)}{1 - k\left(2x_{n+1} - 3x_{n+1}^2\right)}.$$

Using the code

```
void Newton (double (*func)(double), double (*funcder)(double), double x0,
             double epsilon = 0.5 * 1e-6, int itmax = 50){

    double x {x0};    //initialization of variable x to our initial guess x0
    double newx {};   //initialization of variable newx

    for (int i {0}; i <= itmax; i++){
        cout << "Iteration # " << i << "." << " x = " << x << "." << endl;
        newx = x - ( func(x) )/( funcder(x) );
        if (abs(newx - x) <= epsilon) {
            cout << "Newton Method has converged within given tolerance.
                     The root found is x = " << newx << "." << endl;
            break;
        }
        if (i == itmax) {
            cout << "Newton Method has failed to converge after the maximum
                    allowed number of iterations. " << endl;
        }
        x = newx;
    }
}
```

we can see in the results that the apprximate solution is converging to the equilibrium $x = 1$:

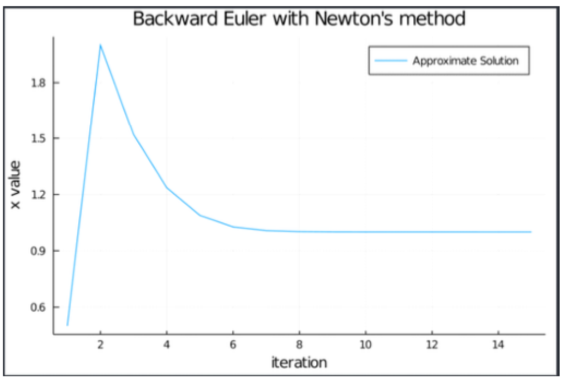
```
0.5
0.5128205128205128
0.5259431926098592
0.5393541475635261
0.5530363210713958
0.5669693406269469
0.5811294153404437
0.595489293551665
0.61001829159374
0.6246824035632718
0.6394444999175809
0.6542646198211766
0.6691003584524298
0.6839073461014952
0.6986398110807395
0.7132512135540579
⋮
0.9211840851133744
0.9274327343471348
0.9332522505307009
0.9386619449822574
0.9436818635903209
0.9483324727350265
0.9526343829792471
0.9566081107600132
0.9602738770865917
0.9636514413293217
0.9667599675351874
0.9696179202959495
0.9722429869826438
0.9746520231066147
0.9768610176346681
0.9788850752411983
```

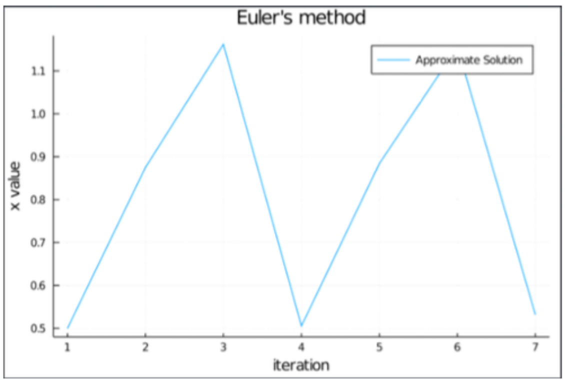
Similarly, using Euler's method we get convergence towards $x = 1$:

```
0.5
0.5125
0.5253044921875
0.5384034683354696
0.5517841498800274
0.5654307872898874
0.5793244854262165
0.5934430649295934
0.6077609704274387
0.6222492367132191
⋮
0.9999999947368131
0.9999999952631318
0.9999999957368186
0.9999999961631367
0.999999996546823
0.9999999968921407
0.9999999972029266
0.999999997482634
0.9999999977343706
```

Now, to find the approximate range of k where Euler can be used successfully to converge to $x = 1$, let us define the function $\Xi(x) := x + k(x^2 - x^3)$. Then Ξ has a fixed point at $x = 1$ and will converge to this point if $|\Xi'(1)| = |1 - k| < 1$, which in turn implies that $k \in (0, 2)$.

To finalize, we run both Backward and Forward Euler with an excessive step size (say, $k = 3$), and plot the results:





The horrible outcome of Forward Euler's implementation should come at no surprise, given that we chose a value of k outside of the stable range $0 < k < 2$. □

Problem 11. *Apply the Adams-Bashforth Two-Step method to the initial-value problem*

$$x' = \frac{t^3}{x^2} \qquad x(0) = 1.$$

Using step size $h = 0.1$, *calculate the approximation on the interval* $[0, 1]$. *Print a table of* t *values, approximations, and global truncation error at each step.* $\boxed{C}$

Solution. This is the same IVP that we dealt with in Problem 8. The Adams-Bashforth Two-Step method is given by

$$x_{n+1} = x_n + k\left[\frac{3}{2}f_n - \frac{1}{2}f_{n-1}\right].$$

We code this method in the following snippet, using $x_0 = 1$ and applying first RK4 to get x_1:

```
# Adams-Bashforth Two-Step Method
function adam_b2(f::Function, t0::Float64, t1::Float64, a::Float64, b::Float64, k::Float64)::Array
    mesh = collect(a:k:b)
    n = length(mesh)
    xval = zeros(n)
    xval[1] = t0
    xval[2] = t1
    for i = 3:n
        xval[i] = xval[i-1] + k*(1.5*f(mesh[i-1],xval[i-1]) - 0.5*f(mesh[i-2],xval[i-2]))
    end
    return xval
end
```

The results are given in the following table:

Time step	Approximation	Global Error
0.0	1.0	0.0
0.1	1.0000249995312636	1.562374674080047e-10
0.2	1.0001749920316854	0.00022484807489608905
0.3	1.001324574660883	0.000696338497924609
0.4	1.0049640068425822	0.0013954645492622042
0.5	1.0131229721727706	0.002264052938649286
0.6	1.0282219155482935	0.0031815816521385543
0.7	1.0527786029324615	0.003965590025790666
0.8	1.088983985079174	0.004420068777111143
0.9	1.1382721319032127	0.00442258087538705
1.0	1.2010818156270293	0.003989316460585757

□

Problem 12. *Carry out the steps of Problem 10, using the Adams-Bashforth Three-Step method. Use the Runge-Kutta method to compute* w_1 *and* w_2. $\boxed{C}$

Solution. The Adams-Bashforth Three-Step method is given by

$$x_{n+1} = x_n + \frac{k}{12}\left[23f_n - 16f_{n-1} + 5f_{n-2}\right].$$

We code this method in the following snippet, using $x_0 = 1$ and applying first RK4 to get x_1 and x_2:

```
#Adams-Bashforth Three-Step Method
function adam_b3(f::Function, t0::Float64, t1::Float64, t2::Float64, a::Float64, b::Float64, k::Float64)::Array
    mesh = collect(a:k:b)
    n = length(mesh)
    xval = zeros(n)
    xval[1] = t0
    xval[2] = t1
    xval[3] = t2
    for i = 4:n
        xval[i] = xval[i-1] + (k/12)*(23*f(mesh[i-1],xval[i-1]) - 16*f(mesh[i-2],xval[i-2])
                  + 5*f(mesh[i-3],xval[i-3]))
    end
    return xval
end
```

The results are given in the following table:

Time step	Approximation	Global Error
0.0	1.0	0.0
0.1	1.0000249995312636	1.562374674080047e-10
0.2	1.0003998416710735	1.5644920914326121e-9
0.3	1.001798622891266	0.00022229026754150283
0.4	1.0059309076492557	0.00042856374258870567
0.5	1.0147993358469196	0.0005876892645002574
0.6	1.0307519294935537	0.0006515677068783532
0.7	1.0561697059901778	0.0005744869680743836
0.8	1.093055066990189	0.0003489868660964124
0.9	1.1426635912205272	3.1121558072610966e-5
1.0	1.2053508004373559	0.00027966834974080257

□

Problem 13. *Compute the coefficients in a multi-step method of the form*

$$x_{n+1} = x_n + k[Af_n + Bf_{n-2} + Cf_{n-4}].$$

The formula should correctly integrate an equation $x' = f(x)$ when the right-hand side is of the form $f(t, x) = a + bt + ct^2$.

Solution. We need

$$\int_{t_n}^{t_{n+1}} f(t, x)\,\mathrm{d}t = k[Af_n + Bf_{n-2} + Cf_{n-4}]$$

to be exact for polynomials of degree ≤ 2; i.e., for polynomials

$$p_0(t) = 1, \qquad p_1(t) = t, \qquad p_2(t) = t(t+1).$$

Let, WLOG, $t_n = 0$ and $t_{n+1} = 1$. Then, from

$$\int_0^1 p_n(t)\,\mathrm{d}t = Ap_n(0) + Bp_n(-2) + Cp_n(-4),$$

and applying the method of undetermined coefficients, we get the following system:

$$\begin{aligned} A + \quad B + \quad C &= 1 \\ -4B - \quad 8C &= 1 \\ 2B + 12C &= \frac{5}{6}. \end{aligned}$$

The solution set is $\{A = 17/12,\ B = -7/12,\ C = 1/6\}$; thus we end up with the multi-step method

$$x_{n+1} = x_n + k\left[\frac{17}{12}f_n - \frac{7}{12}f_{n-2} + \frac{1}{6}f_{n-4}\right].$$

□

Problem 14. *Write and test the fourth-order Runge-Kutta procedure for solving the following system on the interval* $1 \leq t \leq 2$. *Use* $k = -0.01$ *(that is, t decreasing from 2 to 1).* $\boxed{C}$

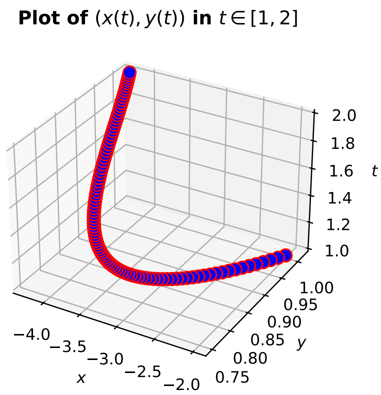
$$x' = x^{-2} + \log y + t^2,$$
$$y' = e^y - \cos x + (\sin t)x - (xy)^{-3},$$
$$x(2) = -2, \quad y(2) = 1.$$

Solution. The following code implements RK4 and solves the given system:

```julia
# RK4 for a first-order system of equations
function rk4_sys(f::Array, init::Array, a::Float64, b::Float64, k::Float64)::Array

    mesh = collect(a:k:b)
    m = length(f)       # number of equations
    n = length(mesh)    # number of steps

    sol = zeros(m,n)
    sol[:,1] = init
    for i in 2:n
        for j = 1:m
            k1 = f[j](mesh[i-1], sol[:,i-1]...)
            k2 = f[j](mesh[i-1] + (k/2), sol[:,i-1] .+ (k/2)*k1...)
            k3 = f[j](mesh[i-1] + (k/2), sol[:,i-1] .+ (k/2)*k2...)
            k4 = f[j](mesh[i-1] + k, sol[:,i-1] .+ k * k3...)
            sol[j,i] = sol[j,i-1] + (k/6)*(k1 + 2*k2 + 2*k3 + k4)
        end
    end
    return sol
end

# define the given functions
func1(t,x,y) = x^(-2) + log(y) + t^2
func2(t,x,y) = exp(y) - cos(x) + x*sin(t) - (x*y)^(-3)
func_grid = [func1, func2]

solution = rk4_sys(func_grid, [-2.0, 1.0], 2.0, 1.0, -0.01)

# output data to plot in Matplotlib
using DelimitedFiles
writedlm( "output_data.csv",  solution, ',')
```

We now plot the results, as a parametric curve in $2D$, on Matplotlib:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

Z = pd.read_csv("~/Desktop/output_data.csv", header = None)
Z = Z.transpose()

ax = plt.figure().add_subplot(projection='3d')

t = np.linspace(1, 2, 101)
x = Z[0]
y = Z[1]

ax.plot(x, y, t, color='red', linestyle='dashed', marker='o', markerfacecolor='blue',
    markersize=8)

ax.set_xlabel(r'$x$')
ax.set_ylabel(r'$y$')
ax.set_zlabel(r'$t$')

plt.title(r'Plot of $(x(t), y(t))$ in $t \in [1,2]$', fontweight ='bold')
# plt.show()
plt.savefig('../../Figures/rk4_output.pdf')
```



$\square$

Problem 15. *Test the implicit midpoint method*

$$x_n - x_{n-1} = kf\left(t_{n-1} + \frac{1}{2}k, \frac{1}{2}(x_n - x_{n-1})\right)$$

on the equation $x' = \lambda x$, with $\lambda < 0$, to determine whether its performance will be good on stiff problems.

Solution. Letting $f(t, x) := \lambda x$, we have

$$\begin{aligned} x_n - x_{n-1} &= kf\left(t_{n-1} + \frac{1}{2}k, \frac{1}{2}(x_n - x_{n-1})\right) \\ &= \frac{k\lambda}{2}\left(x_n - x_{n-1}\right). \end{aligned}$$

In order for this equation to hold when $|\lambda| \gg 0$ (i.e., when the magnitude of λ is large), we would need to have either $x_n = x_{n-1}$, or a very small time-step $k = 2/\lambda \approx 0$. Neither situation is desired, of course. As $k \to 0$ the performance worsens, whereas if $x_n = x_{n-1}$ the algorithm gets stuck with same output for all time-steps. $\square$