`

CHEMICAL ENGINEERING 2E04

Chapter 5 – Ordinary Differential Equations

Module 5A: Introduction and Univariate Methods

Dr. Jake Nease

Kieran McKenzie

Steven Karolat

Cynthia Pham

Chemical Engineering

McMaster University

Updated November 28, 2019

Contents

Suggested Readings

Gilat, V. Subramaniam: *Numerical Methods for Engineers and Scientists.* Wiley. Third Edition (2014)

Euler's Method
  - Chapter 10 → *10.2.1* (Only Euler's Explicit Method)
- Heun's Method (Referred to as 'Modified Euler's Method' in the Book)
  - Chapter 10 → *10.3*
- Fourth Order Runge-Kutta Method
  - Chapter 10 → *10.5.3*

Online Material

Euler's Method Video *(Click to follow hyperlink)*

- Provides a visual animation of Euler's Method as well as its error

Introduction and Perspective

We've made it to the final chapter – differential equations! First, let's go over an explanation to tell you exactly what a differential equation represents, so that we understand why and where they are used.

A *differential equation (DE)* is any equation that contains the derivatives of a function as well as the function itself. Why would an equation do this?

We've seen many mathematical relationships usually described by writing the dependent variable as a function of the independent variable, such as $y = f(x)$. However, suppose we do not know the relation $y = f(x)$, but we know how the dependent variable, y, changes slope with respect to the independent variable, x, (*i.e.* the rate of change, $\frac{dy}{dx}$) and that this rate of change depends on BOTH x and y:

$$\frac{dy}{dx} = f(x, y)$$

This is known as a differential equation. For example, the rate of liquid flowing out of a tank might depend on the liquid volume in the tank (ever noticed how a bathtub drains faster when full compared to empty?).

Our task is to use this information to solve for $y = F(x)$, the so-called *solution* to the differential equation (DE).

Primary Learning Outcomes

- Introduce differential equations and what it means to *solve* an ordinary differential equation.
- Derive *Euler's method* to numerically solve differential equations.
- Modify Euler's method to create *Heun's method.*
- Familiarize ourselves with *Runge-Kutta methods* for solving differential equations.
- Compare each method's merits based on *computational complexity* and accuracy.

Applications

The applications of differential equations are VAST in chemical engineering, including areas such as:

- Chemical kinetics, reaction processes, equilibrium problems, fluid mechanics.
- Non-steady-state heat propagation through materials (Heat Transfer).
- Banking (how your saving account grows with interest) and stock market analysis.
- Classic physics problems: Hooke's Law, free motion, circuit analysis, wave/vibration equations…

Differential Equations: Quick Terminology Review

Differential Equation Terminology

Order of a DE :	Determined by the highest derivative contained in the equation.
Homogenous DE :	A DE that has NO functions of the independent variable in the equation. If the dependent variable and its derivatives are isolated onto one side of the equation, the other side will be zero.
Nonhomogeneous DE :	A DE that does not satisfy the definition of a homogenous DE.

Let's look at some examples:

$$\frac{dy}{dx} + y = 0$$

- Homogeneous
- 1st order Ordinary Differential Equation (ODE)

$$\frac{dy}{dx} + y = g(x)$$

- Nonhomogeneous
- 1st order ODE

$$\frac{d^4y}{dx} + \frac{d^3y}{dx} + \frac{d^2y}{dx} + \frac{dy}{dx} + y = 0$$

- Homogeneous
- 4th order ODE

DEs are separated into two main categories: *Ordinary Differential Equations (ODEs)* and *Partial Differential Equations (PDEs)*. We have seen examples of ODEs above. The only difference in a PDE is that the dependent variable is a function of more than one independent variable (ex: $u(x, t)$ describes u as a function of x and t), and therefore the differential equation must contain partial derivatives!

In this course, we'll focus on *numerically solving first order ODEs*, meaning that we will only be dealing with first derivatives of the dependent variable.

Arranging our ODEs for Solving

For the purpose of this course, we will be setting up our first order ODEs as follows:

$$\frac{dy}{dx} = f(x, y)$$

Where, $f(x, y)$ will represent the *slope* of the function and can be any function of x and/or y. Our final goal is to achieve $y = F(x)$ numerically.

Solving a First Order ODE: Initial Conditions vs. Boundary Conditions

Suppose we have a function that demonstrates the flux of contaminants J through a membrane. The rate of change of flux for a change in pressure ($\frac{dJ}{dP}$) is directly proportional to the pressure and the current rate of flux, as follows:

$$\frac{dJ}{dP} = 2P - 5J$$

We have the *slope of the function* $J(P)$ (the one we're trying to solve for) at ANY point (P, J) - if we plug a point into this equation, we get the slope value at that point! Plugging in a bunch of sample points, and plotting the slopes at those points as unit vectors, we get a *direction field* (also known as a slope field):

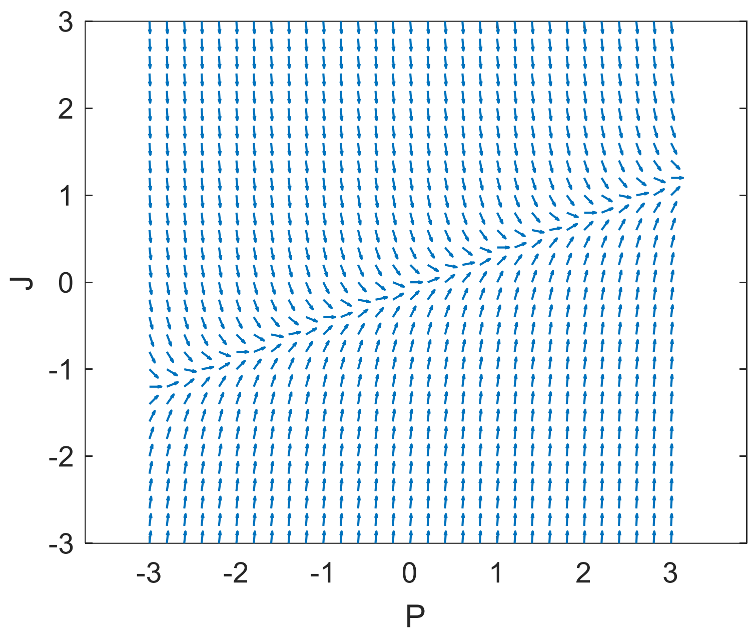


Figure 1: Direction field for $\frac{dJ}{dP} = 2P - 5J$

The function $J(P)$ has the analytical solution, $J(P) = c_1 e^{-5P} + \frac{2P}{5} - \frac{2}{25}$. The c_1 can be solved for using an initial condition for the initial value of J and P. Depending on this initial condition, the solution looks very different:

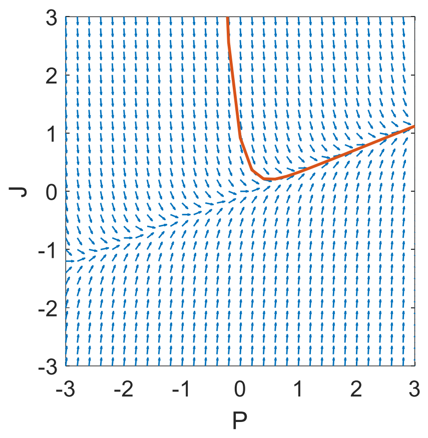


Figure 2: A solution for $J(P)$ *when* $c_1 = 1$.

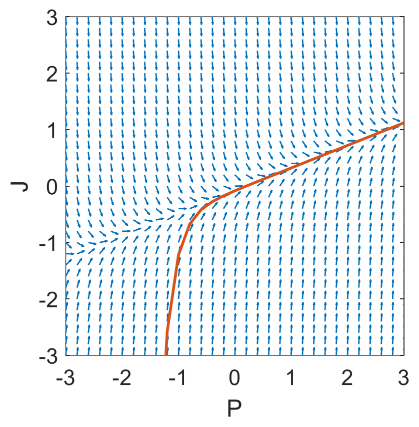


Figure 3: A solution for $J(P)$ *when* $c_1 = -\frac{1}{200}$.

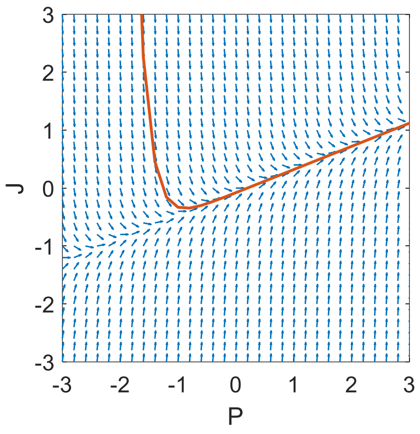


Figure 4: A solution for $J(P)$ *when* $c_1 = \frac{1}{1000}$.

Every situation will only have one exact solution. To arrive at this solution, we require *initial conditions* or *boundary conditions* (boundary conditions tell you the value of the function at the endpoints of the function's domain).

Initial Conditions and Initial Value Problems (IVPs)

Initial Condition	A statement which tells you the value of a function curve or its derivatives when the independent variable is zero, such as $F(0)$ if our solution function is $y = F(x)$.
Initial Value Problems	When initial condition(s) are used to solve for the constants of integration in the general form of an ODE.

The same number of conditions as there are constants in our general solution will be required to achieve zero DOF and find the exact solution. For example, if we have two constants to solve for, then we could use $y(0)$ and $y'(0)$ as our two initial conditions.

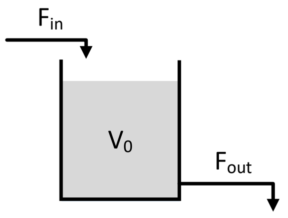
Most ODEs in chemical engineering are FAR too difficult to solve by hand. Thus, we must develop numerical methods for solving ODEs!

Workshop: Initial Conditions

A tank is being simutaneously filled, F_{in}, and drained, F_{out}, starting with a volume, V_0.

The ODE describing the relationship is shown. Examine the impact of changing initial conditions on this scenario by *sketching* the volume in the tank with respect to time (base case).

Then, consider the situations where V_0 is different or F_{out} is different and sketch those.

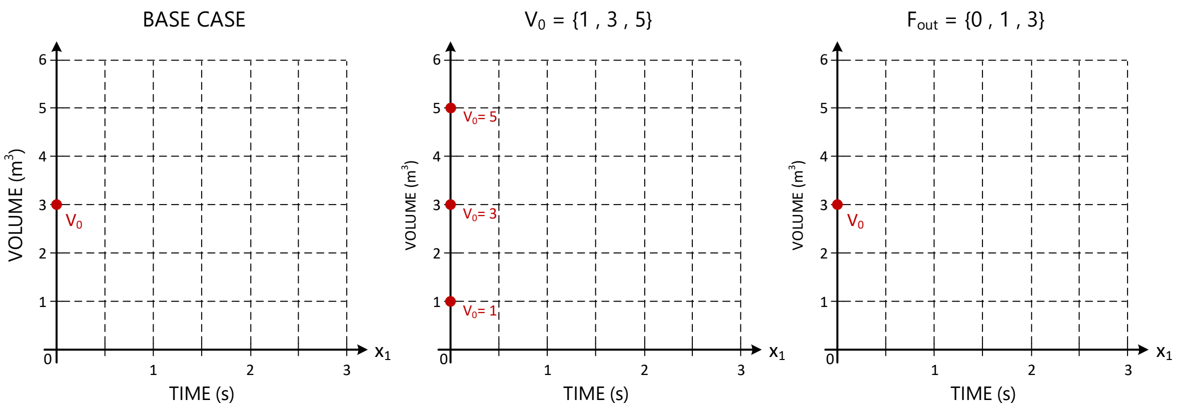


The relationship is described with:

$$\boxed{\frac{dV}{dt} = F_{in} - F_{out}}$$

The base case is:

$$F_{in} = 2\frac{m^3}{s}, \quad F_{out} = 1\frac{m^3}{s}, \quad \text{and } V_0 = 3\ m^3.$$



Numerical Solutions of Differential Equations

We'll cover *Euler's* method, *Heun's* method, and *Runge-Kutta* methods to numerically solve ODEs.

The general concepts behind these methods follow the same general procedure:

1) Start at a given point on the function solution. Use the ODE to determine the slope of the function at this point.
2) Extrapolate that slope for a certain amount of time/space to estimate the NEXT point on the function solution.
3) Repeat the procedure for the span of integration.

Euler's Method

Euler's method (also known as "Forward Euler's method") begins at a *specified initial condition* (initial temperature of the surface, height of the tank, concentration of species in the reactor, *etc.*). Euler's method assumes that over a small step size (h) the *slope* of the function is constant (Figure 5 - *Figure 6* show why this assumption works reasonably well).

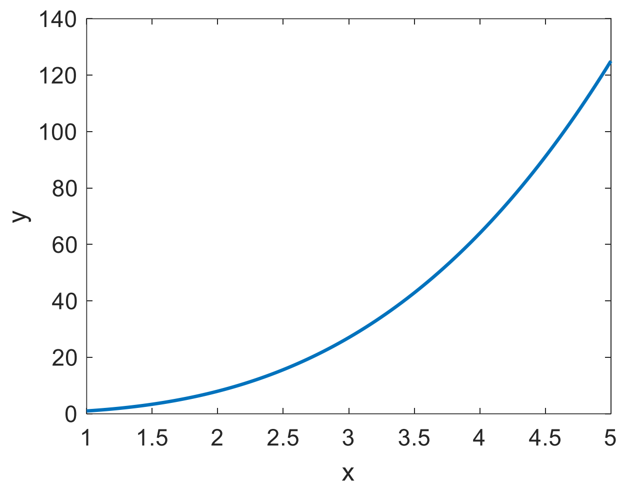


Figure 5: Plot of $y = x^3$.

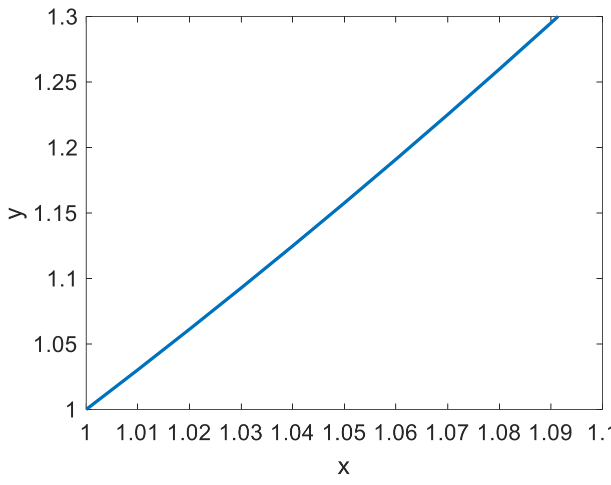


Figure 6: Zoomed in plot of $y = x^3$ showing near-constant slope at an h of 0.1.

Under this assumption, *starting at the initial point*, the next point can be calculated using the slope at the current point, (x_i, y_i), and the step size h. The general Euler's method procedure is simple:

1. The ODE $\frac{dy}{dx} = f(x_i, y_i)$ represents the slope of $F(x_i)$ at some point (x_i, y_i)
2. If that slope is assumed *constant* for some step size h, the values of x_{i+1} and $y_{i+1} = F(x_{i+1})$ are just:

$$x_{i+1} = x_i + h$$
$$y_{i+1} \approx y_i + h \times f(x_i, y_i)$$

Note: Since we can assume that $\frac{dy}{dx} = f(x, y)$ is *constant* over a small change in x, we actually integrate $f(x, y)$ using the left-rectangle rule and apply that change to $y = F(x)$!

The *local error* (the error inherent at each step of a numerical method), is the difference between the true y_{i+1} and the numerical solution, y_{i+1}^{NS}. For Euler's Method, the local error is $\mathcal{O}(\mathrm{h}^2)$ (from terms truncated in its Taylor Series derivation).

Local errors will compound throughout the solution – this 'total' is termed *global error*. In other words, the local error is applied $N = \frac{b-a}{h}$ times. This is oversimplifying it, but the global error for Euler's Method ends up being $\mathcal{O}(\mathrm{h})$.

Euler's Method

The following is the algorithm to integrate an ODE $\frac{dy}{dx} = f(x, y)$ on the interval $[x_1, x_{end}]$ using Euler's Method from an initial condition x_1 and $y_1^{NS} = y_1 = F(x_1)$:

1. Set the given initial point as the first point of the numerical solution. Select a step size h and an iteration counter $i = 1$.

2. Evaluate the slope at the current point, $f(x_i, y_i^{NS})$:

$$f(x_i, y_i^{NS}) \triangleq \left.\frac{dy}{dx}\right|_{(x_i, y_i^{NS})}$$

3. Calculate the next point, (x_{i+1}, y_{i+1}^{NS}), as:

$$x_{i+1} = x_i + h:$$
$$y_{i+1}^{NS} = y_i^{NS} + f(x_i, y_i^{NS})(h)$$

4. IF $x_{i+1} = x_{end}$

```
4.  IF $x_{i+1} = x_{end}$
            STOP
    ELSE
            Update $i = i + 1$. Return to step 2.
```

Example: Euler's Method

Retrieved from Gilat and Subramaniam, Numerical Methods for Engineers and Scientists, 2014. Page 403:

The first-order ODE $\frac{dy}{dx} = -1.2y + 7e^{-0.3x}$ has an analytical solution with initial conditions $(x_1 = 0, y_1 = 3)$ to be $y = \frac{70}{9}e^{-0.3x} - \frac{43}{9}e^{-1.2x}$. Below, the plot of the exact solution is shown in blue on the range $0 \leq x \leq 2.5$, while Euler's Method (using two different step sizes) is shown as red asterisks:

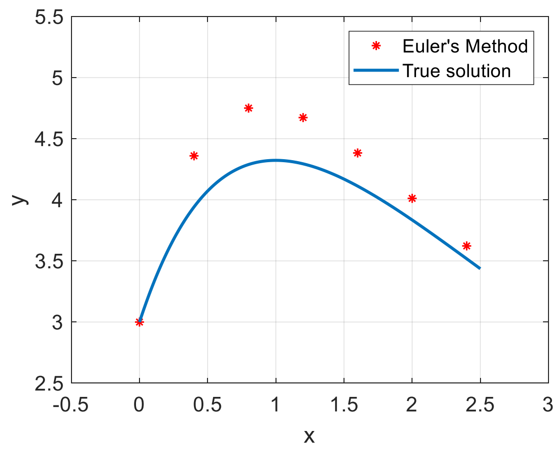


Figure 7: Using Euler's method with $h = 0.4$ to solve the first order linear ODE with the true solution shown in blue.

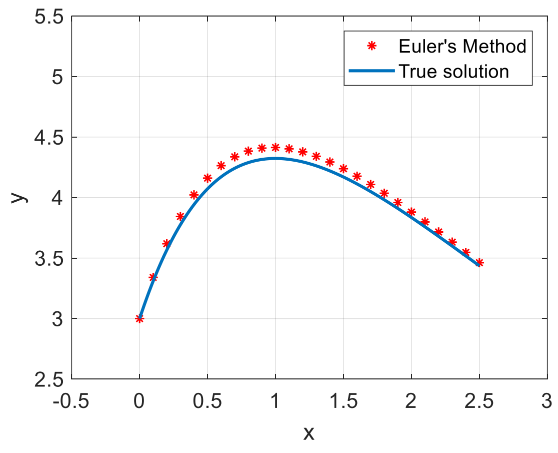


Figure 8: Using Euler's method with $h = 0.1$ to solve the first order linear ODE with the true solution shown in blue.

Workshop: Euler's Method

The curve shown to the right is the solution to the IVP:

$$\frac{dy}{dx} = 2 + 2xy \qquad y(0) = 1$$

Starting at the initial point$(x = 0, y = 1)$, calculate the next 4 points *with Euler's Method using* $\mathrm{h} = 0.5$ and draw them on the graph!

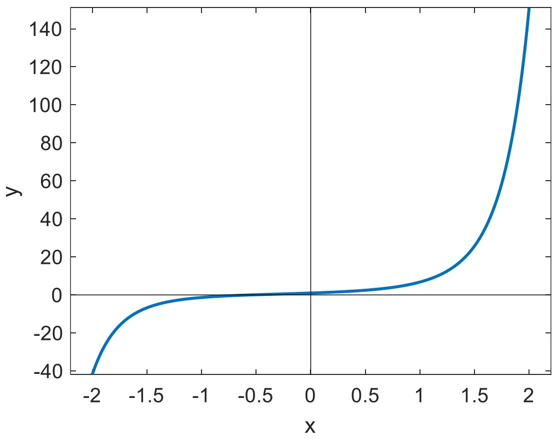


Figure 9: True Solution for Workshop Example

Great! As you can tell, the points began to veer off from the true solution – our step size must have been too large. Together, let's *code* Euler's Method, then *modify the step size* until we can get an accurate representation ☺.

```
function [x, y] = Euler(dydx, x0, xf, y0)









end
```

Heun's Method

We can modify the explicit Euler's Method slightly to derive the procedure for *Heun's Method*, a method called a *predictor-corrector (PC) method*. You'll soon see why this name makes sense.

The error involved in Euler's Method is largely from the assumption that the function slope is constant over the step size, h. This assumption only works well when the step size we define is VERY small (think why?). Heun's Method attempts to account for the possibility that the *slope is not constant* over the step, h.

The concept is simple. Consider an ODE $\frac{dy}{dx} = f(x, y)$ at some point (x_i, y_i^{NS}):

- Use Euler's Method to make a *prediction* for the next point and call this predicted point (x_{i+1}, y_{i+1}^E).
- From this predicted point, compute the slope, $f(x_{i+1}, y_{i+1}^E)$.
- Heun's Method realizes that the slopes $f(x_i, y_i^{NS})$ and $f(x_{i+1}, y_{i+1}^E)$ is likely different, so it takes the *average* between them and *uses that average slope* as the slope at the current point, (x_i, y_i).
- Now, use this averaged slope to give you the *corrected* next point, (x_{i+1}, y_{i+1}^{NS}) to complete Heun's method.

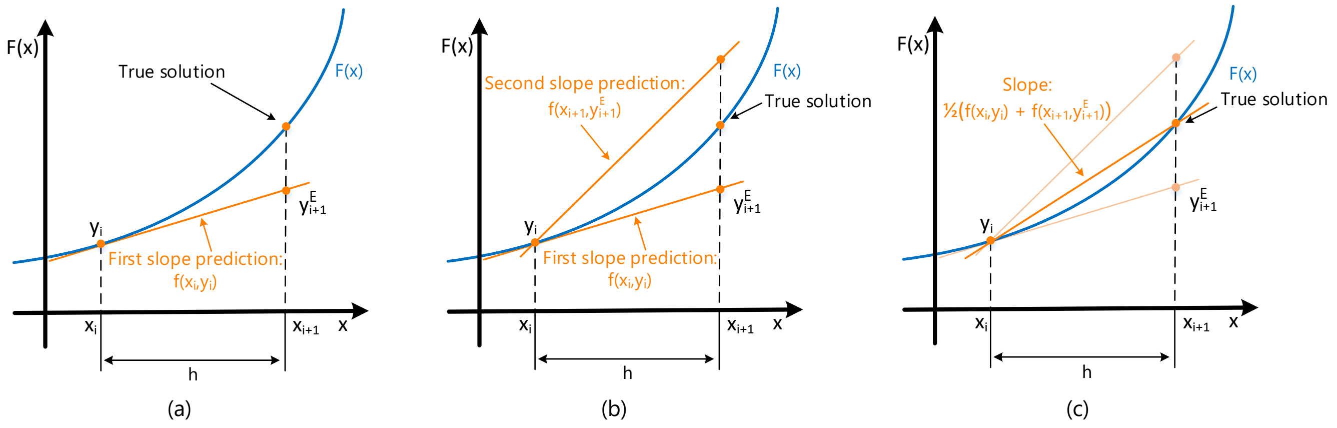


Figure 10: Diagram of Heun's method.

For Heun's method, the local error is $\mathcal{O}(h^3)$, while the global error is $\mathcal{O}(h^2)$.

Heun's Method

The following is the algorithm to integrate an ODE $\frac{dy}{dx} = f(x,y)$ on the interval $[x_1, x_{end}]$ using Heun's Method from an initial condition x_1 and $y_1^{NS} = y_1 = F(x_1)$:

1. Set the given initial point as the first point of the numerical solution. Select a step size h and an iteration counter $i = 1$.

2. Evaluate the slope at the current point, $f(x_i, y_i^{NS})$:

$$f(x_i, y_i^{NS}) \triangleq \left.\frac{dy}{dx}\right|_{(x_i, y_i^{NS})}$$

3. Determine the Euler's estimate, (x_{i+1}, y_{i+1}^E), using the slope from the current point and the step size h:

$$x_{i+1} = x_i + h$$
$$y_{i+1}^E = y_i^{NS} + f(x_i, y_i^{NS})(h)$$

4. Evaluate the slope at Euler's estimate, $f(x_{i+1}, y_{i+1}^E)$:

$$f(x_{i+1}, y_{i+1}^E) \triangleq \left.\frac{dy}{dx}\right|_{(x_{i+1}, y_{i+1}^E)}$$

5. Use the average from steps 2 and 4 to find the Heun's method numerical solution, y_{i+1}^{NS}:

$$y_{i+1}^{NS} = y_i^{NS} + \frac{f(x_i, y_i^{NS}) + f(x_{i+1}, y_{i+1}^E)}{2}(h)$$

6.
```
IF $x_{i+1} = x_{end}$
        STOP
ELSE
        Update $i = i + 1$. Return to step 2.
```

Using the same ODE and conditions from our first Euler's Method example, Heun's Method can be seen to achieve much more accurate results:

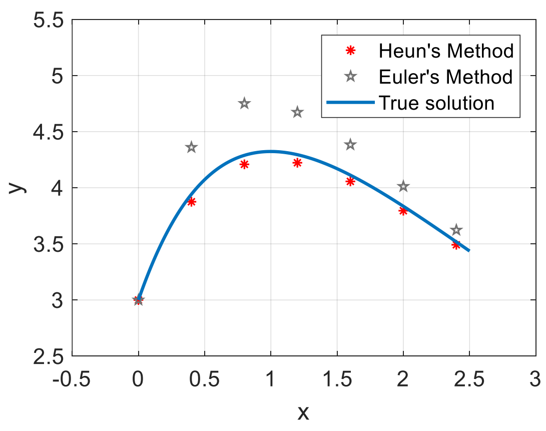


Figure 11: Using Heun's method with $h = 0.4$ to solve the ODE. The true solution, and Euler's method with $h = 0.4$, is shown.

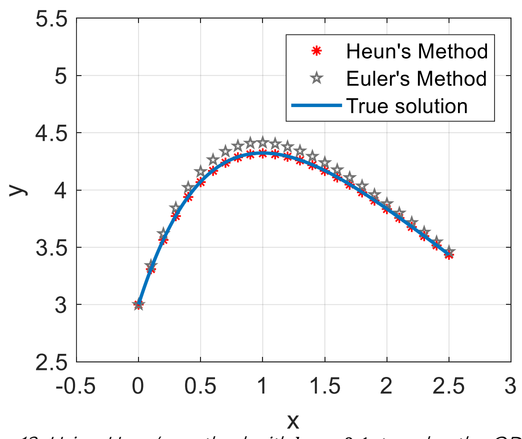


Figure 12: Using Heun's method with $h = 0.1$ to solve the ODE. The true solution, and Euler's method with $h = 0.1$, is shown.

Why don't we whip up an extension to our Euler's Method workshop using Heun's Method instead?

Workshop: Coding Heun's Method

The curve shown to the right is the solution to the IVP:

$$\frac{dy}{dx} = 2 + 2xy \qquad y(0) = 1$$

Modify the Euler Method code from the previous workshop to employ Heun's Method and compare their performances for this ODE.

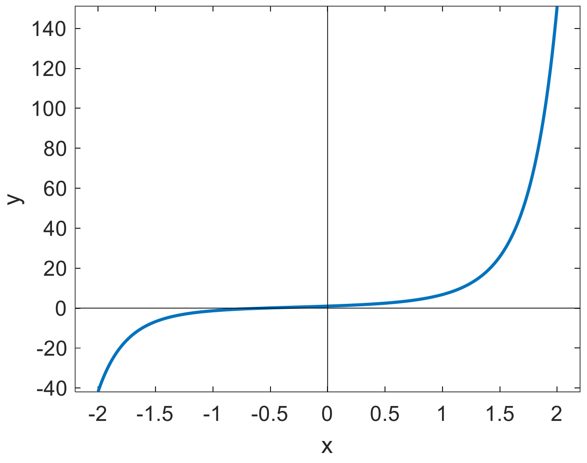


Figure 13: True Solution

```
function [x, y] = Heun(dydx, x0, xf, y0)











end
```

Question: So, why are methods like these named 'predictor-corrector' methods?

Answer: It is because these methods involve a 'prediction step' and a 'correction step'. As seen, Heun's Method used Euler's Method to *predict* what the next point WOULD be, (x_{i+1}, y_{i+1}^{E}), along with the slope at that predicted point. After averaging the slope at the current point, $f(x_i, y_i)$ and the slope at the predicted point, $f(x_{i+1}, y_{i+1}^{E})$, Heun's Method *went back* to (x_i, y_i^{NS}) and extrapolated that average slope to find (x_i, y_{i+1}^{NS}) as the *corrected* next point. All predictor-corrector methods work in a similar way!

Runge-Kutta Methods

Runge-Kutta (RK) methods are a family of similar (in framework) procedures to solve first order ODEs. Just like in Euler's and Heun's method, all RK variations utilize the following *single step method* to compute the next point of the function:

$$y_{i+1}^{NS} = y_i^{NS} + (slope) \times (h)$$

The difference in RK methods is how the *slope* is computed. RK methods use multiple points to get better estimates the slope of the function. An RK method of order N will use N points to determine the slope and will also have global error $\mathcal{O}(h^N)$.

Higher-order RK methods can reduce truncation error but require more FLOPs to find the slope.

Fourth Order Runge-Kutta (RK4)

Fourth Order Runge-Kutta (RK4) is the most common RK form, and uses 4 slope evaluations to determine the best overall slope/direction to take. Locally, RK4 is accurate to $\mathcal{O}(h^5)$ and is $\mathcal{O}(h^4)$ globally.

The derivation of RK4 is beyond us, let's just focus on the process it follows and how to implement it.

Various RK Methods NOT Covered

We're going to focus on the RK4 method here. RK4 is the basis of some pretty popular solvers such as `ode45` in `MATLAB`, so it is a good method to understand should you ever need to solve ODEs in the future.

There are several other flavours of RK used for stiff functions, (in `ode15s` for example) and other applications, but they all work essentially the same way. You just need to use the right tool for the job.

Fourth Order Classical Runge-Kutta (RK4)

The following is the algorithm to integrate an ODE $\frac{dy}{dx} = f(x,y)$ on the interval $[x_1, x_{end}]$ using Classical RK4 Method from an initial condition x_1 and $y_1^{NS} = y_1 = F(x_1)$:

1. Set the given initial point as the first point of the numerical solution. Select a step size h and an iteration counter $i = 1$.

2. Using a step size of h, evaluate the four slope approximations, K_1, K_2, K_3 and K_4, of the function at the current point (x_i, y_i^{NS}):

$$\begin{aligned} K_1 &= f(x_i, y_i^{NS}) \\ K_2 &= f\left((x_i + \frac{1}{2}h), (y_i^{NS} + \frac{1}{2}K_1 h)\right) \\ K_3 &= f\left((x_i + \frac{1}{2}h), (y_i^{NS} + \frac{1}{2}K_2 h)\right) \\ K_4 &= f\left((x_i + h), (y_i^{NS} + K_3 h)\right) \end{aligned}$$

3. Calculate the RK4 numerical solution to your next point, (x_{i+1}, y_{i+1}^{NS}), where

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

4. IF $x_{i+1} = x_{end}$

   STOP

   ELSE

   Update $i = i + 1$. Return to step 2.

RK4 is $\mathcal{O}(h^5)$ locally, and $\mathcal{O}(h^4)$ globally. Because K_i is used in the formula to calculate $K_{i+1} \rightarrow K_n$, the values of K_i cannot be solved simultaneously and must be solved in order from K_1 to K_4. Note that there are variations to RK4 methods that do exist! They follow the same format but use different constants to determine K and different weightings when summing all the K constants in step 3.

The below diagram helps to visualize the classical RK4 method in action:

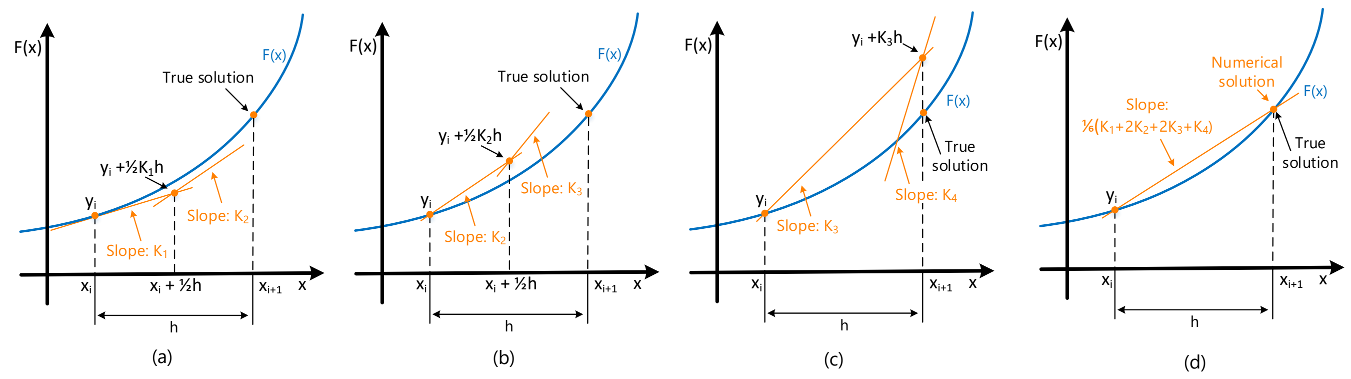


Figure 14. Diagram adapted from Gilat and Subramaniam, Numerical Methods for Engineers and Scientists, 2014. Page 412. In essence, we take the slope at x_i and use it to find a new point and test for slope there. Then, use our new slope to find another point – this continues until we have four slope approximations which are then combined to get us our FINAL slope/direction.

To demonstrate how strong of a method RK4 is, the ODE from our previous example: $\frac{dy}{dx} = -1.2y + 7e^{-0.3x}$ was solved with various h values and plotted in *Figure 15*-*Figure 17*.

Even using a large step size of 0.25, RK4 nearly replicates the true function. As h is increased, error is only concerning at the apex of the plot - once the function becomes steadier it returns to representing the true function well.

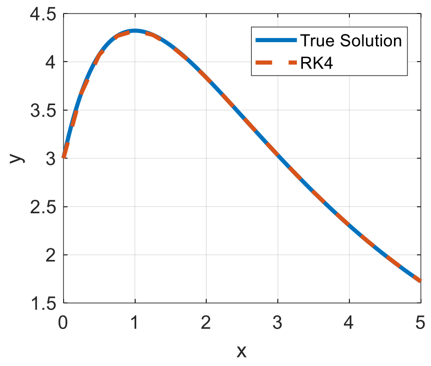


Figure 15: RK4 method with a step size of $h = 0.25$ *vs. true solution.*

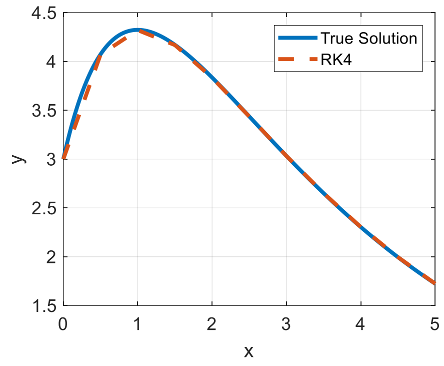


Figure 16: RK4 method with a step size of $h = 0.5$ *vs. true solution.*

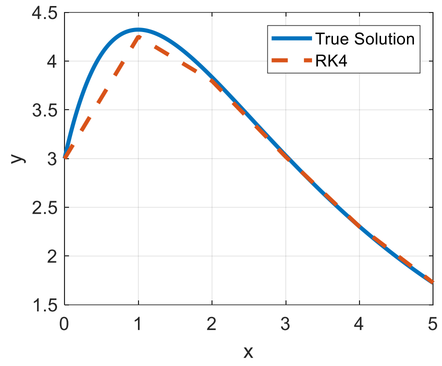


Figure 17: RK4 method with a step size of $h = 1$ *vs. true solution.*

Shown below for your reference is a plot of classical RK4, Heun's method and Euler's method all together for comparison using a step size of 0.5.

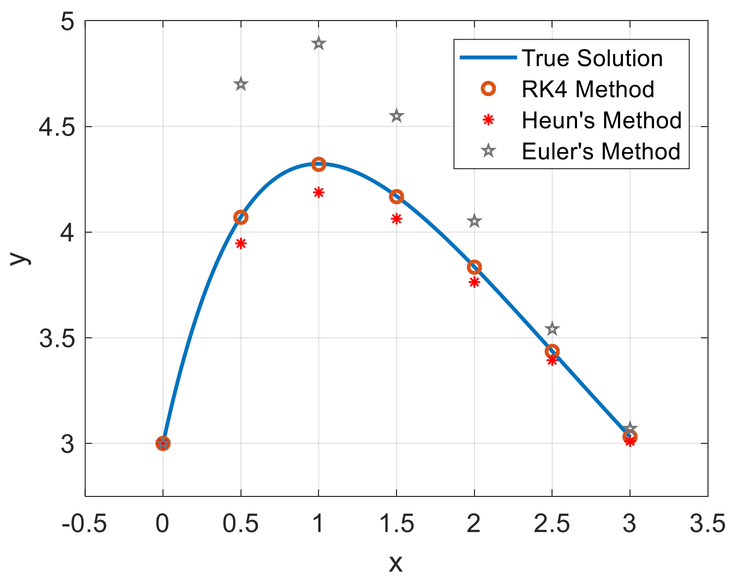
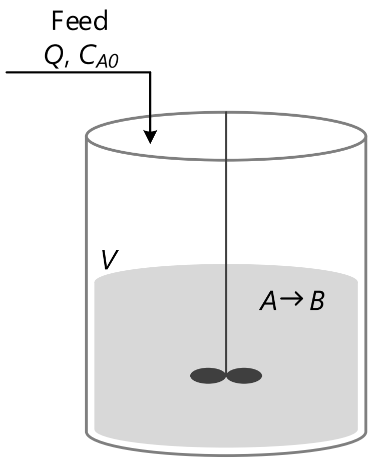


Figure 18: All ODE methods covered plotted on one plot for comparison.

Workshop: RK4 – From Computational Methods for Engineers, by J. Riggs, Page 298

Consider a semi-batch reactor. There is flow into the reactor while it's "active", but nothing is removed, so material builds up inside as the reaction proceeds. In this case, the reaction is $A \rightarrow B$, with a reaction rate, $r = kC_A^2$ where k is a constant and C_A is the concentration of A.

Below is the ODE modeling the change in moles of material A in the reactor with respect to time:



$$\frac{dn_A}{dt} = QC_{A_0} - \frac{kn_A^2}{Qt + V_0}$$

Initial moles of A:	$n_A(0)$	$= 50\ mol$
Concentration of A in Feed:	C_{A_0}	$= 1\ mol/L$
Reaction Constant:	k	$= 0.1\ \frac{L}{mol \cdot s}$
Feed flow rate:	Q	$= 10\ L/s$
Initial volume:	V_0	$= 50\ L$

$$\boxed{\frac{dn_A}{dt} = 10 - \frac{0.1n_A^2}{10t + 50}}$$

Draft a pseudo-code for RK4 and apply it to this example to determine the steady-state number of moles of A (n_A) and how long it takes to reach that steady-state using $h = 20$ seconds. Compare the results to those obtained by `ode45` in `MATLAB`.

```
function [x, y] = RK4(dydx, x0, xf, y0)













end
```

Conclusion and Summary

There are many problems in chemical engineering and other fields that use ODEs to describe various relationships. Once formulated, numerical methods for ODEs allow one to efficiently solve a close approximation of the desired function – a task not easily performed, and sometimes impossible to do analytically.

In this module we have covered the following:

- An introduction and overview of differential equations.
- Three different numerical methods for solving ODEs.
    - Euler's method
    - Heun's method
    - Runge-Kutta method

Next up: Systems of ODEs and Step Size

Appendix A: Explicit vs Implicit Numerical Methods

The method we covered is known as the *explicit* Euler's Method. The *Implicit Euler's Method*, sometimes called the "Backwards Euler's Method" is slightly different. It still assumes a constant slope over a small interval, h, however, that constant slope is taken as slope at the *endpoint* of the interval (*i.e.* $y_{i+1} = y_i + f(x_{i+1}, y_{i+1})(h)$).

The equation above has unknown terms *on both sides of the equation*, and cannot be solved explicitly – that is why it is called implicit, and must itself be solved using numerical methods from our previous units! All implicit methods are like this, using future values to help calculate future values.

Implicit methods aren't as simple to code as explicit methods, however, are typically more stable/reliable. To achieve the same accuracy, larger step sizes can be used in implicit than in explicit methods, and thus computational speed and efficiency has potential to increase! 🏃