Name: Budhrani, Laksh Anil

Course: B280

Assignment: Final Project

Topic: Ordinary differential equations (ODE) with initial value conditions

1. **Introduction – Background and Definitions**

   A first order ODE has the form:
   $dy/dx = f(x, y)$ with the initial condition: $y(x1) = y1$

   The **solution** is the function y(x) that satisfies the equation and the initial condition.

   A **differential equation** is an equation that contains derivatives of an unknown function. The solution of the equation is the function that satisfies the differential equation.

   A differential equation that has one independent variable is called an **ordinary differential equation** (ODE).

   A **first-order ODE** involves the first derivative of the dependent variable with respect to the independent variable.
   For example, if x is the independent variable and y is the dependent variable, the equation has combinations of the variables x, y, and dy/dx. A first-order ODE is linear, if it is a linear function of y and dy/dx (it can be a nonlinear function of x).

2. **Importance and Applications**

   ODEs are used in a lot of mathematical models used in various fields like mechanics, physics, chemistry, economic predictive models etc.
   They are useful in all the distributions wherein there is a relation of independent variable as a function of dependent variable.
   For example –
   a. in fluid mechanics (Bernoulli equation, drainage of fluid in reservoirs, tanks, drainage of tapered funnels),
   b. in heat transfer mechanisms in solids, liquids and gases (Fouriers law, heat flux in space, heat spreaders in microelectronic coolers),
   c. in rigid body dynamics under influence of gravity (newtons laws, rocket launch, helicopter, free fall of parachutes),
   d. growth and decay problems in radiology,
   e. electrical circuits,
   f. sociology (population growth and fall along with resources predictive models),
   g. disease burden trends prediction based on independent environmental variables etc.

3. General Strategy

When we're dealing with first-order ordinary differential equations (ODEs), we have two main strategies to find the solution: analytical methods and numerical methods.

Analytical methods aim to find an exact mathematical expression that represents the function y(x) which satisfies the differential equation. This method works well for simpler ODEs, but for many equations, finding an analytical solution is not possible. This is because the equations might be too complex or do not have a known solution in terms of elementary functions.

That's where numerical methods come in handy. Numerical solution provides a set of discrete values which approximate the exact solution. We first determine solution at the initial point. Using the solution of initial point, we calculate solution at the second adjacent point. Thus, in a stepwise or incremental manner the solution process approximates the exact solution of the ODE.

4. Types of Methods

a. Single Step and Multi Step

The approach used can be single step or multi step.
In single step approach we determine the solution at x(i+1) based on the known solution at previous point i.e. x(i).
Whereas in a multistep approach we can use solutions at multiple previous points to determine the solution at x(i+1) and so this multistep approach tends to provide a better estimate for the trend of solution.

b. Explicit and Implicit

As for how to calculate solution at each step there are two methods – Explicit and implicit.
Explicit methods us explicit formulae to calculate value of dependent variable for next value of independent variable. The right-hand side of the equation consists of known quantities only.
$y_{i+1} = F(x_i, x_{i+1}, y_i)$ for all x_i, x_{i+1}, y_i are known qunatities.
For example, y2 (the second unknown value of the dependent variable) will be calculated by evaluating a function of x1, y1, x2 values of which will be already known to us.
Implicit methods on the other hand derive solutions based on unknown quantities and equations are nonlinear in general. The equation can be written as
$y_{i+1} = F(x_i, x_{i+1}, y_{i+1})$
So, the unknown quantity appears on both sides of the equation. The equation has to be solved numerically for y(i+1). Hence implicit methods require more effort at each step but give more accurate solutions as compared to explicit methods

5. Methods –

1. Euler's explicit method.

a. Strategy

It is a technique used to calculate numerical solution of a problem involving first-order ODE. It is a single-step method.
The general idea behind it is to increment the value of x in each step by some constant value and get the values of y based on the value of x, using function that gives us slope of y at a given x. We continue getting these points(y's) iteratively until we have achieved our desired accuracy or covered our domain.

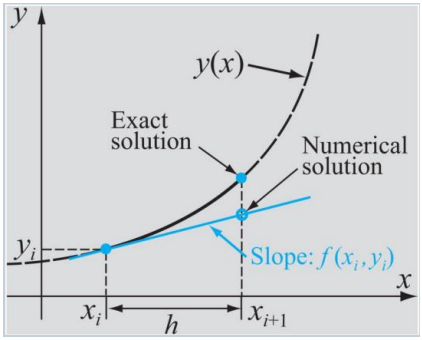
b. Big Picture

The idea is that we calculate the values of x and y's as follows –

$$x_{i+1} = x_i + h$$

$$y_{i+1} = y_i + Slope \cdot h$$

Here, h is a constant value that determined how close we want the values of x.
The reason for adding Slope*h with y_i is that it approximates the height with which y will be incrementing within the next iteration.
This is because
Slope(tan) = Opposite/Base
Therefore, Opposite = Slope* Base



As we can even see in the image above, using the slope and h, we can approximately calculate the value of y.

In euler explicit method, we have value of slopes as a function f that takes x and y as parameter and tells the value of slope at that point.
We can directly use that function to calculate the slope.

This is an iterative process where value of next term depends on values of previous terms. Each time we calculate the value of x and y, we will use that to calculate the values of our next x and y till we reach our desired tolerance or finish our domain.

Hence, we can replace now the general idea of euler explicit method with technical parameters.

Therefore, the expression for euler explicit method becomes –

$$x_{i+1} = x_i + h$$

$$y_{i+1} = y_i + f(x_i, y_i)h$$

c. Algorithm

Step 1: Start with an initial point (x0, y0).
Step 2: Choose a small step size h.
Step 3: Calculate the slope of the function at the current point, f(x0, y0).
Step 4: Estimate the next y-value using Euler's method formula: y1 = y0 + h * f(x0,y0).
Step 5: Update the current x-value: x1 = x0 + h.
Step 6: Repeat steps 3-5 for the next points until the desired range is covered.

d. Example

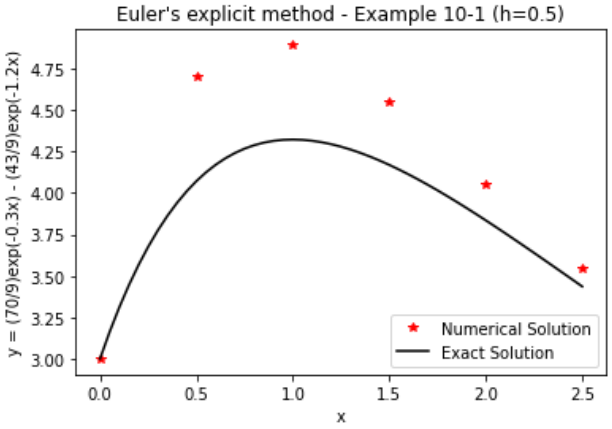
Use Euler's explicit method to solve the ODE dy/dx = - 1.2 y + 7 e ^(-0.3x) from x = 0 to x = 2.5 with the initial condition y = 3 at x = 0.

Soln.

When we use the above algorithm in python to find the solution, we get the following result –

```
In [1]: runfile('C:/Users/Laksh/Downloads/Euler_Explicit_Example10_1(1)
(1).py', wdir='C:/Users/Laksh/Downloads')
 i    x[i]    y[i]    yTrue[i]    err[i]

 1   0.0000   3.0000   3.0000     0.0000

 2   0.5000   4.7000   4.0723     -0.6277

 3   1.0000   4.8925   4.3229     -0.5696

 4   1.5000   4.5499   4.1696     -0.3803

 5   2.0000   4.0516   3.8351     -0.2165

 6   2.5000   3.5415   3.4361     -0.1054
```

Graphically, when we plot these values, we see that our method as closely approximated its solution to true solution values.



e. Pros and Cons of using this method

The biggest benefit of this method is that as it is less complex than other methods to find numerical solution, its computation cost is very less.

On the other hand, as it calculates the slope of the initial value of x, even though it is simple algorithm, it makes the approximation of numerical solution less accurate. This problem is solved by modified euler method and other numerical methods like midpoint method and runge-kutta method. We are going to discuss them later.

f. Error analysis

As we know, the total error is the sum of round-off error and truncation error. We get round-off errors due to computer's limit for calculations and storing number's while we get truncation error because of the approximate nature of algorithm. Additionally, the truncation errors in numerical methods consists of local truncation error and propagated truncation error.

The difference between numerical solution and true solution is the total error. Part of total error is local truncation error and the rest due to accumulation of truncation errors due to previous steps. We do not include round-off errors here. The total error due to truncation alone is called the global truncation error.

It was important to understand these terms before understanding the estimate for error for euler explicit method. We wont discuss these same terms again for other methods. If the definition for term is different in some case, only then the term will be defined again.

For euler explicit method,

The local truncation error of Euler's explicit method is $0(h^2)$.

The global truncation error is 0(h).

Note:

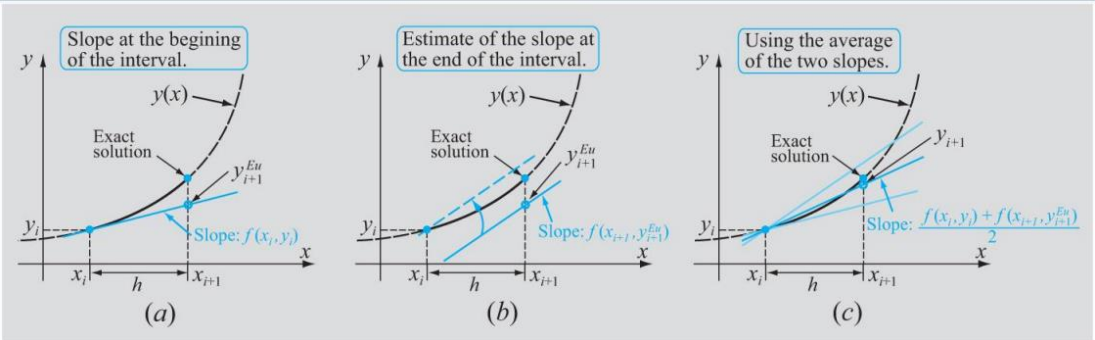
The truncation error can be reduced by using smaller h (step size). However, if h becomes too small such that round-off errors become significant, the total error might increase.

2. Modified Euler's method.

a. Strategy

Like euler explicit method, euler modified method is also a single-step explicit method to find numerical solution of first order ODE.
It is different from euler explicit method in calculating the slope for the equation – $y_{i+1} = y_i + \text{slope} * h$. It uses the average of 2 slopes to find the better approximate slope for equation. As this method approximates the value for slope better than euler explicit method, the solutions that we get through this method are more accurate.

b. Big Picture



In the Modified Euler's method, we calculate the values of x and y as follows:
$x_{i+1} = x_i + h$
$\text{slope1} = F(x_i, y_i)$
$y_{i+1}(eu) = y_i + F(x_i, y_i)*h$
$\text{slope2} = F(x_{i+1}, y_{i+1}(eu))$
$y_{i+1} = y_i + ((\text{slope1} + \text{slope2}) / 2) * h$

Here, h is a constant value that determines how close we want the values of x.
As we can see in the above picture, we take 2 slopes – one at beginning of interval and one at end of interval and take average of them to find final slope. The reason that it gives better approximation of slope is because we assume that if h is small, then the value of slope in any point between the interval would be between the value of slope at end points. Finding the average would give better approximation for slope. Hence, better the estimate for slope, better will be the estimate for solution.

This is an iterative process where the value of the next term depends on the values of the previous terms. Each time we calculate the value of x and y, we will use that to calculate the values of our next x and y until we reach our desired tolerance or finish our domain.

c. Algorithm

Step 1: Given a solution at point (xi, yi), find $x_{i+1} = x_i + h$

Step 2: Calculate $f(x_i, y_i)$

Step 3: Estimate y_{i+1} using Euler's explicit method

$y_{i+1}(Eu) = y_i + f(x_i, y_i)h$

Step 4: Calculate $f(x_{i+1}, y_{i+1}(Eu))$

Step 5: Calculate the numerical solution at $x = x_{i+1}$

$y_{i+1} = y_i + (f(x_i, y_i) + f(x_{i+1}, y_{i+1}(Eu)))/2 * h$

Step 6: Repeat steps 1-5 for the next points until the desired range is covered.

d. Example

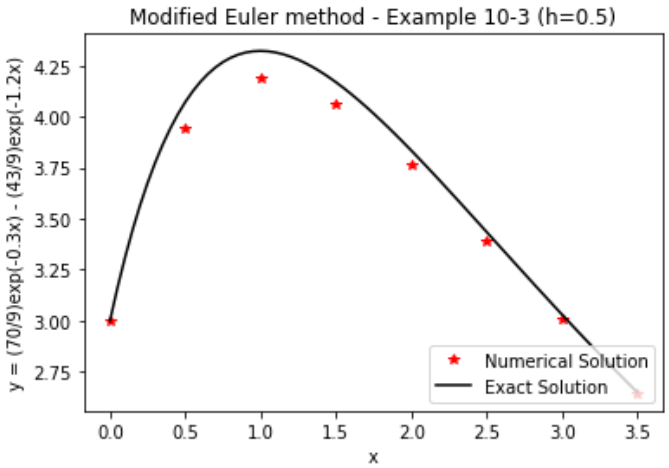
Use modified Euler method to solve the ODE dy/dx = - 1.2 y + 7 e ^(-0.3x) from x = 0 to x = 3.5 with the initial condition y = 3 at x = 0.

Soln.

When we use the above algorithm in python to find the solution, we get the following result –

```
In [2]: runfile('C:/Users/laksh/Downloads/Euler_Modified_Example10_3(1)
(1).py', wdir='C:/Users/laksh/Downloads')
 i    x[i]    y[i]    yTrue[i]    err[i]

 1   0.0000   3.0000   3.0000     0.0000

 2   0.5000   3.9462   4.0723     0.1261

 3   1.0000   4.1877   4.3229     0.1351

 4   1.5000   4.0633   4.1696     0.1063

 5   2.0000   3.7635   3.8351     0.0716

 6   2.5000   3.3936   3.4361     0.0425

 7   3.0000   3.0105   3.0317     0.0212

 8   3.5000   2.6431   2.6501     0.0070
```

Graphically, when we plot these values, we see that our method as closely approximated its solution to true solution values.



e. Pros and Cons of using this method

Modified Euler's method provides better approximation of the numerical solution compared to euler's explicit method. The problem is the computational overhead because of using more steps than euler's explicit method to estimate the result.

f. Error Analysis

The local truncation error with the improved Euler method is $O(h^3)$.
For the Modified Euler's Method, the global truncation error is of order $O(h^2)$.

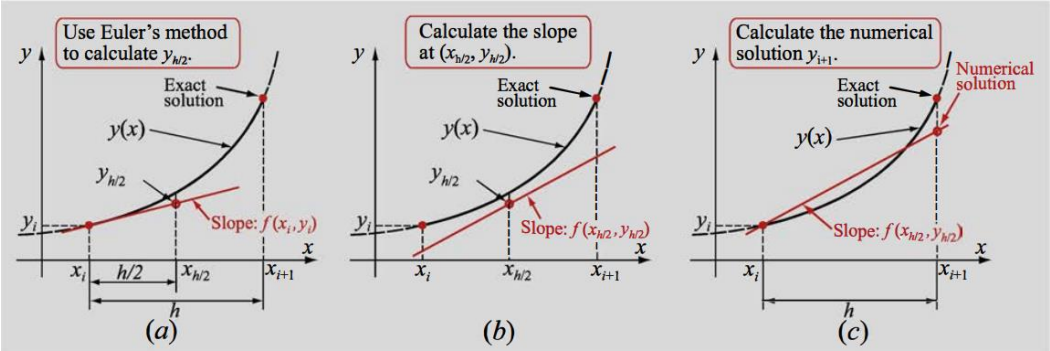
3. Midpoint method

a. Strategy

The general idea for midpoint method is almost same as euler's explicit method and modified euler's method. It is also used to find the numerical solution of first-order ODE and is a single-step explicit method.

It is different than euler's explicit method and modified euler method in the way of calculating slope. Here, the slope for calculating y_{i+1} is an estimate of the middle point of the interval.

b. Big Picture



In the Modified Euler's method, we calculate the values of x and y as follows:

$x_m = x_i + (h/2)$

slope1 = $F(x_i, y_i)$

$y_m = y_i + F(x_i, y_i)*(h/2)$

slope2 = $F(x_m, y_m)$

$y_{i+1} = y_i$ + slope2 * h

Here, h is a constant value that determines how close we want the values of x.

As we can see from the above image, we use the euler's explicit method first to calculate the value of y at half h(x_m), say y_m. Then we use the middle points of x and y, i.e., (x_m, y_m) to calculate the slope for that point. We do this because we assume that the slope at (x_m, y_m) would be more accurate. Hence, the value for numerical solution is also more accurate. This has similar concept to euler's explicit method and modified euler's method – just that this time we use a different method to estimate the value for slope.

Like previous methods, this is also an iterative process where the value of the next term depends on the values of the previous terms. Each time we calculate the value of x and y, we will use that to calculate the values of our next x and y until we reach our desired tolerance or finish our domain.

c. Algorithm

Step 1: Given a solution at point (x_i, y_i), find $x_m = x_i + (h/2)$
Step 2: Calculate $f(x_i, y_i)$
Step 3: Estimate y_{i+1} using Euler's explicit method

$y_m = y_i + f(x_i, y_i)*(h/2)$

Step 4: Calculate $f(x_m, y_m)$
Step 5: Calculate the numerical solution at $x = x_{i+1}$

$y_{i+1} = y_i + f(x_m, y_m) * h$

Step 6: Repeat steps 1-5 for the next points until the desired range is covered.

d. Example

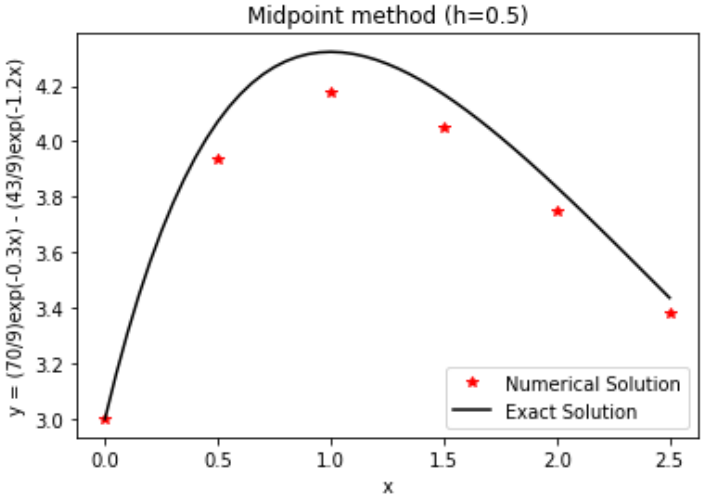
Use midpoint method to solve the ODE dy/dx = - 1.2 y + 7 e ^(-0.3x) from x = 0 to x = 2.5 with the initial condition y = 3 at x = 0.

Soln.

When we use the above algorithm in python to find the solution, we get the following result –

```
In [4]: runfile('C:/Users/laksh/Downloads/Midpoint_script(1).py',
wdir='C:/Users/laksh/Downloads')
 i    x[i]    y[i]    yTrue[i]    err[i]

 1   0.0000   3.0000   3.0000     0.0000

 2   0.5000   3.9371   4.0723     0.1352

 3   1.0000   4.1746   4.3229     0.1483

 4   1.5000   4.0489   4.1696     0.1207

 5   2.0000   3.7493   3.8351     0.0858

 6   2.5000   3.3804   3.4361     0.0557
```

Graphically, when we plot these values, we see that our method as closely approximated its solution to true solution values.



e. Pros and Cons of using this method

The biggest advantage of using this method is its accuracy to estimate the numerical solution. It estimates the numerical solution better than euler's explicit method.
On the other hand, although this method gives better estimate of result, it has more computational overhead than euler's method.
Comparing with modified euler method, it depends on the problem in hand which one is better to use. Some problems are solved better using using midpoint method and some problems are solved better using modified Euler method. Finally, it depends on the experience or expertise of the user which method wants to use among these.
Other method such as runge-kutta also gives good approximations many times. We will see runge-kutta later.

f. Error analysis

Error analysis for midpoint method is similar to modified Euler method.
The local truncation error with the midpoint method is $O(h^3)$.
For the midpoint Method, the global truncation error is of order $O(h^2)$.

4. Second-Order Runge-Kutta methods.

a. Strategy

Runge-Kutta methods are a family of single-step, explicit, numerical techniques for solving a first-order ODE.

It is similar to the previous methods we did, except this method is classified according to its order. The order identifies the number of points within the subinterval that are used for determining the value of slope. For e.g., second order runge-kutta methods use 2 points, third order runge-kutta methods use 3 points and so on.

In fact, the previous methods (modified euler's method and midpoint method) we did can be seen as special cases for second-order runge-kutta method.

In particular, the idea for second-order runge-kutta method is to take 2 slopes and we can give them weights to get more accurate results for numerical solution. The place where the slope will be calculated will depend on constants attached to slopes.

b. Big Picture

The general form of second-order Runge-Kutta methods is:

$$y_{i+1} = y_i + (c_1K_1 + c_2K_2)h$$

where,

$$K_1 = f(x_i, y_i)$$
$$K_2 = f(x_i + a_2h, y_i + b_{21}K_1h)$$

where c_1, c_2, a_2, and b_{21} are constants.

If you observe, **modified euler method** is also similar to this method when we place the values of constants to certain values.

When the values of constants are –

$$c_1 = \frac{1}{2}, \quad c_2 = \frac{1}{2}, \quad a_2 = 1, \quad \text{and} \quad b_{21} = 1$$

Substituting these values in the general form of second order Runge-Kutta method –

$$y_{i+1} = y_i + \frac{1}{2}(K_1 + K_2)h$$

where,

$$K_1 = f(x_i, y_i)$$
$$K_2 = f(x_i + h, y_i + K_1h)$$

We can also derive **midpoint method** from second-order runge-kutta method.

For midpoint, take constants as –

$$c_1 = 0, \quad c_2 = 1, \quad a_2 = \frac{1}{2}, \quad \text{and} \quad b_{21} = \frac{1}{2}$$

Substituting these values in general form of runge-kutta method, we get –

$$y_{i+1} = y_i + K_2h$$

where,

$$K_1 = f(x_i, y_i)$$
$$K_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}K_1h\right)$$

c. Algorithm

Step 1: Start with an initial point (x0, y0).
Step 2: Define the function f(x, y) that represents the differential equation.
Step 3: Choose a small step size h.
Step 4: Define the constants a2, b21, c1, and c2.
Step 5: Calculate the two "k" constants:
    k1 = h * f(x0, y0)
    k2 = h * f(x0 + a2*h, y0 + b21*k1*h)
Step 6: Estimate the next y-value using the formula: y1 = y0 + (c1*k1 + c2*k2)*h.
Step 7: Update the current x-value: x1 = x0 + h.
Step 8: Repeat steps 5-7 for the next points until the desired range is covered.

d. Example

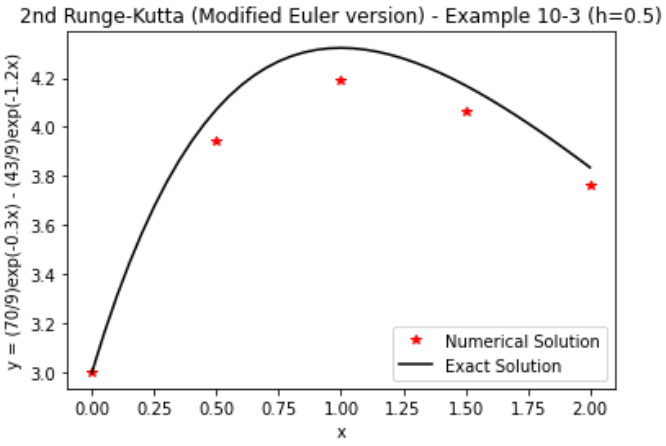
Use second-order runge-kutta method to solve the ODE dy/dx = - 1.2 y + 7 e ^(-0.3x) from x = 0 to x = 2.0 with the initial condition y = 3 at x = 0.

Soln.

When we use the above algorithm in python to find the solution, we get the following result –

```
In [5]: runfile('C:/Users/laksh/Downloads/
RK_Second_order_Example10_4.py', wdir='C:/Users/laksh/Downloads')
 i    x[i]    y[i]    yTrue[i]    err[i]

 1   0.0000   3.0000   3.0000     0.0000

 2   0.5000   3.9462   4.0723     0.1261

 3   1.0000   4.1877   4.3229     0.1351

 4   1.5000   4.0633   4.1696     0.1063

 5   2.0000   3.7635   3.8351     0.0716
```

Graphically, when we plot these values, we see that our method as closely approximated its solution to true solution values.



e. Pros and Cons of using this method

Runge-Kutta method has better estimation of the solution than euler's explicit method. As it has constants based on which we can find different estimates for slopes; if an expert knows based on experience that in their specific field what constant to use for specific equation – it can yield better approximation of numerical solution than any other method.

The biggest consequence is computational overhead – it will take more computation to cover a specific domain than euler's explicit method.

f. Error Analysis

Local truncation error is $O(h^3)$

Global truncation error is $O(h^2)$

5. Third-Order Runge-Kutta methods

a. Strategy

This is similar to second-order Runge-Kutta methods, except this time, three points will be used to estimate the slope based on their weights.

b. Big Picture

The general form of third-order Runge-Kutta methods is:

$$y_{i+1} = y_i + (c_1K_1 + c_2K_2 + c_3K_3)h$$

with

$$K_1 = f(x_i, y_i)$$
$$K_2 = f(x_i + a_2h, y_i + b_{21}K_1h)$$
$$K_3 = f(x_i + a_3h, y_i + b_{31}K_1h + b_{32}K_2h)$$

The approach to find numerical solution through third-order runge-kutta method is similar to second-order runge-kutta method except that this time we have 3 points to estimate slope and 8 constants.

Based on experience of scientists, they have found that certain values of constants are generally used to approximate the numerical solution accurately. One method from those is called classical third-order Runge-Kutta method.

The values of the eight constants in this method are:

$$c_1 = \frac{1}{6},\ c_2 = \frac{4}{6},\ c_3 = \frac{1}{6},\ a_2 = \frac{1}{2},\ a_3 = 1,\ b_{21} = \frac{1}{2},\ b_{31} = -1, \text{ and } b_{32} = 2$$

With these constants the equations for the classical third-order Runge-Kutta method are:

$$y_{i+1} = y_i + \frac{1}{6}(K_1 + 4K_2 + K_3)h$$

where,

$$K_1 = f(x_i, y_i)$$
$$K_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}K_1h\right)$$
$$K_3 = f(x_i + h, y_i - K_1h + 2K_2h)$$

Other than this, there are other constants also that give a good approximation for numerical solutions. These are –

Method	c_1	c_2	c_3	a_2	b_{21}	a_3	b_{31}	b_{32}
Classical	1/6	4/6	1/6	1/2	1/2	1	–1	2
Nystrom's	2/8	3/8	3/8	2/3	2/3	2/3	0	2/3
Nearly Optimal	2/9	3/9	4/9	1/2	1/2	3/4	0	3/4
Heun's Third	1/4	0	3/4	1/3	1/3	2/3	0	2/3

c. Algorithm

Step 1: Start with an initial point (x0, y0).
Step 2: Define the function f(x, y) that represents the differential equation.
Step 3: Choose a small step size h.
Step 4: Define the constants a2, a3, b21, b31, b32, c1, c2, and c3.
Step 5: Calculate the three "k" constants:

k1 = h * f(x0, y0)
k2 = h * f(x0 + a2*h, y0 + b21*k1*h)
k3 = h * f(x0 + a3*h, y0 + b31*k1*h + b32*k2*h)

Step 6: Estimate the next y-value using the formula: y1 = y0 + (c1*k1 + c2*k2 + c3*k3)*h.
Step 7: Update the current x-value: x1 = x0 + h.
Step 8: Repeat steps 5-7 for the next points until the desired range is covered.

d. Example

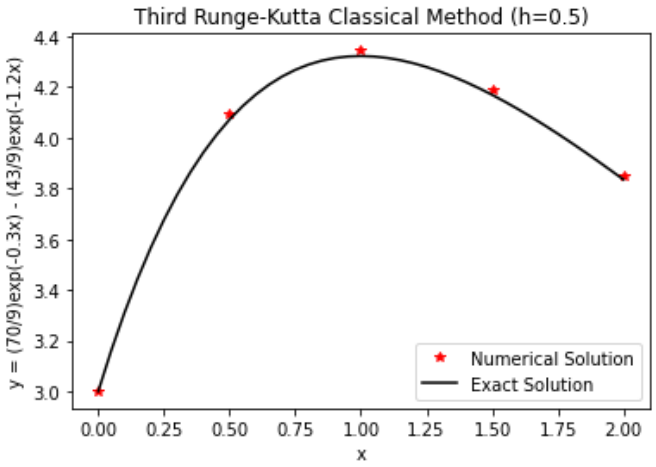
Use classical third-order runge-kutta method to solve the ODE dy/dx = - 1.2 y + 7 e ^(-0.3x) from x = 0 to x = 2.0 with the initial condition y = 3 at x = 0.

Soln.

When we use the above algorithm and put classical third-order runge-kutta method constants in python to find the solution, we get the following result –

```
In [6]: runfile('C:/Users/Laksh/Downloads/RK_Third_order.py', wdir='C:/
Users/Laksh/Downloads')
 i    x[i]    y[i]    yTrue[i]    err[i]

 1   0.0000   3.0000   3.0000     0.0000

 2   0.5000   4.0927   4.0723     -0.0204

 3   1.0000   4.3444   4.3229     -0.0215

 4   1.5000   4.1863   4.1696     -0.0167

 5   2.0000   3.8464   3.8351     -0.0113
```

Graphically, when we plot these values, we see that our method as closely approximated its solution to true solution values.



e. Pros and Cons of using this method

This method finds slope using three points, hence gives better estimate of the numerical solution than second-order runge kutta method.

The disadvantage is that although it gives better estimation for numerical solution, its computational cost is also higher.

As with the second-order runge-kutta method, this method can also be beneficial for certain types of problems and if someone with experience in their field uses this method with constants that give accurate results – this method can be most beneficial in that case.

f. Error Analysis

The local truncation error in third-order Runge-Kutta methods is $0(h^4)$, and the global truncation error is $0(h^3)$.

6. Fourth-Order Runge-Kutta methods

a. Strategy

This is also similar to third-order Runge-Kutta methods, except this time, four points will be used to estimate the slope based on their weights.

b. Big Picture

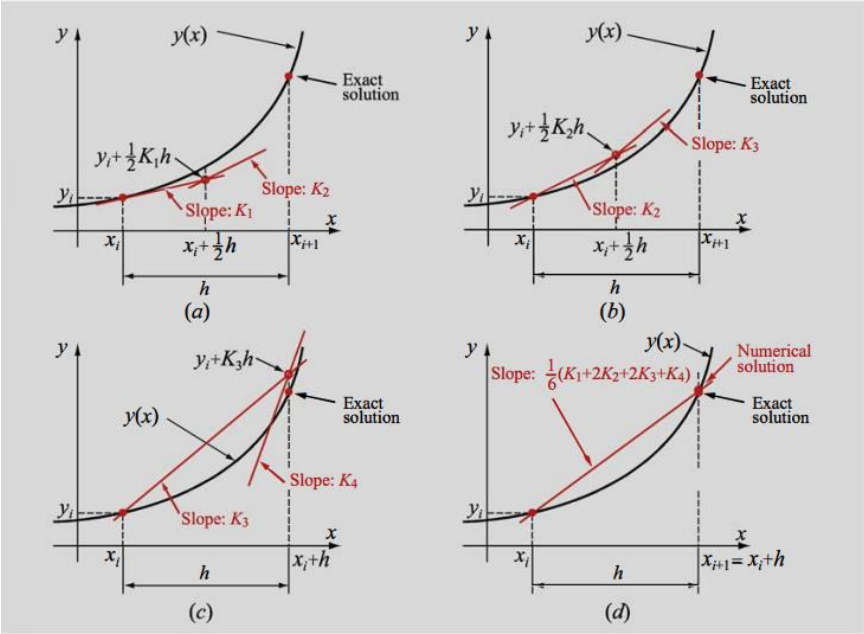


Figure 10-12: The classical fourth-order Runge–Kutta method.

The general form of fourth-order Runge-Kutta methods is:

$$y_{i+1} = y_i + (c_1K_1 + c_2K_2 + c_3K_3 + c_4K_4)h$$

where,

$$K_1 = f(x_i, y_i)$$
$$K_2 = f(x_i + a_2h, y_i + b_{21}K_1h)$$
$$K_3 = f(x_i + a_3h, y_i + b_{31}K_1h + b_{32}K_2h)$$
$$K_4 = f(x_i + a_4h, y_i + b_{41}K_1h + b_{42}K_2h + b_{43}K_3h)$$

The approach to find numerical solution for fourth-order runge-kutta method is similar to third-order runge-kutta method except that this time we use 4 points to estimate the slope and we use 13 constants.

Similar to classical third-order runge-kutta method, we have classical fourth-order runge-kutta method. This method can give good estimate of numerical solution and is commonly used.

The constants of this method are –

$$c_1 = c_4 = \frac{1}{6}, \quad c_2 = c_3 = \frac{2}{6}, \quad a_2 = a_3 = b_{21} = b_{32} = \frac{1}{2}$$
$$a_4 = b_{43} = 1, \quad b_{31} = b_{41} = b_{42} = 0$$

With these constants the equations for the classical fourth-order Runge-Kutta method are:

$$y_{i+1} = y_i + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4)h$$

where,

$$K_1 = f(x_i, y_i)$$

$$K_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}K_1 h\right)$$

$$K_3 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}K_2 h\right)$$

$$K_4 = f(x_i + h, y_i + K_3 h)$$

In the picture above, the graphical representation of fourth-order runge-kutta method is shown. As we can see the first slope is calculated at the initial point, using which the second slope is calculated, using which the third slope is calculated, using which the fourth slope is calculated. Even, another thing to appreciate here is that all the 4 slopes are spread through the graph, i.e., one at the starting point, 2 at the middle and one at end, hence giving better estimation for numerical solution.

c. Algorithm

Step 1: Start with an initial point (x0, y0).
Step 2: Define the function f(x, y) that represents the differential equation.
Step 3: Choose a small step size h.
Step 4: Define the constants a2, a3, a4, b21, b31, b32, b41, b42, b43, c1, c2, c3, and c4.
Step 5: Calculate the four "k" constants:

k1 = h * f(x0, y0)
k2 = h * f(x0 + a2*h, y0 + b21*k1*h)
k3 = h * f(x0 + a3*h, y0 + b31*k1*h + b32*k2*h)
k4 = h * f(x0 + a4*h, y0 + b41*k1*h + b42*k2*h + b43*k3*h)

Step 6: Estimate the next y-value using the formula: y1 = y0 + (c1*k1 + c2*k2 + c3*k3 + c4*k4)*h.
Step 7: Update the current x-value: x1 = x0 + h.
Step 8: Repeat steps 5-7 for the next points until the desired range is covered.

d. Example

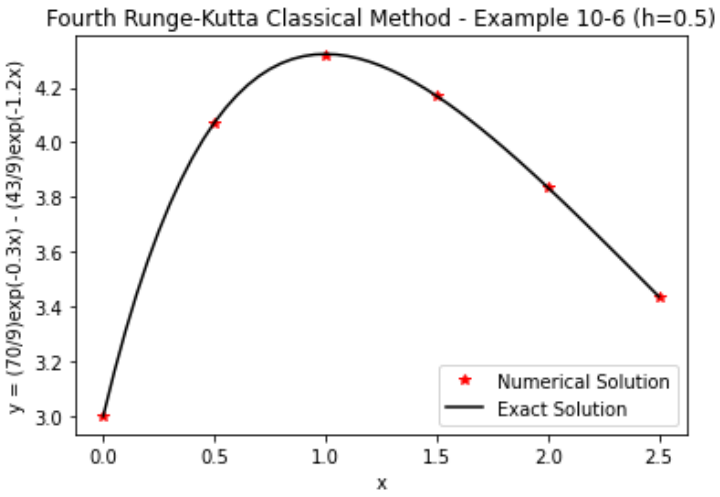
Use classical fourth-order runge-kutta method to solve the ODE dy/dx = - 1.2 y + 7 e ^(-0.3x) from x = 0 to x = 2.5 with the initial condition y = 3 at x = 0.

Soln.

When we use the above algorithm and put classical fourth-order runge-kutta method constants in python to find the solution, we get the following result –

```
In [7]: runfile('C:/Users/laksh/Downloads/RK_Fourth_order.py', wdir='C:/
Users/laksh/Downloads')
 i    x[i]    y[i]    yTrue[i]    err[i]

 1   0.0000   3.0000   3.0000     0.0000

 2   0.5000   4.0698   4.0723     0.0025

 3   1.0000   4.3203   4.3229     0.0026

 4   1.5000   4.1676   4.1696     0.0020

 5   2.0000   3.8338   3.8351     0.0013

 6   2.5000   3.4353   3.4361     0.0008
```

Graphically, when we plot these values, we see that our method as closely approximated its solution to true solution values.



e. Pros and Cons of using this method

This is the most accurate numerical solution estimating method among all the 6 methods we have seen. But as with the other methods, the more the accuracy, trade-off is with the computing cost; same goes with this method too.

f. Error Analysis

The local truncation error in fourth-order Runge-Kutta methods is $O(h^5)$, and the global truncation error is $O(h^4)$.

6. Conclusion

The common pattern that was seen was that as the accuracy of numerical solution was increasing, it came with a trade-off with computing cost.

Comparing all the 6 methods, fourth-order runge-kutta method is the most accurate one because its global truncation error is the least among all the methods. On the other hand, it also takes the most computation.

Among mid-point method and modified euler's method, it depends on the problem to decide which one is more accurate. For some problems, mid-point method can be more beneficial and for some problems, modified euler's method can be more accurate. They both have almost same global truncation error in general. It also makes sense because they are special cases of second-order runge-kutta method.

Euler's explicit method is the simplest among all the 6 methods and even takes less computation, but comes with trade-off for accuracy.

Finally, it depends a lot on problem which method suits the best for it for most accurate results with less computation cost.

7. Learnings

The observation that I made was that the technique of one method is also applied to another method. For e.g., in case of midpoint method, we used the same concept we used in modified euler's method. We didn't had the end point to use – hence we estimated that using euler's explicit method. Then we made different modifications in both the cases, but the initial step was the same.

As I had also taken differential equation course along with this course, it was amazing first seeing analytic method to solve differential equation and then seeing numerical method.

What I observed was computers are good to calculate some value for a problem, but they are not that good when it comes to thinking. For e.g., if I give computer a code to solve ordinary differential equation using one of the 6 methods stated above, it would be easily be able to solve the problem if it comes in its range. But it is difficult to ask computer to model a differential equation. As much as I have seen in physics, modelling an equation is the more sensitive and difficult part – atleast compared to solving them.

At last, I would say the future work for the 6 methods we learnt could be to derive a way such that accuracy is maintained and computing cost also remains low.

8. References

[1] Numerical Methods for Engineers and Scientists – 3 ed., Amos Gilat, Vish Subramaniam

[2] Lecture Slides for Computational Mathematics - CHAPTER 10 ORDINARY DIFFERENTIAL EQUATIONS: INITIAL-VALUE PROBLEMS

[3] Lecture Codes – Chapter 10

[4]https://math.libretexts.org/Bookshelves/Differential_Equations/Elementary_Differential_Equations_with_Boundary_Value_Problems_%28Trench%29/03%3A_Numerical_Methods/3.02%3A_The_Improved_Euler_Method_and_Related_Methods