Numerical Methods for Solving ODE's

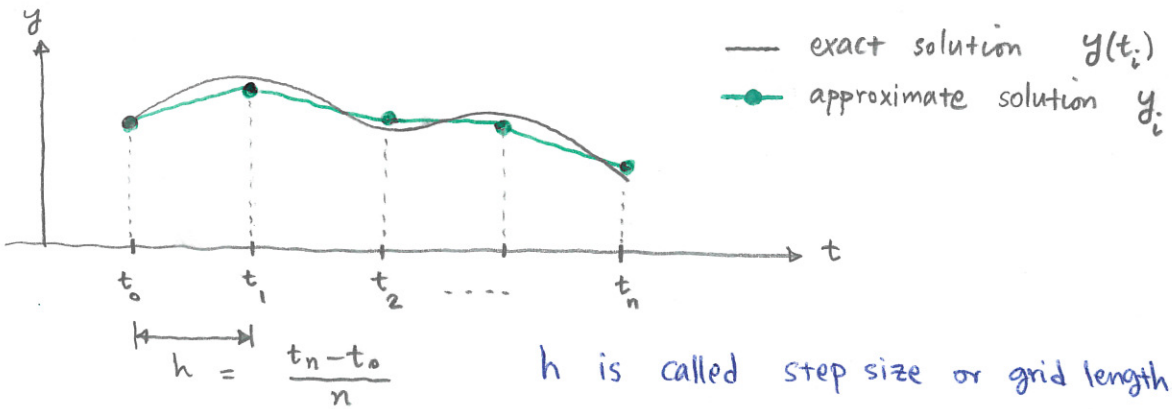
M. MOTAMED
MATH 316

So far we have studied analytical methods for solving ODE's. The diagram on page 2 summarizes the main analytical methods for different types of ODE's.

In many cases, for example when the ODE coefficients are variable or when the ODE is nonlinear, it is either extremely difficult or impossible to find the exact solution by an analytical method.

In such cases, we need to use numerical methods :

Instead of finding an exact expression for the solution (which may be either very complicated or impossible), we find an approximate solution on a grid of discrete points $\{t_i\}_{i=0}^{n} \in [t_0, t_n]$

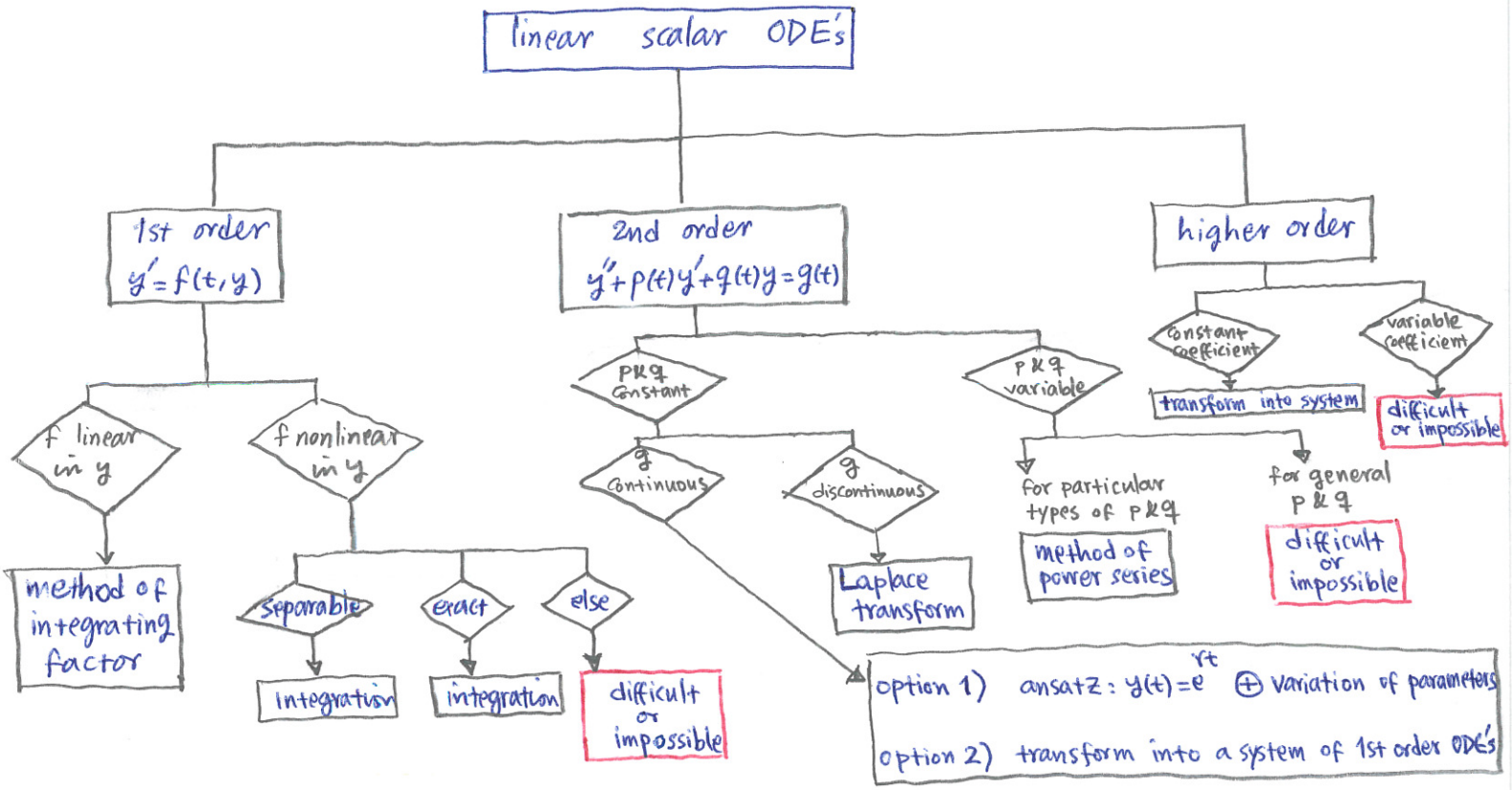


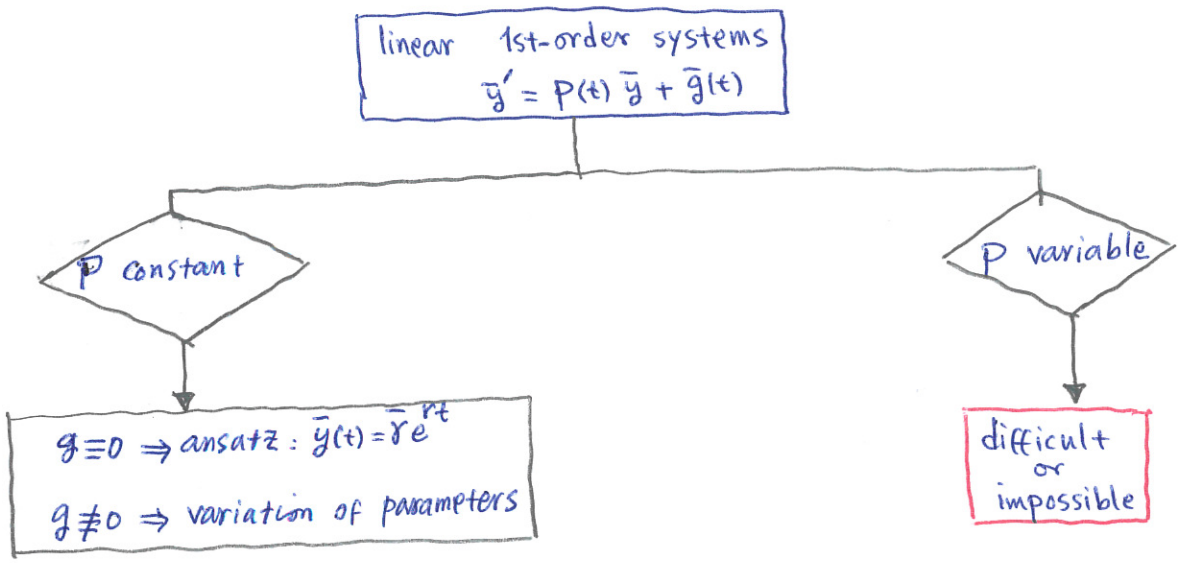
h is called step size or grid length

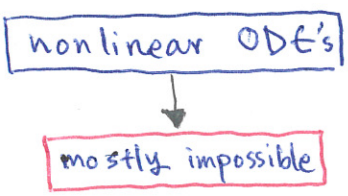
We want our approximate solution to have two properties:

- accuracy : the error in approximation $|y(t_i) - y_i|$ is small.
- convergence : the error approaches zero as $h \to 0$.

the diagram of analytical methods for solving ODE's :







Euler's Method

Euler's method is the simplest numerical method for solving ODE's.

It is based on Taylor expansion and is in the class of Taylor methods.

Consider the IVP:
$$\begin{cases} y' = f(t, y) & t \in [t_0, T] \\ y(t_0) = y_0 \end{cases}$$

and assume that f and $\partial_y f$ are continuous around (t_0, y_0). This assumption will guarantee that there exists a unique solution around t_0.

Taylor's theorem $\Rightarrow$ $y(t+h) = y(t) + h\, y'(t) + \frac{h^2}{2} y''(c)$, $\quad t < c < t+h$

If h is small, then h^2 is very small. Moreover, if $|y''(c)|$ is bounded, then we can neglect the term $\frac{h^2}{2} y''(c)$ and write:

$$\boxed{y(t+h) \approx y(t) + h \cdot f(t, y(t))} \qquad (*)$$

Now, if we discretize $[t_0, T]$ into $n+1$ evenly distributed grid points

$$t_0 < t_1 < t_2 < \cdots < t_n = T$$

and let y_i be an approximation for the exact solution $y(t_i)$:

$$y_i \approx y(t_i), \qquad i = 0, 1, 2, \cdots, n,$$

Then we can use $(*)$ and write Euler's algorithm:

$$\boxed{\begin{aligned} y_0 &= \text{given by IC} \\ y_{i+1} &= y_i + h \cdot f(t_i, y_i) \quad \text{for } i = 0, 1, 2, \cdots, n-1 \end{aligned}}$$

EX.1 Apply Euler's method to the IVP: $\begin{cases} y' = ty + t^3, & t \in [0,1] \\ y(0) = 1 \end{cases}$

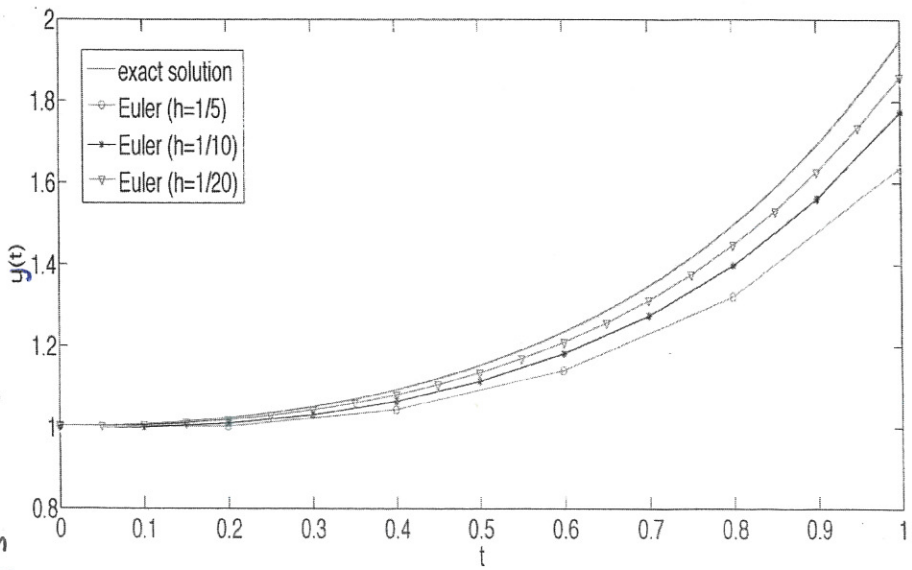
with step size $h = 0.2$.

$$h = \frac{b-a}{n} \Rightarrow n = \frac{b-a}{h} = \frac{1-0}{0.2} = 5 \qquad f(t,y) = ty + t^3$$

step i	t_i	y_i	$f(t_i, u_i)$	$y_{i+1} = y_i + h f(t_i, y_i)$
0	0	1	0	1
1	0.2	1	0.2080	1.0416
2	0.4	1.0416	0.4806	1.1377
3	0.6	1.1377	0.8986	1.3175
4	0.8	1.3175	1.5660	1.6306
5	1	1.6306		

Note: The exact solution is $\boxed{y(t) = 3e^{t^2/2} - t^2 - 2}$.

The following figure shows the exact solution $y(t)$ on $t \in [0,1]$ and the approximate solutions obtained by Euler's method with $h = 0.2$, $h = 0.1$, $h = 0.05$.



We observe that as h decreases the corresponding approximation converges to the exact solution

See the matlab code Numerics_EX1.m

Accuracy of Euler's method

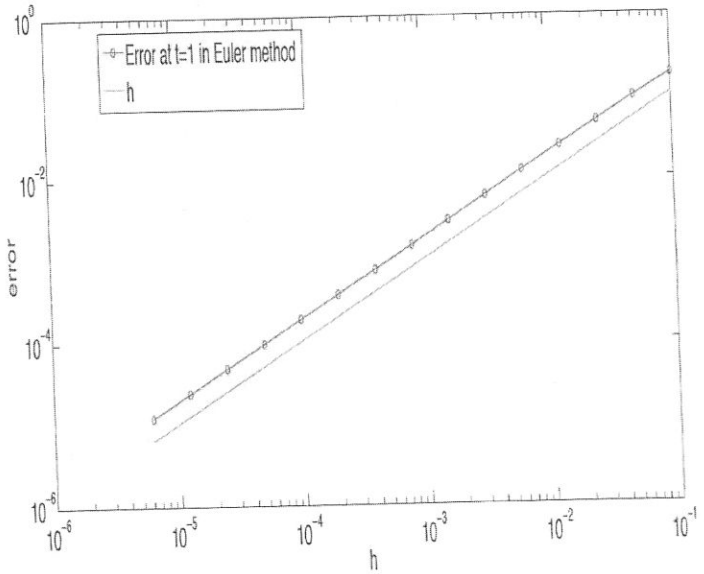
Consider the previous example. Since the exact solution is known, we can compute the error in the approximate solutions obtained with different step sizes. Let us compute the error at $t=1$:

exact solution at $t=1$: $y(1) = 3\sqrt{e} - 3$

Denote the approximate solution at $t=1$ obtained with step size h by $\tilde{y}_h(1)$.

$\Rightarrow$ error at $t=1$: $\varepsilon_h(1) = \left| y(1) - \tilde{y}_h(1) \right|$.

The following figure shows $\varepsilon_h(1)$ for different h values:



observation:
The error is proportional to h.

$$\boxed{\varepsilon_h(1) = O(h)}$$

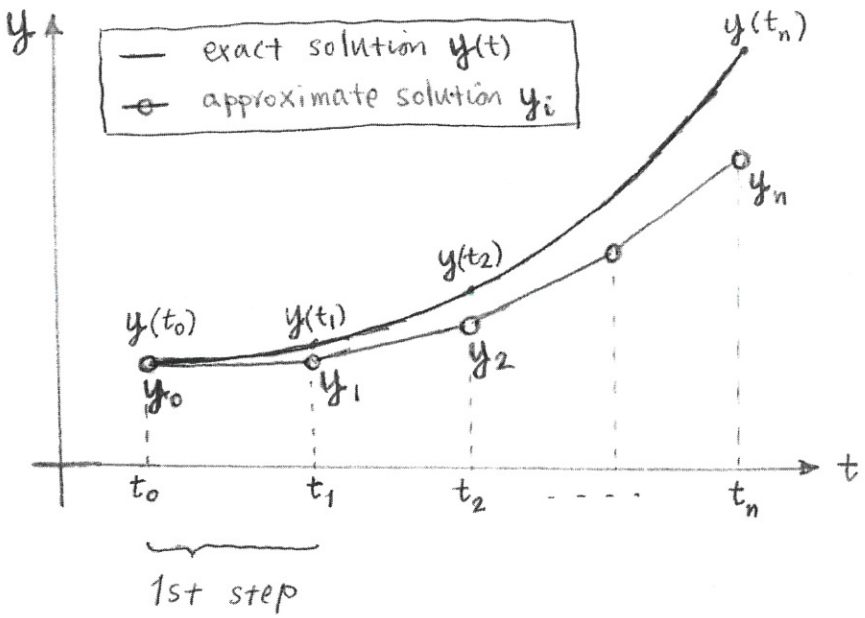
The error decreases linearly with h.

This suggests that Euler's method is a first-order accurate method.

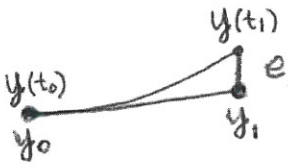
We will next do error analysis for Euler's method to understand why the error is proportional to h

Error Analysis for Euler's method

Consider the 1st step of Euler's method when solving $\begin{cases} y' = f(t,y), & t \in [t_0, T] \\ y(a) = y_0 \end{cases}$



1st step



$e_1 = |y_1 - y(t_1)|$

$\begin{cases} y' = f(t,y), & t \in [t_0, t_1] \\ y(t_0) = y_0 \end{cases}$

e_1 is a local error, i.e. the error made in only one step of the algorithm. It is the local error for step 1 of the algorithm.

By Taylor's thm: $y(t_1) = y(t_0+h) = \underbrace{y(t_0)}_{y_0} + h \underbrace{y'(t_0)}_{f(t_0, y(t_0)) = f(t_0, y_0)} + \frac{h^2}{2} y''(c)$ where $t_0 < c < t_1$

$$\Rightarrow y(t_1) = y_0 + h f(t_0, y_0) + \frac{h^2}{2} y''(c) \qquad (1)$$

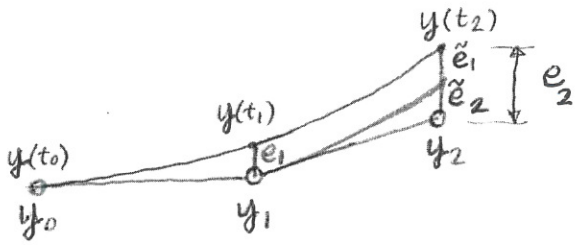
Now, by Euler's method: $y_1 = y_0 + h f(t_0, y_0)$ (2)

(1) & (2) $\Rightarrow e_1 = |y_1 - y(t_1)| = \frac{h^2}{2} |y''(c)|$ $\xrightarrow{\text{If } y \in C^2([t_0,T]) \Rightarrow |y''| \le M < \infty}$ $\boxed{e_1 \le \frac{M}{2} h^2}$

Hence, the local error in step 1 is proportional to h^2.

Now consider the 2nd step of Euler's method:

2nd step



$$\begin{cases} y' = f(t,y), & t \in [t_1, t_2] \\ y(t_1) = y_1 \end{cases}$$

This time, the local error for step 2 is $\tilde{e}_2$, the difference between the exact and approximate solutions of the above one-step IVP.

Similar to step 1, we can show that $\tilde{e}_2 \le \frac{M}{2} h^2$. Therefore the local error in step 2 is also proportional to h^2.

However, $\tilde{e}_2$ is not the total error in step 2.

The total error in step 2 is $e_2 = |y(t_2) - y_2|$ → (it is also called global error in step 2)

$$e_2 = |y(t_2) - y_2| = \tilde{e}_2 + \tilde{e}_1$$

Since f and $\frac{\partial f}{\partial y}$ are continuous, then we can show $\tilde{e}_1 = c_1 \cdot e_1$ (where $0 < c_1 < \infty$)

$$\Rightarrow \quad e_2 = \tilde{e}_2 + c_1 e_1 \le M_1 h^2 + M_2 h^2 \quad (\text{where } 0 < M_1, M_2 < \infty)$$

If we continue until the last step (the n-th step), the total/global error is:

$$e_n = |y(t_n) - y_n| \le M_1 h^2 + M_2 h^2 + \cdots + M_n h^2$$

$$\{ \text{if } \tilde{M} = \max(M_1, M_2, \cdots, M_n) \} \le n \tilde{M} h^2 = \frac{T - t_0}{h} \tilde{M} h^2 = (T - t_0) \tilde{M} h = O(h)$$

Conclusion: Although the local error (or the one-step error) in each single step of Euler's method is $O(h^2)$, the total error (or the global error) is $O(h)$. Therefore, Euler's method is a 1st-order accurate method. This phenomenon is called error accumulation.

Runge-Kutta (RK) Methods

RK methods are popular and powerful numerical methods for solving IVP's:

$$\begin{cases} y'(t) = f(t, y(t)) & t \in [t_0, T] \\ y(t_0) = y_0 \end{cases}$$

Euler's method is the simplest and least accurate RK method.

More accurate RK methods are obtained using higher order Taylor expansions involving higher derivatives of y, such as $y'', y''', \ldots$

We will study two RK methods:

RK2: a 2nd-order accurate numerical method

RK4: a 4th-order accurate numerical method

Remark: Euler's method, which can be considered as the 1st-order accurate RK method, is sometimes referred to as **RK1**.

RK2

Recall the derivation of RK1, where we take only two terms in Taylor's expansion:

$$y(t+h) = \underbrace{y(t) + h \cdot y'(t)}_{\text{two terms}} + \mathcal{O}(h^3)$$

To derive RK2, we take three terms in Taylor's expansion:

$$y(t+h) = \underbrace{y(t) + h\, y'(t) + \frac{h^2}{2} y''(t)}_{\text{three terms}} + \mathcal{O}(h^3) \qquad (*)$$

But, how to find $y''(t)$? We only know $y'(t) = f(t, y(t))$.

Let us write a two-term Taylor's expansion for y':

$$y'(t+h) = y'(t) + h\, y''(t) + \mathcal{O}(h^2)$$

$$\Rightarrow h\, y''(t) = y'(t+h) - y'(t) + \mathcal{O}(h^2)$$

$$\Rightarrow \frac{h^2}{2} y''(t) = \frac{h}{2} y'(t+h) - \frac{h}{2} y'(t) + \mathcal{O}(h^3)$$

Then $(*)$ can be written as:

$$y(t+h) = y(t) + \frac{h}{2} y'(t) + \frac{h}{2} y'(t+h) + \mathcal{O}(h^3)$$

where $\begin{cases} y'(t) = f(t, y(t)) \\ y'(t+h) = f(t+h,\ y(t+h)) \end{cases}$

This means:

$$y(t+h) = y(t) + \frac{h}{2} f(t, y(t)) + \frac{h}{2} f(t+h, y(t+h)) + O(h^3)$$

We aim at finding a formula that gives $y(t+h)$ in terms of $y(t)$.

The above formula contains $y(t+h)$ in the right-hand-side.

We use Taylor's formula once more: $y(t+h) = y(t) + h \cdot f(t, y(t)) + O(h^2)$

Then we get:

$$y(t+h) = y(t) + \frac{h}{2} f(t, y(t)) + \frac{h}{2} f(t+h, y(t) + h \cdot f(t, y(t)) + O(h^2)) + O(h^3)$$

Since $f(t, Y + O(h^2)) = f(t, Y) + O(h^2)$, we will obtain:

$$y(t+h) = y(t) + \frac{h}{2} f(t, y(t)) + \frac{h}{2} f(t+h, y(t) + h f(t, y(t))) + O(h^3)$$

Therefore, the RK2 method reads:

y_0 = given by IC at t_0 ← RK2 pseudocode

for $i = 0, 1, 2, \ldots, n-1$

  $K1 = f(t_i, y_i)$ ← RK2 algorithm

  $K2 = f(t_i + h, y_i + h \cdot K1)$

  $y_{i+1} = y_i + \frac{h}{2}(K1 + K2)$

  $t_i = t_i + h$

end

The local error in RK2 is $O(h^3)$, and hence RK2 has a global error $O(h^2)$. Therefore, RK2 is a 2nd-order accurate numerical method. It is also referred to as the improved Euler's method.

RK 4

In a similar way, by taking five terms in Taylor's expansion and using Taylor's formula several times, we can show that:

$$y(t+h) = y(t) + \frac{h}{6}\left[f(t,y(t)) + 4\,f\left(t+\tfrac{h}{2}, y(t+\tfrac{h}{2})\right) + f(t+h, y(t+h))\right] + O(h^5)$$

After some work, we obtain RK4 method:

RK4 pseudocode

$y_0 =$ given by IC at t_0

for $i = 0, 1, 2, \cdots, n-1$

RK4 algorithm

$K1 = f(t_i, y_i)$

$K2 = f\left(t_i + \frac{h}{2}, y_i + \frac{h}{2} \cdot K1\right)$

$K3 = f\left(t_i + \frac{h}{2}, y_i + \frac{h}{2} \cdot K2\right)$

$K4 = f(t_i + h, y_i + h \cdot K3)$

$y_{i+1} = y_i + \frac{h}{6}(K1 + 2K2 + 2K3 + K4)$

$t_i = t_i + h$

end

The local error in RK4 is $O(h^5)$, and hence RK4 has a global error $O(h^4)$. Therefore, RK4 is a 4th-order accurate numerical method. RK4 is one of the most popular numerical methods for solving IVP's.

Geometrical interpretation

RK1 uses the slope of $y(t)$ at t_i : $y_i' = f(t_i, y_i)$

RK2 uses the slope of $y(t)$ at t_i and t_{i+1} :

$$\begin{cases} K1: \text{ slope at } t_i \\ K2: \text{ slope at } t_{i+1} \end{cases}$$

RK4 uses the slope of $y(t)$ at t_i, t_{i+1}, and $t_{i+\frac{1}{2}} = \dfrac{t_i + t_{i+1}}{2}$

$$\begin{cases} K1: \text{ slope at } t_i \\ \dfrac{K2+K3}{2}: \text{ slope at } t_{i+\frac{1}{2}} \\ K4: \text{ slope at } t_{i+1} \end{cases}$$

The more information about $y(t)$ a method uses, the more accurate the method will be.

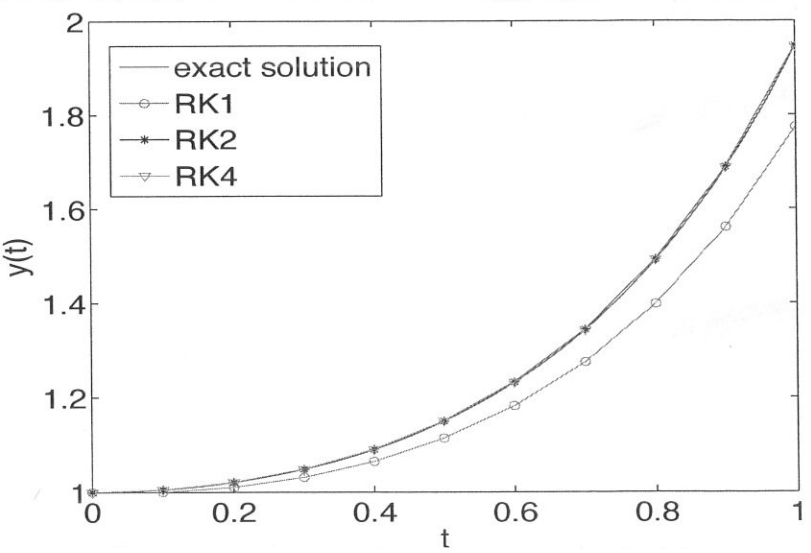
EX.2 Consider the same IVP as in EX.1 :

$$\begin{cases} y' = ty + t^3, & t \in [0,1] \\ y(0) = 1 \end{cases}$$

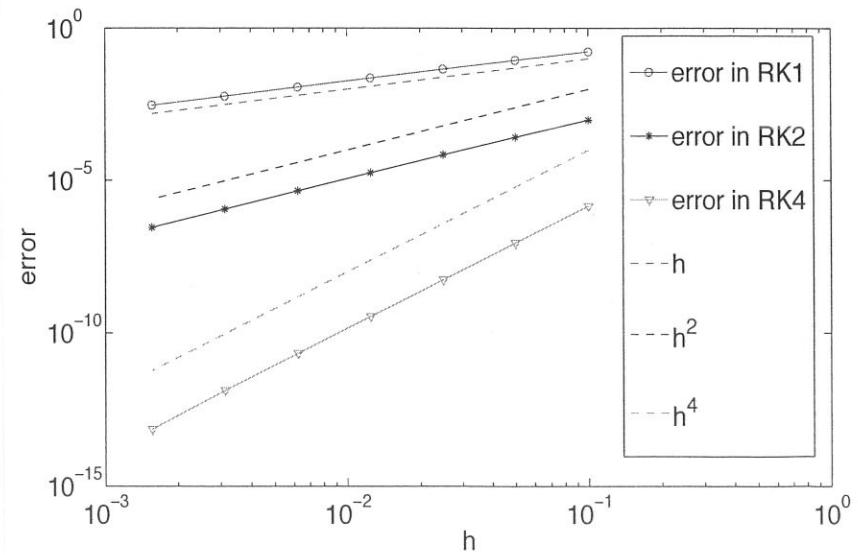
Implement RK2 and RK4 in MATLAB and numerically study the error at $t=1$ in terms of step size h. Compare and discuss.

See the Matlab code Numerics_EX2.m

with the same $h = 0.1$, we observe that RK2 and RK4 generate more accurate numerical solutions than RK1.



The plot verifies that the global error in RK1, RK2, and RK4 is $O(h)$, $O(h^2)$, and $O(h^4)$ respectively.



ODE systems

Consider the general form of a 1st-order system of ODE's:

$$
\begin{aligned}
y_1'(t) &= f_1(t, y_1, \cdots, y_m) \\
y_2'(t) &= f_2(t, y_1, \cdots, y_m) \qquad\qquad t \in [t_0, T] \\
&\vdots \\
y_m'(t) &= f_m(t, y_1, \cdots, y_m)
\end{aligned}
$$

with the initial condition:

$$
\begin{aligned}
y_1(t_0) &= y_{10} \\
y_2(t_0) &= y_{20} \\
&\vdots \\
y_m(t_0) &= y_{m0}
\end{aligned}
$$

If we set $\bar{y} = (y_1, \cdots, y_m)^T$ and $\bar{f}(t, \bar{y}) = (f_1, \cdots, f_m)^T$, then the IVP reads:

$$
\begin{cases}
\bar{y}'(t) = \bar{f}(t, \bar{y}(t)), & t \in [t_0, T] \\
\bar{y}(t_0) = \bar{y}_0
\end{cases}
$$

Note that $\bar{f}$ can be linear or nonlinear in $\bar{y}$.

We can easily extent RK1, RK2, and RK4 that we derived for scalar IVP's to solve IVP systems.

EX.3 Consider the following IVP system:

$$\bar{y}' = P\,\bar{y} \quad \text{with} \quad P = \begin{pmatrix} -\frac{1}{2} & 1 \\ -1 & -\frac{1}{2} \end{pmatrix} \quad \text{and IC:} \quad \bar{y}(0) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

It can also be written as:

$$\begin{cases} y_1' = -\frac{1}{2} y_1 + y_2 = f_1(t, y_1, y_2) \\ y_2' = -y_1 - \frac{1}{2} y_2 = f_2(t, y_1, y_2) \end{cases}$$

NOTE: We studied and solved this IVP in EX.12 in chapter 7.

The exact solution is $\begin{cases} y_1(t) = (\cos t + \sin t)\, e^{-t/2} \\ y_2(t) = (\cos t - \sin t)\, e^{-t/2} \end{cases}$

An easy approach to apply RK1, RK2, or RK4 to the IVP systems is to write them in the form:

$$\bar{y}' = \bar{f}(t, \bar{y}) \quad \text{where here, } \bar{y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \text{ and } \bar{f} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}.$$

RK1

$\bar{y}_0$ = given by IC

$\bar{K} = \bar{f}(t_i, \bar{y}_i)$

$\bar{y}_{i+1} = \bar{y}_i + h \cdot \bar{K}$

RK2

$\bar{y}_0$ = given by IC

$\bar{K1} = \bar{f}(t_i, \bar{y}_i)$

$\bar{K2} = \bar{f}(t_i + h, \bar{y}_i + h\bar{K1})$

$\bar{y}_{i+1} = \bar{y}_i + \frac{h}{2}(\bar{K1} + \bar{K2})$

RK4

$\bar{y}_0$ = given by IC

$\bar{K1} = \bar{f}(t_i, \bar{y}_i)$

$\bar{K2} = \bar{f}(t_i + \frac{h}{2}, \bar{y}_i + \frac{h}{2}\bar{K1})$

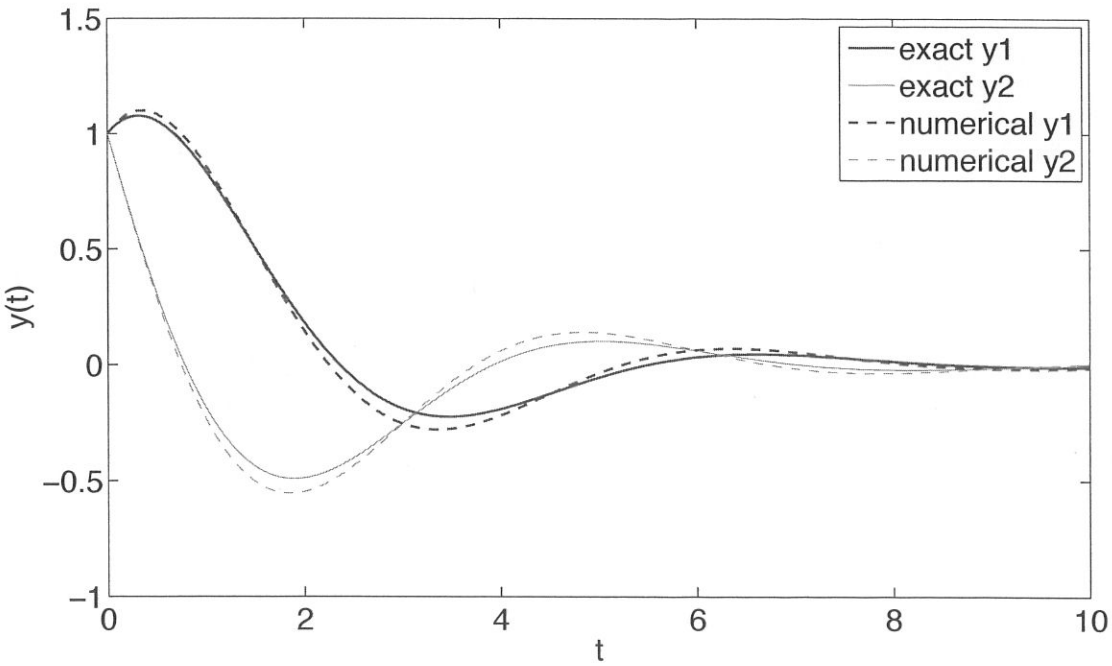
$\bar{K3} = \bar{f}(t_i + \frac{h}{2}, \bar{y}_i + \frac{h}{2}\bar{K2})$

$\bar{K4} = \bar{f}(t_i + h, \bar{y}_i + h\bar{K3})$

$\bar{y}_{i+1} = \bar{y}_i + \frac{h}{6}(\bar{K1} + 2\bar{K2} + 2\bar{K3} + \bar{K4})$

See the Matlab code Numerics_EX3.m.

The following figure shows the exact solution $\bar{y} = \begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix}$ and the numerical solution obtained by RK1 with $h=0.1$.



Higher order equations

A single ODE of higher order can be converted to a system of first-order ODE's:

Consider a 2nd-order ODE: $y'' + a\,y' + b\,t\,y = c$ (1)

where a, b, c are constant numbers.

First, we introduce the new variables y_1 and y_2 by setting

$$\begin{cases} y_1 = y \\ y_2 = y' \end{cases} \qquad (2)$$

Then, from (2) $\Rightarrow$ $\quad y_1' = y_2$

And from (1) $\Rightarrow$ $\quad y_2' = -a\,y_2 - b\,t\,y_1 + c$

This is a system of two first-order ODE's.

EX.4 Write the algorithm for Euler's method that solves the following IVP $\longrightarrow$

$$\begin{cases} y''' = -y''^2 - y\,y' + \sin t \\ y(0) = 1 \\ y'(0) = 0 \\ y''(0) = 2 \end{cases} \qquad t \in [0,2]$$

NOTE: It is a 3rd order nonlinear ODE.

First, we convert the third-order ODE into a system of three first-order ODE's

Set $\begin{cases} y_1 = y \\ y_2 = y' \\ y_3 = y'' \end{cases}$ $\Rightarrow$

$$\begin{cases} y_1' = y_2 = f_1 \\ y_2' = y_3 = f_2 \\ y_3' = -y_3^2 - y_1\,y_2 + \sin t = f_3 \end{cases}$$

From the 3rd order ODE $\Rightarrow$

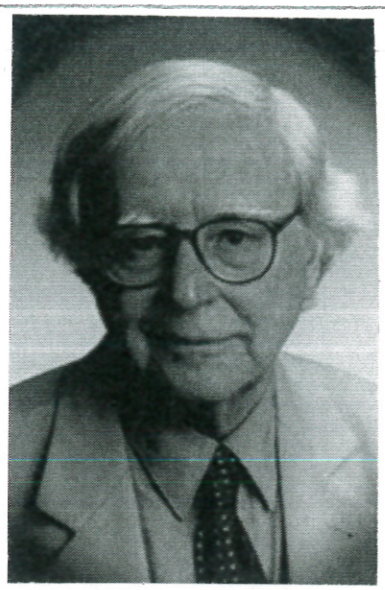
EM $\Rightarrow$

$$\begin{cases} y_{1,i+1} = y_{1,i} + h \cdot y_{2,i} \\ y_{2,i+1} = y_{2,i} + h \cdot y_{3,i} \\ y_{3,i+1} = y_{3,i} + h \cdot \left(-y_{3,i}^2 - y_{1,i}\,y_{2,i} + \sin t_i \right) \end{cases}$$

Initial conditions: $y_1(0) = 1$, $y_2(0) = 0$, $y_3(0) = 2$

See Numerics_EX4.m

Convergence of numerical methods



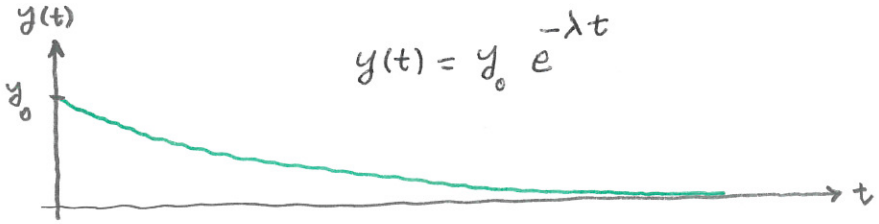
Germund Dahlquist
1925 – 2005

Consider the Dahlquist model problem:

$$\begin{cases} y'(t) = -\lambda\, y(t), & \lambda > 0, \quad t \in [0,T] \\ y(0) = y_0 \end{cases}$$

The exact solution can easily be found:

$$y(t) = y_0\, e^{-\lambda t}$$



We now use Euler's method to find an approximate solution:

$$y_{i+1} = y_i + h\,(-\lambda\, y_i) = (1 - h\lambda)\, y_i \qquad i = 0, 1, 2, \ldots$$

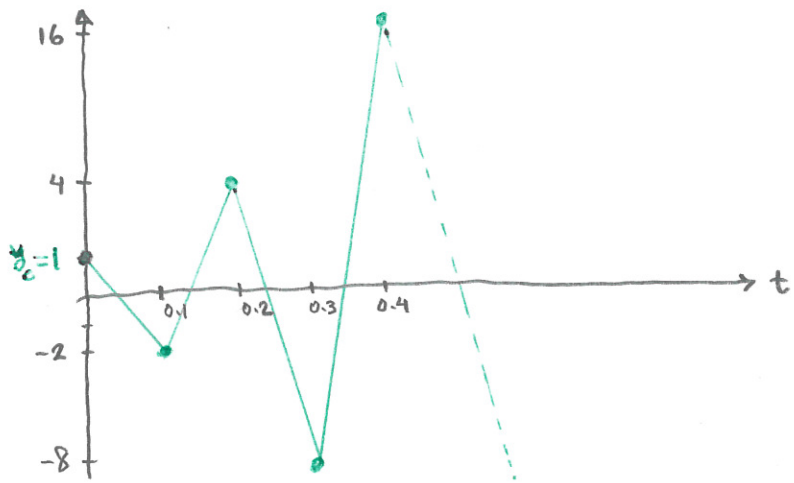
For example, let $\lambda = 30$ and $y_0 = 1$. Take $h = 0.1$. $\Rightarrow 1 - h\lambda = -2$

$i = 0$: $y_1 = -2\, y_0 = -2$

$i = 1$: $y_2 = -2\, y_1 = 4$

$i = 2$: $y_3 = -2\, y_2 = -8$

$i = 3$: $y_4 = -2\, y_3 = 16$



We observe that the numerical solution does not approximate the true solution, which exponentially decays. Instead of an exponential decay, the numerical solution has a growing oscillatory behavior !!

What went wrong in the previous example?

Every numerical method such as Euler's method must have two properties.
1) convergent 2) accurate

Let $\tilde{y}_h$ be the numerical solution with step size h.

Convergence: tells us that the numerical solution $\tilde{y}_h$ approaches the exact solution y as the step size h goes to zero

$$\lim_{h \to 0_+} \| y - \tilde{y}_h \| = 0$$

accuracy: tells us how well the numerical solution $\tilde{y}_h$ approximates the exact solution. For Euler's method, the global error is $\mathcal{O}(h)$. Hence, Euler's method has accuracy of order 1.

Accuracy is a necessary condition for convergence, but not sufficient!

For a method to be convergent, it must be both accurate and stable.

In the previous example, we chose a wrong step size ($h = 0.1$) for which the Euler's method was not stable. Hence, it did not converge to the correct solution.

Let us consider the example again: $y_{i+1} = (1 - h\lambda)\, y_i$

$$\Rightarrow \frac{y_{i+1}}{y_i} = 1 - h\lambda \quad \Rightarrow \quad G_i = \left| \frac{y_{i+1}}{y_i} \right| = |1 - h\lambda|$$

G_i is called the amplification factor at time step i.

For Euler's method: $\boxed{G_i = |1 - h\lambda|}$

For Euler's method, the amplification factor is $G_i = |1 - h\lambda|$.

If the amplification factor is less than 1, then we are guaranteed that the numerical sol. will not grow without bound $\left(G_i < 1 \Rightarrow |u_{i+1}| < |u_i|\right)$.

In this case, we say the numerical solution will be stable.

$$G_i < 1 \quad \Rightarrow \quad |1 - h\lambda| < 1 \quad \Rightarrow \quad -1 < 1 - h\lambda < 1 \quad \Rightarrow \quad \boxed{0 < \lambda h < 2}$$

Condition for stability

This translates to a step size restriction for stability: We need $0 < h < \frac{2}{\lambda}$.

In the previous example $h = 0.1$ and $\frac{2}{\lambda} = \frac{2}{30} \approx 0.0667 \Rightarrow h > \frac{2}{\lambda} \Rightarrow$ Unstable

For instance, if we take $h = 0.05$, the approximate sol. will converge to the true solution.

Conclusion: For Dahlquist model problem with $\lambda > 0$, Euler's method is stable only when the step size h satisfies the stability condition $0 < h < \frac{2}{\lambda}$.

This conditional stability is referred to as partial stability. Thus, Euler's method is partially stable.

Def. A method is called partially stable if when applied to Dahlquist model problem with $\lambda > 0$, the corresponding numerical solution is stable only for some values of $h\lambda$.

So far we have seen that:

> For an ODE solver, such as a Runge-Kutta (RK) method (Euler's Method (RK1), RK2, RK4), we have
>
> convergence = accuracy + stability

In order to investigate the stability of a numerical ODE solver, we consider Dahlquist model problem:

$$\begin{cases} y'(t) = -\lambda\, y(t) & t \in [0,T] \\ y(0) = y_0 \end{cases}$$

where $\lambda > 0$. We then obtain the amplification factor of the numerical solver: $G_i = \left| \frac{y_{i+1}}{y_i} \right|$. The method will then be stable if $G_i < 1$.

Region of Stability

It is more common to let λ be a complex number:

$$\lambda = \lambda_R + i\,\lambda_I \qquad \text{with} \qquad \lambda_R > 0.$$

Because, in practice we usually solve systems of ODE's, and λ represents the eigenvalues of the coefficient matrix, which may be complex even if the matrix is real.

Def. The region in the complex λh-plane where the numerical solver is stable is called the region of stability (or the region of absolute stability) of the solver.

EX.5 Find and sketch the region of absolute stability for Euler's method.

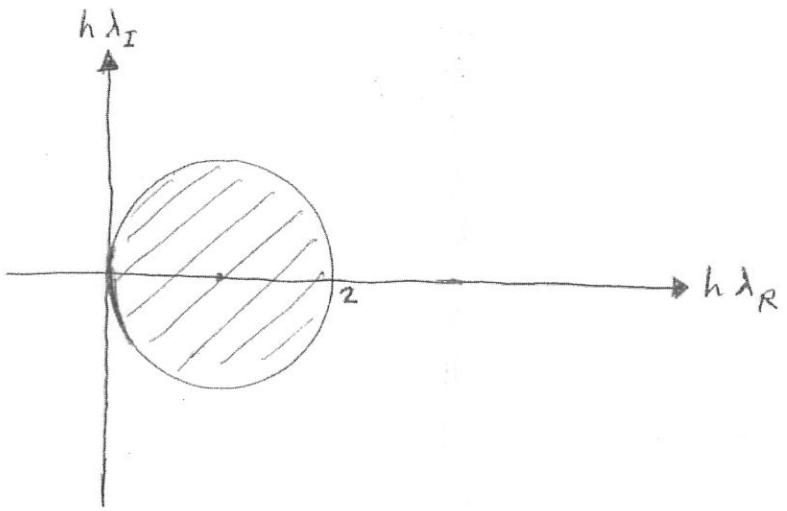
Dahlquist model problem: $y' = -\lambda y$, $Re(\lambda) > 0$

Euler's method: $y_{j+1} = y_j + h(-\lambda y_j) = (1 - h\lambda) y_j$

$$\Rightarrow G_j = \left| \frac{y_{j+1}}{y_j} \right| = |1 - h\lambda| = |1 - h(\lambda_R + i\lambda_I)|$$

$$= |(1 - h\lambda_R) - ih\lambda_I| = \sqrt{(1 - h\lambda_R)^2 + (h\lambda_I)^2} < 1$$

$\Rightarrow$ The region of stability of Euler's method is the region inside the circle with radius 1: $(1 - h\lambda_R)^2 + (h\lambda_I)^2 = 1$



EX.6 Find and plot the stability region for RK2.

Dahlquist model: $y' = -\lambda y$
where $\lambda = \lambda_R + i\lambda_I$, $\lambda_R > 0$

RK2

$$\begin{cases} K_1 = -\lambda y_j \\ K_2 = -\lambda (y_j - h\lambda y_j) \\ y_{j+1} = y_j + \frac{h}{2}(-2\lambda y_j + h\lambda^2 y_j) = y_j \left(1 - h\lambda + \frac{1}{2}(h\lambda)^2\right) \end{cases}$$

$$\Rightarrow G_j = \left| \frac{y_{j+1}}{y_j} \right| = \left| 1 - (h\lambda) + \frac{1}{2}(h\lambda)^2 \right| < 1$$

Now, how to plot the stability region?

We want to plot all $z = h\lambda$ values in the complex plane for which

$$\left| 1 - z + \frac{1}{2} z^2 \right| = 1.$$

This means that $\forall\, \phi \in [0, 2\pi]$ we need to find all z such that

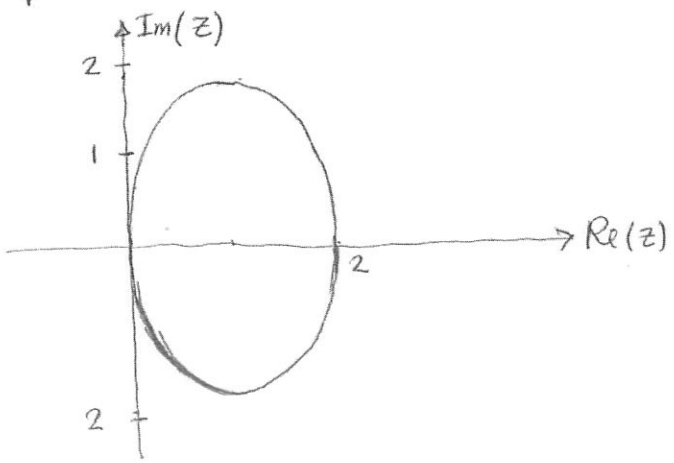
$$1 - z + \frac{1}{2} z^2 = e^{i\phi}.$$

Therefore, we plot all solutions to the quadratic equation

$$\frac{1}{2} z^2 - z + (1 - e^{i\phi}) = 0$$

There are two solutions: $z_{1,2} = 1 \pm \sqrt{2e^{i\phi} - 1}$

```
Phi = linspace(0, 2*pi, 1000);
M = exp(1i*Phi);
z1 = 1 + sqrt(2*M-1);
z2 = 1 - sqrt(2*M-1);
plot(z1)
hold on
plot(z2)
```



EX. 7 Find and plot the stability region for RK4.

$$K_1 = -\lambda u_j$$

$$K_2 = -\lambda\left(y_j - \frac{h}{2}\lambda y_j\right) = \left(-\lambda + \frac{1}{2}h\lambda^2\right) y_j$$

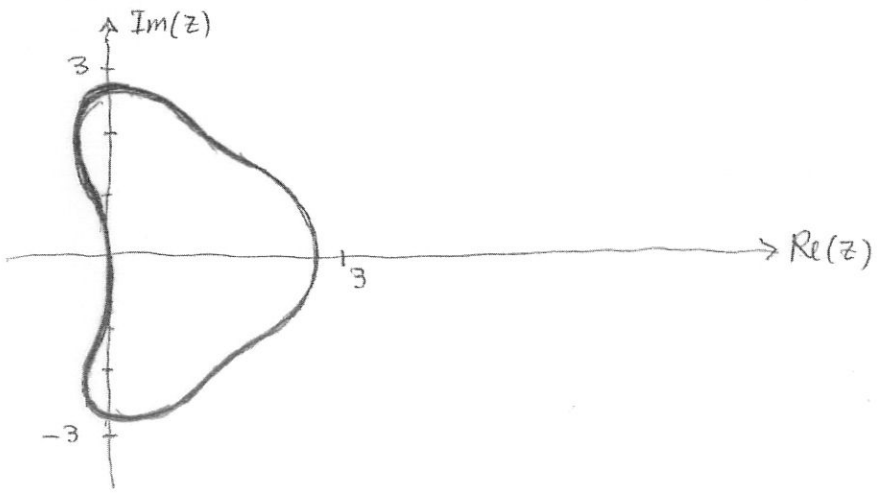
$$K_3 = -\lambda\left(y_j + \frac{h}{2}\left(-\lambda + \frac{1}{2}h\lambda^2\right) y_j\right) = \left[-\lambda + \frac{1}{2}\lambda^2 h - \frac{1}{4}\lambda^3 h^2\right] y_j$$

$$K_4 = -\lambda\left(y_j + h\left[-\lambda + \frac{1}{2}\lambda^2 h - \frac{1}{4}\lambda^3 h^2\right] y_j\right) = \left[-\lambda + \lambda^2 h - \frac{1}{2}\lambda^3 h^2 + \frac{1}{4}\lambda^4 h^3\right] y_j$$

$$y_{j+1} = y_j + \frac{h}{6}\left[-\lambda - 2\lambda + h\lambda^2 - 2\lambda + \lambda^2 h - \frac{1}{2}\lambda^3 h^2 - \lambda + \lambda^2 h - \frac{1}{2}\lambda^3 h^2 + \frac{1}{4}\lambda^4 h^3\right] y_j$$

$$\Rightarrow G_j = \left|\frac{u_{j+1}}{u_j}\right| = \left|1 - (h\lambda) + \frac{1}{2}(h\lambda)^2 - \frac{1}{6}(h\lambda)^3 + \frac{1}{24}(\lambda h)^4\right|$$

We want to plot $z = h\lambda$ s.t. $\left|1 - z + \frac{1}{2}z^2 - \frac{1}{6}z^3 + \frac{1}{24}z^4\right| = 1$

This time, we take another approach: We plot $G_j(z)$ for all possible z values and then plot the 1-contour of G_j:

```
[x,y] = meshgrid( linspace(-4,4,1000), linspace(-4,4,1000) );
z = x + 1i * y;
contour( x, y, abs(1 - z + (1/2)*z.^2 - (1/6)*z.^3 + (1/24)*z.^4), [1, 1], 'k-' )
```