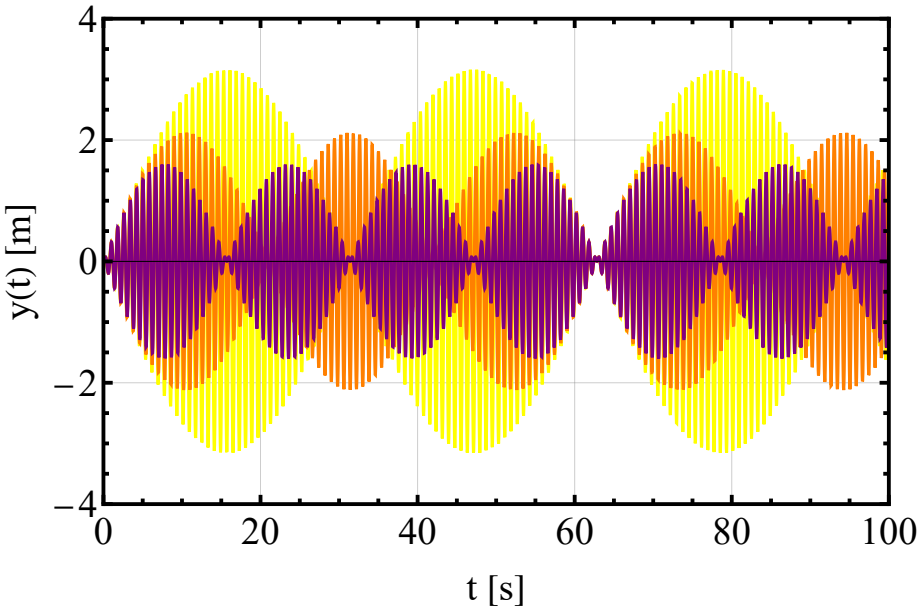
NAZARBAYEV UNIVERSITY

SCHOOL OF ENGINEERING AND DIGITAL SCIENCES

Differential Equations & Linear Algebra with Wolfram Mathematica

Student Guidebook | 1st Edition



Balnur Zhaidarbek Aruzhan Tleubek Yanwei Wang

Nazarbayev University - Chemical Engineering, July 2022

Preface

We are pleased to present this first edition of the "Differential Equations and Linear Algebra with Wolfram Mathematica" student guidebook. This book is very comprehensive, but we hope it does not hold it back from being an enjoyable read.

This book is written primarily for students enrolled in the course "Engineering Mathematics III (Differential Equations and Linear Algebra)" (ENG200) here at Nazarbayev University (NU). This is a common compulsory course offered to all 2nd-year engineering students. There is a Computational Lab session in this course where students are expected to practice what they have learned in the theoretical sessions on computers with Wolfram Mathematica. Such a design of the course was credited to Prof. V. Zarikas (now at the University of Thessaly, Greece), who was the course leader when YW taught this course in Fall 2020 and Fall 2021. We want to thank Prof. Zarikas for selflessly sharing his course materials, which we have benefited from when developing this book.

Every topic has been summarized and supported by a sufficient number of solved problems. The present book has been designed to equip young engineering students with as much knowledge on all topics as is desirable from the point of view of the ENG 200 learning outcomes. Efforts have been made to make Differential equations and Linear algebra, the fundamental subjects in every engineering curriculum, more interesting and engaging with the help of Wolfram Mathematica language. In addition to the above-mentioned math skills, the book helps to learn a new programming software, Wolfram Mathematica. Just like learning any new skill, learning Wolfram Mathematica takes time, effort, and dedication. Therefore, we believe this journey will also benefit our readers to become self-disciplined learners.

BZ wishes to express her appreciation to Dr. YW, the instructor for the ENG200 course at NU. She is grateful to him for all the knowledge gained in the ENG200 course and for awakening her interest in learning the newly introduced Wolfram Mathematica tool/language. BZ hopes that this book, developed under the supervision of YW, will help the reader learn the basics of Wolfram Mathematica and use their acquired skills for further work/research. In addition, BZ thanks Dr. Devendra Kapadia for developing the interactive course "Introduction to Linear Algebra" for learning linear algebra using the Wolfram Language used in the preparation of this book and strongly encourages students to take a look at other Wolfram U Interactive Courses listed in the References section of the book.

AT would like to express her gratitude to Dr. YW for his invaluable advice, continuous support, and patience both during the ENG 200 course and the book writing process. Without YW's encouragement and supervision, this book would not have been possible. AT also would like to thank her ENG 200 coursemates and friends for a cherished time spent together solving

rigorous math problems and learning new skills in class and social settings.

YW would like to express his sincere gratitude to Prof. H. Tobita (University of Fukui, Japan), who introduced Wolfram Mathematica to him in Fall 2002. Since then, YW has been in love with this fantastic tool/language. Life would be different if he didn't know about Wolfram Mathematica, and this book would certainly not have been possible. YW would also like to express his gratitude to students enrolled in the ENG200 course. YW has benefited from close interactions with students since he started to teach this course in Fall 2020. The two coauthors (BZ and AT) were also students enrolled in ENG200 in Fall 2021. This book would not have been finished now without those two brilliant and hardworking students/coauthors.

YW would like to thank Nazarbayev University for funding this book under the Social Policy Grant.

We acknowledge that this version of the book might have uncorrected mistakes, spelling/grammatical errors, and ambiguities. We are trying to eliminate them in a 2nd version (to be released in July 2023). We are grateful to the readers, students, instructors, or anyone who somehow encountered this book if they send us their valuable feedback so that we may make further improvements in future editions.

Balnur Zhaidarbek (BZ)	Email: balnur.zhaidarbek@nu.edu.kz
Aruzhan Tleubek (AT)	Email: aruzhan.tleubek@nu.edu.kz
Yanwei Wang (YW)	Email: yanwei.wang@nu.edu.kz

Department of Chemical and Materials Engineering
School of Engineering and Digital Sciences
Nazarbayev University

Table of Contents

Week 0: Preliminary

Introduction to Wolfram Mathematica

The secret to getting ahead is getting started. --- Mark Twain

Table of Contents

Commands list

- N
- Table
- D
- Integrate
- Solve
- Coefficient
- ClearAll
- Clear
- Quit
- Plot
- Plot3D
- ParametricPlot
- StreamPlot
- VectorPlot
- ContourPlot
- DSolve
- Manipulate
- Sound

0.1

Prerequisites

0.1.1

To Begin With

There a few things to keep in mind when using Mathematica.

☑ When using a PC, in order to **execute a command** you must hit **Shift-Enter.**

☑ Mathematica is **Case-SenSitive** (AA is not the same as aA), so be careful about what you type.

☑ All **built-in Mathematica functions** are spelled out and **capitalized**, such as Table, ListPlot, IntegerPart, Plot, Sin, Cos, etc.

☑ The **parameters inside a function** are always enclosed with **square brackets**, **[ ]**.

```
In[ ]:= log
Out[ ]=
log

In[ ]:= Log[10]
Out[ ]=
Log[10]
```

☑ You can use a **semicolon** (;) at the end of a line if you want to perform the action, but **don't want to see the output.**

☑ Don't forget about the copy and paste commands. This will be useful if you have to type similar commands and don't want to have to retype the entire command.

☑ In Mathematica, it is important to distinguish between parentheses (), brackets [], and braces {}:

- **Parentheses ()**: Used to group mathematical expressions, such as (3 + 4) / (5 + 7).

```
In[ ]:= (3 + 4) / (5 + 7)
Out[ ]=
7/12
```

- **Brackets []**: Used when calling functions, such as N[Log[10]].

```
In[ ]:= N[Log[10]]
Out[ ]=
2.30259
```

- **Braces {}**: Used when making lists, such as {i,1,20}.

```
In[ ]:= Table[i, {i, 1, 20}]
Out[ ]=
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20}
```

☑ In Mathematica, there are **four types of equals**: = , := , == , and === .

- To **define a variable** to store it in memory, use **=**. For example, to define z to be 3, write $z = 3$. Syntax for setting a variable is ***x = …*** (definition of a variable).
- You use **==** to **check for equality**. For example, $1 - 1 == 0$ will evaluate to True and $1 == 0$ will evaluate to False.
- You use **:=** to **define your own command**. (This is advanced.)
- You will likely not use **===** in this class (URL).

0.1.2

Basic Algebra and Calculus

☑ Use ^ (or hit CTRL+6) to put something to a **power**.

```
In[ ]:= Table[n^2, {n, 10}]
Out[ ]=
{1, 4, 9, 16, 25, 36, 49, 64, 81, 100}

In[ ]:= Table[n^2, {n, 0, 10}]
Out[ ]=
{0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100}

In[ ]:= Table[n^2, {n, 0, 10, 2}]
Out[ ]=
{0, 4, 16, 36, 64, 100}
```

☑ ***pi*** is **Pi**, ***e*** is **E** and ***sqrt(-1)*** is **I**.

☑ If you want to see the **numerical approximation** to a fraction or irrational number, use the function **N**.

For example, to find the decimal represenation of ***pi***, write N[Pi].

```
In[ ]:= N[Pi]
Out[ ]=
3.14159

In[ ]:= N[E]
Out[ ]=
2.71828

In[ ]:= Sqrt[-1]
Out[ ]=
i
```

☑ Use **E^x** or **Exp[x]** to represent the function e^x.

☑ To take the **derivative of a function**, use **D** and specify the derivative with respect to which variable.

For instance, find the first derivative of $x^2 + 3x$.

```
In[ ]:= D[x^2 + 3 x, x]
Out[ ]=
3 + 2 x
```

☑ To take the **integral of a function**, use **Integrate** and specify the integral with respect to which variable.

For instance, find the integral of $x^2 + 3x$.

In[ ]:=
```
Integrate[x^2 + 3 x, x]
```

Out[ ]=

$$\frac{3\,x^2}{2} + \frac{x^3}{3}$$

☑ To **solve for the roots** of $a\,x^2 + b\,x + c = 0$ symbolically, use Solve[a x^2+ b x+c == 0, x].

☑ Notice the **double equals sign** (==). (Mathematica is searching for when the expression is True.)

In[ ]:=
```
Solve[a x^2 + b x + c == 0, x]
```

Out[ ]=

$$\left\{\left\{x \to \frac{-b - \sqrt{b^2 - 4\,a\,c}}{2\,a}\right\}, \left\{x \to \frac{-b + \sqrt{b^2 - 4\,a\,c}}{2\,a}\right\}\right\}$$

☑ **Coefficient**[(1 + x)^10, x^3] gives the coefficient of x^3 in the expansion of $(1 + x)^{10}$.

In[ ]:=
```
Coefficient[(1 + x)^10, x^3]
```

Out[ ]=

```
120
```

0.1.3

Some of the Basic Operations

- Sqrt[x]
- Exp[x]
- Log[x]
- Log[b, x] (logarithm with base b)
- Sin[x]
- Cos[x]
- Tan[x]
- ArcSin[x]
- ArcCos[x]
- ArcTan[x]
- Sinh[x]
- n! (factorial)
- Abs[x] (absolute value)
- Round[x] (closer integer)
- Floor[x] (integer part)

- Mod[n, m]
- Random[ ]
- Max[x, y, …]
- Min[x ,y, …]

0.2

Help Options

0.2.1

Help Browser

To access the help browser, go the **Help** menu and choose **Wolfram Documentation**. If you want to know about a particular function in Mathematica, select it and then go to **Find Selected Function** or simply hit the **F1** key on your keyboard.

There are a lot of fun examples on the **Wolfram Demonstrations Project** (URL). You may also share your work with the world. Getting started is simple.

0.2.2

Text-based Help

The **Question Mark function ?** allows you to get basic information about a particular Mathematica function.

In[ ]:= `? /.`

Out[ ]=

Symbol

expr /. *rules* or ReplaceAll[*expr*, *rules*] applies a rule or list of rules in an attempt to transform each subpart of an expression *expr*.

ReplaceAll[*rules*] represents an operator form of ReplaceAll that can be applied to an expression.

For example, suppose we want to find out how to use the *derivative function D*, the q mark function **?** yields:

In[ ]:= **? D**

Out[ ]=

Symbol

D[*f*, *x*] gives the partial derivative $\partial f/\partial x$.

D[*f*, {*x*, *n*}] gives the multiple derivative $\partial^n f/\partial x^n$.

D[*f*, *x*, *y*, …] gives the partial derivative $\cdots (\partial/\partial y)(\partial/\partial x) f$.

D[*f*, {*x*, *n*}, {*y*, *m*}, …] gives the multiple partial derivative $\cdots (\partial^m/\partial y^m)(\partial^n/\partial x^n) f$.

D[*f*, {{x_1, x_2, …}}] for a scalar *f* gives the vector derivative $(\partial f/\partial x_1, \partial f/\partial x_2, \ldots)$.

D[*f*, {*array*}] gives an array derivative.

The **double question mark ??** gives the same information as ? but also gives information about attributes and options.

In[ ]:= **?? D**

Out[ ]=

Symbol

D[*f*, *x*] gives the partial derivative $\partial f/\partial x$.

D[*f*, {*x*, *n*}] gives the multiple derivative $\partial^n f/\partial x^n$.

D[*f*, *x*, *y*, …] gives the partial derivative $\cdots (\partial/\partial y)(\partial/\partial x) f$.

D[*f*, {*x*, *n*}, {*y*, *m*}, …] gives the multiple partial derivative $\cdots (\partial^m/\partial y^m)(\partial^n/\partial x^n) f$.

D[*f*, {{x_1, x_2, …}}] for a scalar *f* gives the vector derivative $(\partial f/\partial x_1, \partial f/\partial x_2, \ldots)$.

D[*f*, {*array*}] gives an array derivative.

Documentation Web »
Options NonConstants → {}
Attributes {Protected, ReadProtected}
Full Name System`D

If you are trying to recall a function that has the word **Solve** in it then you can use asterisk * in conjunction with the word **Solve** , as shown below:

In[ ]:=
```
? *Solve*
```

Out[ ]=

System`

AsymptoticDSolveValue	DSolve	LinearSolve	NDSolveValue	RiccatiSolve	SolveDelayed
AsymptoticRSolveValue	DSolveChangeVariables	LinearSolveFunction	NSolve	RSolve	SolveValues
AsymptoticSolve	DSolveValue	LyapunovSolve	NSolveValues	RSolveValue	
DiscreteLyapunovSolve	FrobeniusSolve	MainSolve	ParametricNDSolve	Solve	
DiscreteRiccatiSolve	KnapsackSolve	NDSolve	ParametricNDSolveValue	SolveAlways	

0.3

How to | Clear User Defined Symbols

0.3.1

ClearAll[“Global`*”]

When you set a value to a symbol, that value will be used for the symbol for the entire Wolfram System session. Since symbols no longer in use can introduce unexpected errors when used in new computations, clearing your definitions is very desirable.

> ClearAll[symb1,symb2,…] clears all values, definitions, attributes, messages, and defaults associated with symbols.

To clear all definitions of quantities you’ve introduced in a Mathematica session so far, type: ClearAll[“Global`*”].

In[ ]:=
```
ClearAll["Global`*"]
```

Assign values to two symbols (x and y) and observe their sum:

In[ ]:=
```
x = 5; y = 7; x + y
```

Out[ ]=
```
12
```

Use Clear to clear the definitions for x and y:

In[ ]:=
```
Clear[x, y]
```

Read this page (**How to | Clear My Definitions** | URL) for more details.

0.3.2

Quit[]

To clear all definitions or to reclaim resources used by the kernel, you may want to restart it. There are at least two options.

- **Option 1**: Quit the kernel by choosing Evaluation ▶ Quit Kernel ▶ "kernel name", where "kernel name" is typically "Local".
- **Option 2**: Quit the kernel by evaluating Quit. Quit[] (URL) terminates a Wolfram Language kernel session. Quit[] quits only the Wolfram Language kernel, not the front end. To quit a notebook front end, choose the Quit menu item. All kernel definitions are lost when the kernel session terminates.

In[ ]:=
```
Quit[]
```

0.4

Create Plots

0.4.1

Defining a Function

There are many built-in function in Wolfram Language and some of them were introduced in previous sections. This section will focus on learning how to define our own functions in Mathematica.

☑ Syntax for defining a function that takes any single argument is $f[x\_] := \ldots$ (definition of a function).

For example, the command for defining a function $f(x) = x^2$ is

In[ ]:=
```
f[x_] := x^2
```

⚠ Notice the **underscore** "_" to the right of the variable *y* and/or on the left of "equality". If the character underscore was not used, then the function is only defined for this particular symbol of the variable.

In[ ]:=
```
Clear[f]
f[x] = x^3/2;
```

In[ ]:=
```
f[5]
```

Out[ ]=
```
f[5]
```

⚠ The use of **equality symbol** ":=" in the definition of the function, i.e., the assignment

with a delay(Set Delayed) is most othen the correct choice. The choice of direct, (Set) assignment "=" can lead to undesirable results.

```
In[ ]:= a = 3;
        setDelayed[x_] := x + y + a^2;
        set[x_] = x + y + a^2;

In[ ]:= setDelayed[x]
Out[ ]=
        9 + x + y

In[ ]:= set[x]
Out[ ]=
        9 + x + y

In[ ]:= a = 4;
        setDelayed[x]
Out[ ]=
        16 + x + y

In[ ]:= set[x]
Out[ ]=
        9 + x + y
```

☑ The argument of a function may be a number or any complex algebraic expression.

```
In[ ]:= f[4]
Out[ ]=
        16

In[ ]:= f[a^2 + a + 1]
Out[ ]=
        (1 + a + a^2)^2
```

☑ It is also possible to use a function in a **calculation.**

Define a function $q(y) = y - \frac{1}{2} + c_1 e^{-2y}$:

```
In[ ]:= q[y_] := y - 1/2 + C1 * Exp[-2 y];
```

Find the first derivative of this function:

```
In[ ]:= D[y - 1/2 + C1 * Exp[-2 y], y]
Out[ ]=
        1 - 2 C1 ℯ^(-2 y)
```

```
In[ ]:= D[q[y], y]
Out[ ]=
        1 - 2 C1 ℯ^(-2 y)
```

Find the second derivative of the function:

```
In[ ]:= D[y - 1/2 + C1 * Exp[-2 y], {y, 2}]
Out[ ]=
        4 C1 ℯ^(-2 y)

In[ ]:= D[q[y], {y, 2}]
Out[ ]=
        4 C1 ℯ^(-2 y)
```

☑ The **Question Mark function ?** allows you to get the definition of *f*.

```
In[ ]:= ? q
Out[ ]=
```

Symbol

Global`q

Definitions

q[y_] := y - 1/2 + C1 Exp[-2 y]

Full Name Global`q

☑ The **name of a function** i.e. **f**, is just a symbol for Mathematica. Thus **do not** define a function with **capital letter** to avoid the confusion with other built-in Mathematica functions. Also this symbol must not have been previously used for definition of another element (variable, table, etc.).

☑ Function in Mathematica can have more than one argument. So we can define **multiple variables function.**

```
In[ ]:= product[x_, y_] := x * y;

In[ ]:= 1 + product[2, 3]
Out[ ]=
        7
```

☑ If later you will give a new definition to the function, the latter definition is the one that applies while the previous is canceled.

```
In[ ]:= product[x_, y_] := 1 + x * y
```

In[ ]:=
```
product[2, 3]
```

Out[ ]=

7

0.4.2

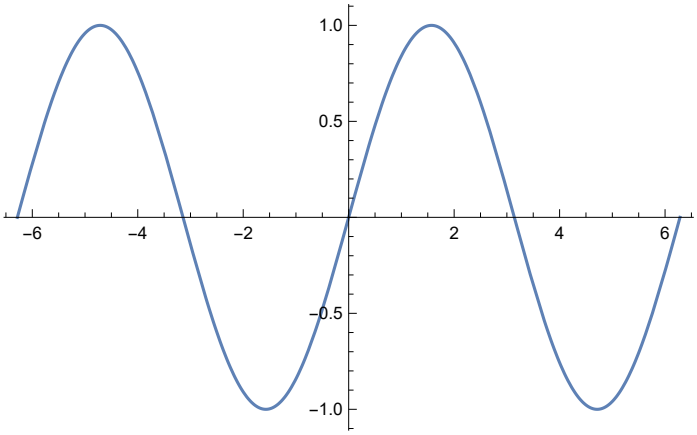
Graph of a Function of One Variable

The command for plotting a functions of one variables is

Plot[*function*, {*variable, lower bound, upper bound*}]

In[ ]:=
```
Plot[Sin[x], {x, -2 Pi, 2 Pi}]
```

Out[ ]=



0.4.3

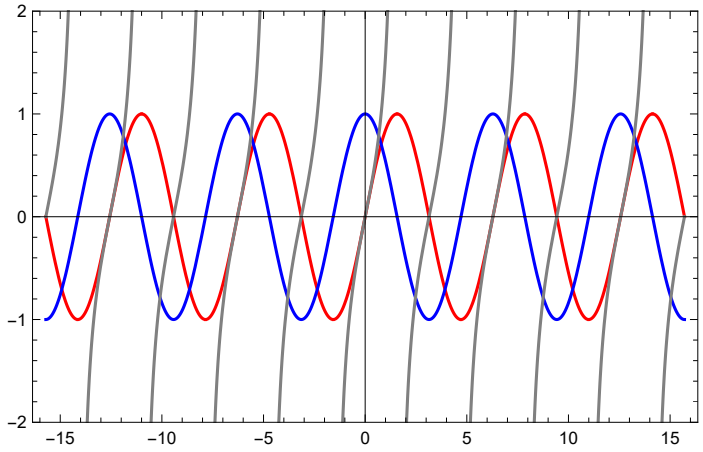
Multiple Functions on a Graph

To include two functions on the same graph, we simply write the Plot command using two functions which slip with the "**,**".

Plot[{ f_x, f_y , …}, {*variable, lower bound, upper bound*}]

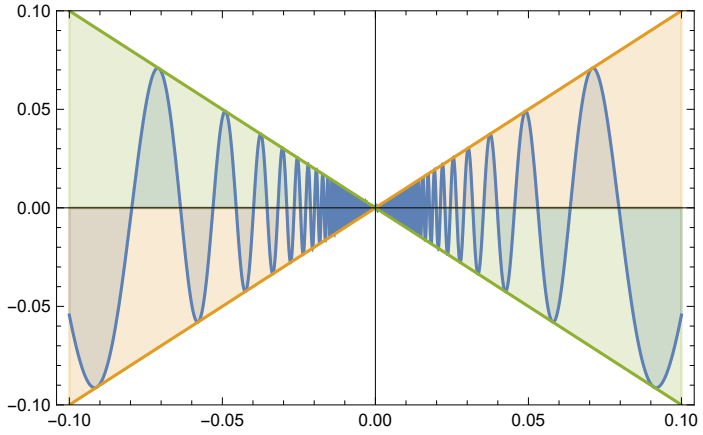
In[ ]:=
```
Plot[{Sin[x], Cos[x], Tan[x]}, {x, -5 Pi, 5 Pi},
 PlotRange → {-2, 2}, Frame → True, PlotStyle → {Red, Blue, Gray}]
```

Out[ ]=



In[ ]:=
```
Plot[{x * Sin[1 / x], x, -x}, {x, -0.1, 0.1}, PlotRange → 0.1,
 Filling → Axis, Frame → True, AspectRatio → 1 / GoldenRatio]
```

Out[ ]=



0.4.4

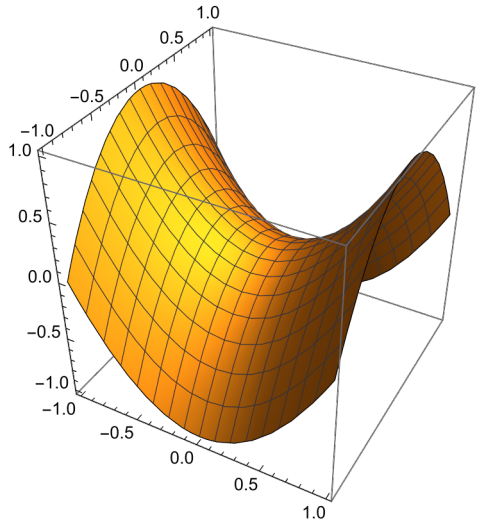
Graph of a Function of Two Variables

The relative command for functions of two variables is

> ***Plot3D[function, {variable_ 1, lower bound, upper bound}, {variable_ 2, lower bound, upper bound}]***

In[ ]:=
```
Plot3D[x^2 - y^2, {x, -1, 1}, {y, -1, 1}, BoxRatios → {1, 1, 1}, ImageSize → {270, 270}]
```

Out[ ]=



0.4.5

Parametric Plots

The relative command for making a parametric plot is

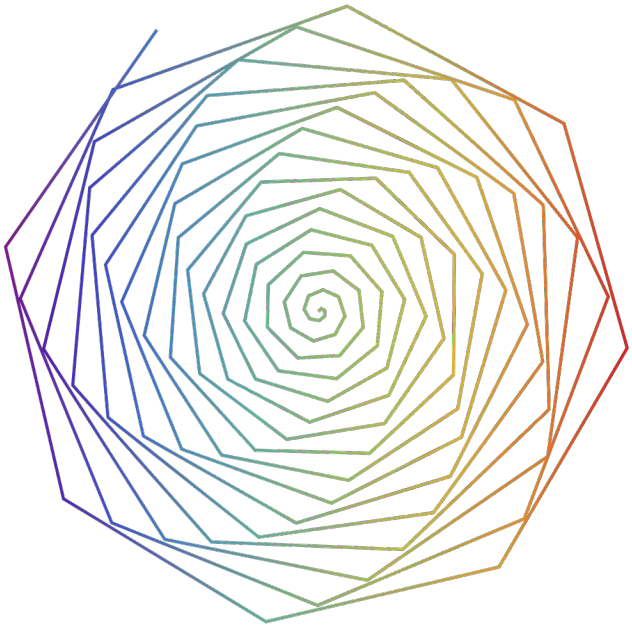
ParametricPlot[{ f_x, f_y }, { t, t_{min} , t_{max} }]

The relative command for plotting several parametric curves together is

ParametricPlot[{{ f_x, f_y }, { g_x, g_y }, ...}, { t, t_{min} , t_{max} }]

In[ ]:=
```
ParametricPlot[{u * Sin[u], u * Cos[u]}, {u, 0, 100},
 PlotPoints → 125, Axes → False, MaxRecursion → 0, ColorFunction → "Rainbow"]
```

Out[ ]=



0.5

DSolve Command

The **DSolve** Command is used to solve differential equations, list of differential equations, and a partial differential equations.

In[ ]:=
```
? DSolve
```

Out[ ]=

Symbol

DSolve[*eqn*, *u*, *x*] solves a differential equation for the function *u*, with independent variable *x*.

DSolve[*eqn*, *u*, {*x*, x_{min}, x_{max}}] solves a differential equation for *x* between x_{min} and x_{max}.

DSolve[{eqn_1, eqn_2, …}, {u_1, u_2, …}, …] solves a list of differential equations.

DSolve[*eqn*, *u*, {x_1, x_2, …}] solves a partial differential equation.

DSolve[*eqn*, *u*, {x_1, x_2, …} ∈ Ω] solves the partial differential equation *eqn* over the region Ω.

For example, find the general solution to the given ODE: $y' = -2\,x\,y$.

In[ ]:=
```
ClearAll["Global`*"]
```

In[ ]:=
```
DSolve[y'[x] == -2 x y[x], y[x], x]
```

Out[ ]=

$$\{\{y[x] \to e^{-x^2} C_1\}\}$$

Find the particular solution to the same ODE with inital consition: $y(0) = 1.8$.

In[ ]:=
```
solution = DSolve[{y'[x] == -2 x y[x], y[0] == 1.8}, y[x], x]
```

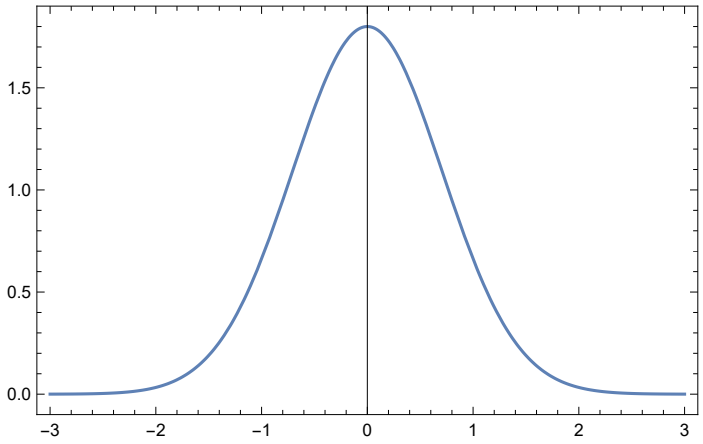
Out[ ]=

$$\{\{y[x] \to 1.8\, e^{-x^2}\}\}$$

Plot its graph:

In[ ]:=
```
Plot[y[x] /. solution, {x, -3, 3}, Frame → True]
```

Out[ ]=



0.6

How to | Visualize the Direction Field

0.6.1

Stream Plots

In[ ]:=
```
? StreamPlot
```

Out[ ]=

Symbol

StreamPlot[$\{v_x, v_y\}$, $\{x, x_{min}, x_{max}\}$, $\{y, y_{min}, y_{max}\}$]

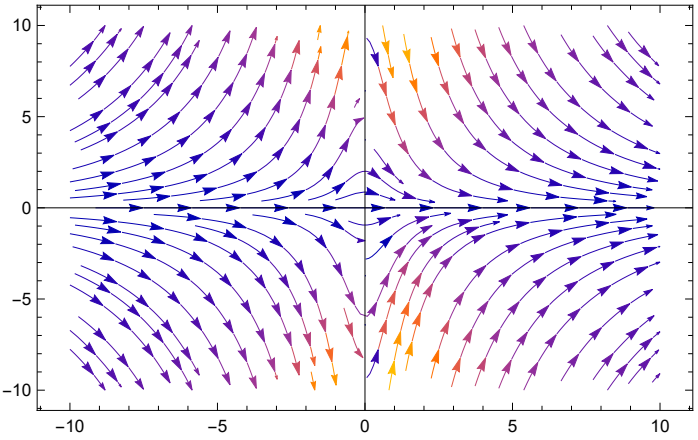
generates a stream plot of the vector field $\{v_x, v_y\}$ as a function of x and y.

StreamPlot[$\{\{v_x, v_y\}, \{w_x, w_y\}, \ldots\}$, $\{x, x_{min}, x_{max}\}$, $\{y, y_{min}, y_{max}\}$] generates plots of several vector fields.

StreamPlot[…, $\{x, y\} \in reg$] takes the variables $\{x, y\}$ to be in the geometric region reg.

In[ ]:=
```
f1[x_, y_] := -(2 x * y)/(1 + x^2);
plot1 = StreamPlot[{1, f1[x, y]}, {x, -10, 10},
  {y, -10, 10}, Frame → True, Axes → True, AspectRatio → 1 / GoldenRatio]
```

Out[ ]=



0.6.2

Vector Plots

In[ ]:=
```
? VectorPlot
```

Out[ ]=

Symbol

VectorPlot[{v_x, v_y}, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}]

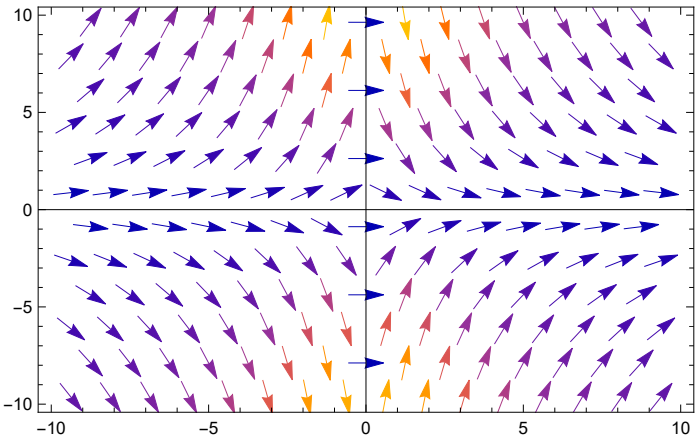
generates a vector plot of the vector field {v_x, v_y} as a function of x and y.

VectorPlot[{{v_x, v_y}, {w_x, w_y}, …}, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}] plots several vector fields.

VectorPlot[…, {x, y} ∈ reg] takes the variables {x, y} to be in the geometric region reg.

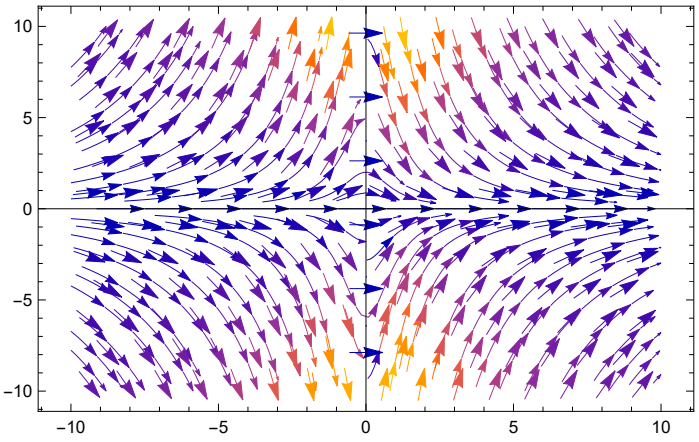
In[ ]:=
```
plot2 = VectorPlot[{1, f1[x, y]}, {x, -10, 10},
  {y, -10, 10}, Frame → True, Axes → True, AspectRatio → 1 / GoldenRatio]
```

Out[ ]=



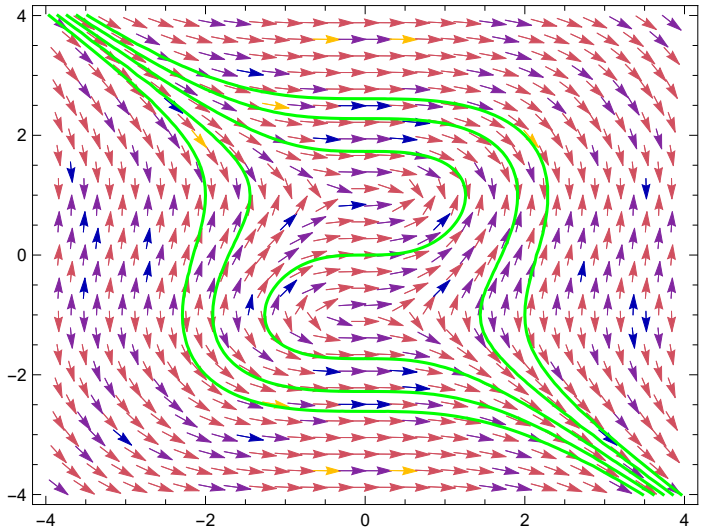
In[ ]:=
```
Show[plot1, plot2]
```

Out[ ]=



In[ ]:=
```
f2[x_, y_] := x^2 / (1 - y^2);
Show[VectorPlot[{1, f2[x, y]} / Sqrt[1 + f2[x, y]^2], {x, -4, 4},
  {y, -4, 4}, VectorScale → 0.03, VectorPoints → Fine, VectorStyle → "Arrow"],
 Table[ContourPlot[-x^3 + 3 y - y^3 == c, {x, -4, 4}, {y, -4, 4}, ContourStyle → Green],
  {c, {-10, -5, 0, 5, 10}}], AspectRatio → 3 / 4]
```

Out[ ]=



0.6.3

Contour Plots

In[ ]:=
```
? ContourPlot
```

Out[ ]=

Symbol

ContourPlot[f, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}] generates a contour plot of f as a function of x and y.

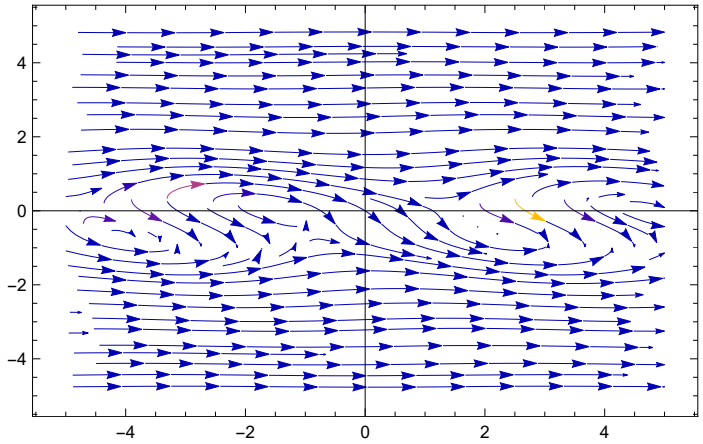
ContourPlot[f == g, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}] plots contour lines for which f = g.

ContourPlot[{f_1 == g_1, f_2 == g_2, …}, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}] plots several contour lines.

ContourPlot[…, {x, y} ∈ reg] takes the variables {x, y} to be in the geometric region reg.

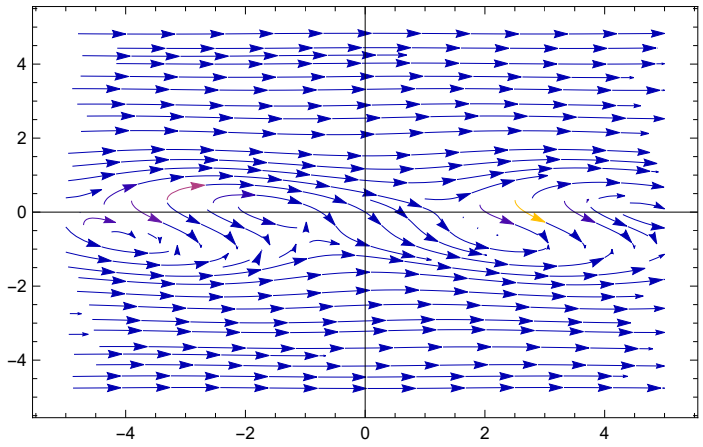
In[ ]:=
```
f3[x_, y_] := -Cos[x + y] / (3 y^2 + 2 y + Cos[x + y]);
p3 = StreamPlot[{1, f3[x, y]}, {x, -5, 5},
  {y, -5, 5}, Frame → True, Axes → True, AspectRatio → 1 / GoldenRatio]
```

Out[ ]=



In[ ]:=
```
p4 = Show[p3,
  Table[ContourPlot[Sin[x + y] + y^3 + y^2 == c, {x, -6, 6}, {y, -5, 5}, ContourStyle → Green],
   {c, {-4, -3, -2, -1, 0, 1, 2, 3, 4, 8, 16, 32, 64}}]]
```

Out[ ]=



0.7

More to Explore

0.7.1

Animation

In[ ]:=
```
? Animate
```

Out[ ]=

Symbol

Animate[*expr*, {*u*, u_{min}, u_{max}}] generates an animation of *expr* in which *u* varies continuously from u_{min} to u_{max}.

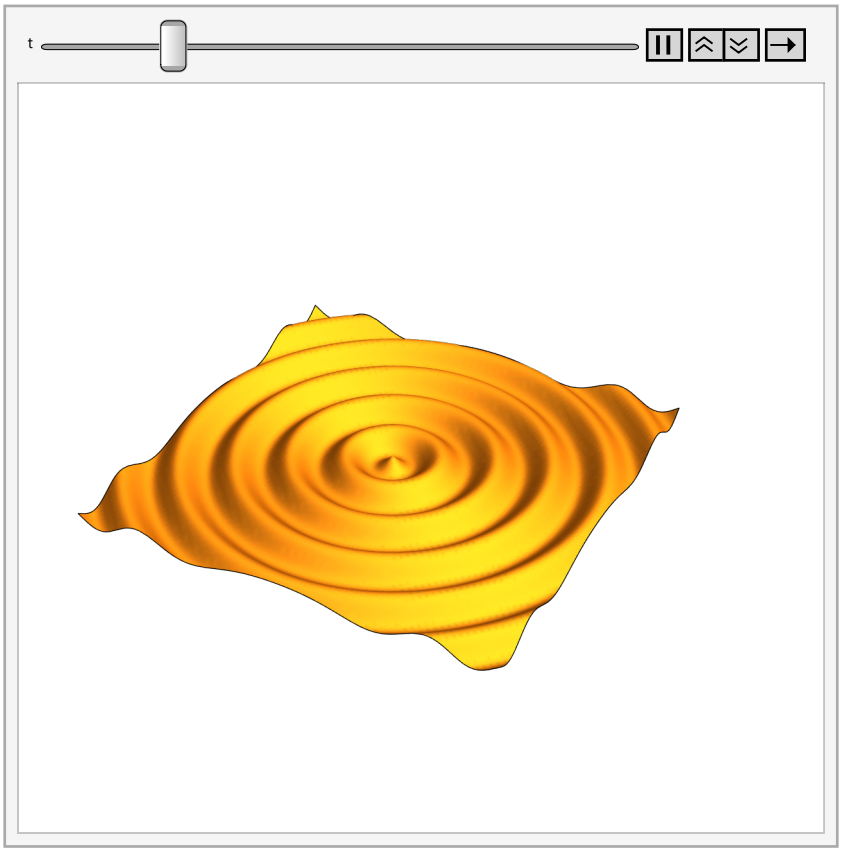
Animate[*expr*, {*u*, u_{min}, u_{max}, *du*}] takes *u* to vary in steps *du*.

Animate[*expr*, {*u*, {u_1, u_2, …}}] makes *u* take on discrete values u_1, u_2, ….

Animate[*expr*, {*u*, …}, {*v*, …}, …] varies all the variables *u*, *v*, ….

In[ ]:=
```
Animate[Plot3D[Sin[Sqrt[x^2 + y^2] + 2 * Pi * t], {x, -8 * Pi, 8 * Pi}, {y, -8 * Pi, 8 * Pi},
  PlotRange → 10, PlotPoints → 50, AspectRatio → 1,
  Boxed → False, Mesh → None, Axes → False], {t, 0, 2}, ControlPlacement → Top]
```

Out[ ]=



0.7.2

Interactive Manipulation

In[ ]:= **? Manipulate**

Out[ ]=

Symbol

Manipulate[*expr*, {*u*, u_{min}, u_{max}}] generates a version of

expr with controls added to allow interactive manipulation of the value of *u*.

Manipulate[*expr*, {*u*, u_{min}, u_{max}, *du*}] allows the value of *u* to vary between u_{min} and u_{max} in steps *du*.

Manipulate[*expr*, {{*u*, u_{init}}, u_{min}, u_{max}, …}] takes the initial value of *u* to be u_{init}.

Manipulate[*expr*, {{*u*, u_{init}, u_{lbl}}, …}] labels the controls for *u* with u_{lbl}.

Manipulate[*expr*, {*u*, {u_1, u_2, …}}] allows *u* to take on discrete values u_1, u_2, ….

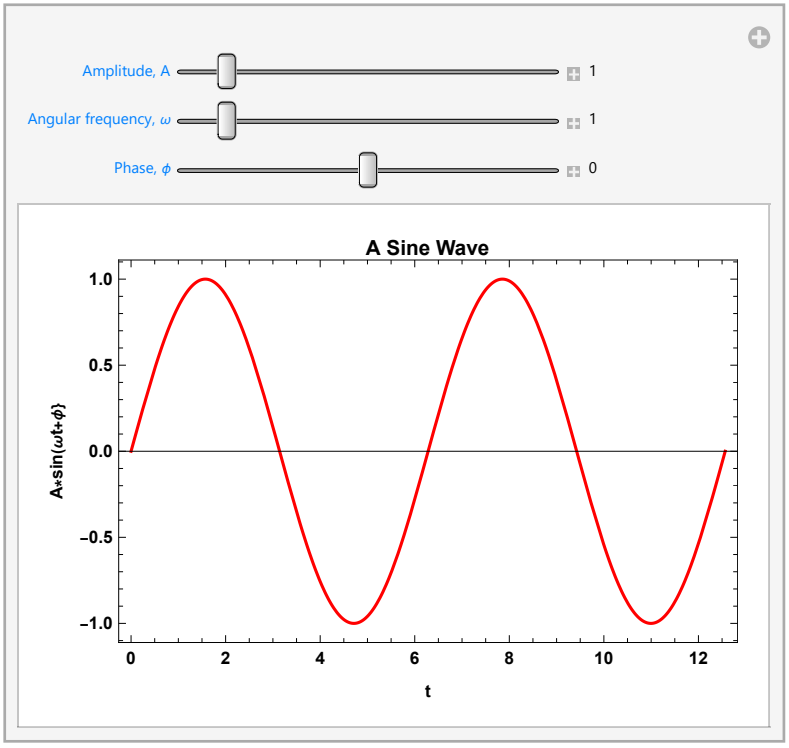
Manipulate[*expr*, {*u*, …}, {*v*, …}, …] provides controls to manipulate each of the *u*, *v*, ….

Manipulate[*expr*, $c_u \to$ {*u*, …}, $c_v \to$ {*v*, …}, …]

links the controls to the specified controllers on an external device.

In[ ]:=
```
g[x_, A_, w_, phi_] := A * Sin[w * x + phi];
Manipulate[
 plt = Plot[g[x, A, w, phi], {x, 0, 4 Pi}, Frame → True, FrameLabel → {"t", "A*sin(ωt+ϕ)"},
   LabelStyle → Directive[Black, Bold], PlotStyle → Red, PlotLabel → "A Sine Wave"],
 {{A, 1, "Amplitude, A"}, 0.1, 10, Appearance → "Labeled"},
 {{w, 1, "Angular frequency, ω"}, 0.1, 10, Appearance → "Labeled"},
 {{phi, 0, "Phase, ϕ"}, -2 Pi, 2 Pi, Appearance → "Labeled"},
 ControlPlacement → Top, SaveDefinitions → True]
 (**Introduction to Manipulate: Demo for a sine wave**)
```

Out[ ]=



0.7.3

Sound Effects

In[ ]:=
```
? Sound
```

Out[ ]=

Symbol

Sound[*primitives*] represents a sound.

Sound[*primitives*, *t*] specifies that the sound should have duration *t*.

Sound[*primitives*, {t_{min}, t_{max}}] specifies that the sound should extend from time t_{min} to time t_{max}.

In[ ]:=
```
? SoundNote
```

Out[ ]=

Symbol

SoundNote[*pitch*] represents a music–like sound note with the specified pitch.

SoundNote[*pitch*, *t*] takes the note to have duration *t*.

SoundNote[*pitch*, {t_{min}, t_{max}}] takes the note to occupy the time interval t_{min} to t_{max}.

SoundNote[*pitch*, *tspec*, "*style*"] takes the note to be in the specified style.

SoundNote[*pitch*, *tspec*, "*style*", *opts*] uses the specified rendering options for the note.

In[ ]:=
```
OdeToJoy = {{"B", "B", "C5", "D5", "D5", "C5", "B", "A", "G", "G", "A", "B", "B", "A", "A", "B",
    "B", "C5", "D5", "D5", "C5", "B", "A", "G", "G", "A", "B", "A", "G", "G", "A", "A",
    "B", "G", "A", "B", "C5", "B", "G", "A", "B", "C5", "B", "A", "G", "A", "D", "B",
    "B", "B", "C5", "D5", "D5", "C5", "B", "A", "G", "G", "A", "B", "A", "G", "G"},
   {0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.75, 0.25, 1, 0.5,
    0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.75, 0.25, 1, 0.5, 0.5,
    0.5, 0.5, 0.5, 0.25, 0.25, 0.5, 0.5, 0.5, 0.25, 0.25, 0.5, 0.5, 0.5, 0.5, 0.5,
    0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.75, 0.25, 1}};
```

Piano sound:

In[ ]:=
```
Sound[SoundNote[##, "Piano"] & @@@ Transpose[OdeToJoy]] // EmitSound
```

Violin sound:

In[ ]:=
```
Sound[SoundNote[##, "Violin"] & @@@ Transpose[OdeToJoy]] // EmitSound
```

Week 1: First-Order ODEs

How to Solve First-Order ODEs Step-by-step?

Table of Contents

Commands list

- Integrate[*f, x*]
- ClearAll [*symb1, symb2, ...*]
- Simplify[*expr*]
- FullSimplify[*expr*]
- Solve[*expr, vars*]

Separable Equations

Many practically useful ODEs can be reduced to the form:

$$g(y)\,y' = f(x)$$

Then, by integrating both sides with respect to x, we obtain:

$$\int g(y)\,y'\,dx = \int f(x)\,dx + C$$

According to Calculus, $y'\,dx = dy$. So, the variable of the integration for left-side becomes y.

$$\int g(y)\,dy = \int f(x)\,dx + C$$

When f and g are continuous functions, the integrals mentioned above exist, and by evaluating them, we obtain a general solution to the given ODE.

Example 1.1: Separable ODE

$$y' = (x+1)\,e^{-x}\,y^2$$

- **Step 1.** The given ODE is separable: $y^{-2}\,dy = (x+1)\,e^{-x}\,dx$

```
In[•]:= ClearAll["Global`*"]
```

```
In[•]:= expr = y'[x] - (x + 1) * Exp[-x] * y[x]^2
Out[•]=
        -ℯ^-x (1 + x) y[x]^2 + y'[x]
```

- **Step 2.** Integrate the left-side with respect to y.

```
In[•]:= Integrate[y^-2, y]
Out[•]=
        -1/y
```

```
In[•]:= ? Integrate
Out[•]=
```

Symbol

Integrate[f, x] gives the indefinite integral $\int f\,dx$.

Integrate[f, {x, x_{min}, x_{max}}] gives the definite integral $\int_{x_{min}}^{x_{max}} f\,dx$.

Integrate[f, {x, x_{min}, x_{max}}, {y, y_{min}, y_{max}}, …] gives the multiple integral $\int_{x_{min}}^{x_{max}} dx \int_{y_{min}}^{y_{max}} dy \ldots f$.

Integrate[f, {x, y, …} ∈ reg] integrates over the geometric region reg.

- **Step 3.** Integrate the right-side with respect to x.

```
In[•]:= Integrate[(x + 1) * Exp[-x], x]
Out[•]=
        ℯ^-x (-2 - x)
```

```
In[•]:= FullSimplify[Integrate[(x + 1) * Exp[-x], x]]
Out[•]=
        -ℯ^-x (2 + x)
```

◆ **Step 4.** By integration, $-\frac{1}{y} = -e^{-x}(2+x) + C$.

```
In[ ]:= FullSimplify[Solve[-1/y == -Exp[-x] * (2 + x) + C, y]]
```

Out[ ]=

$$\left\{\left\{y \to \frac{e^x}{2 - C\,e^x + x}\right\}\right\}$$

◆ **Step 5.** Verify the answer:

```
In[ ]:= ySoln = Exp[x] / (2 - C * Exp[x] + x)
```

Out[ ]=

$$\frac{e^x}{2 - C\,e^x + x}$$

```
In[ ]:= yDSoln = FullSimplify[D[ySoln, x]]
```

Out[ ]=

$$\frac{e^x\,(1+x)}{\left(2 - C\,e^x + x\right)^2}$$

◆ Substituting y and y' to the initially given ODE, we get:

```
In[ ]:= FullSimplify[expr /. {y[x] → ySoln, y'[x] → yDSoln}]
```

Out[ ]=

```
0
```

◆ Check the answer using **DSolve** command:

```
In[ ]:= FullSimplify[DSolve[y'[x] == (x + 1) * Exp[-x] * y[x]^2, y[x], x]]
```

Out[ ]=

$$\left\{\left\{y[x] \to \frac{e^x}{2 + x - e^x\,c_1}\right\}\right\}$$

Example 1.2: Initial Value Problem (IVP)

$$y' = -2xy \quad y(0) = 1.8$$

◆ **Step 1.** The given ODE is separable: $\frac{1}{y}\,dy = -2x\,dx$

```
In[ ]:= ClearAll["Global`*"]
```

```
In[ ]:= expr = y'[x] + 2 x * y[x]
```

Out[ ]=

```
2 x y[x] + y'[x]
```

◆ **Step 2.** Integrate the left-side with respect to y.

In[ ]:=
```
Integrate[1/y, y]
```

Out[ ]=
```
Log[y]
```

- **Step 3.** Integrate the right-side with respect to x.

In[ ]:=
```
Integrate[-2 x, x]
```

Out[ ]=
```
-x^2
```

- **Step 4.** By integration, we got $\ln y = -x^2 + C$. Solving the expression, we get a general solution to ODE:

In[ ]:=
```
? Solve
```

Out[ ]=

Symbol

Solve[*expr*, *vars*] attempts to solve the system *expr* of equations or inequalities for the variables *vars*.

Solve[*expr*, *vars*, *dom*] solves over the domain *dom*. Common choices of *dom* are Reals, Integers, and Complexes.

- We solve the expression over the **domain of Real numbers**, because **the natural logarithm of y** exists only when $y > 0$**:**

In[ ]:=
```
FullSimplify[Solve[Log[y] == -x^2 + c, y, Reals]]
```

Out[ ]=
$$\{\{y \to e^{-x^2+c}\}\}$$

- **Step 5.** Now let's use initial value to get a **particular solution:**

In[ ]:=
```
ySoln = c * Exp[-x^2]
```

Out[ ]=
$$e^{-x^2}\, c$$

In[ ]:=
```
y0 = ySoln /. x → 0
```

Out[ ]=
$$c$$

In[ ]:=
```
Solve[y0 == 1.8, c]
```

Out[ ]=
```
{{c → 1.8}}
```

In[ ]:=
```
yIVP = ySoln /. c → 1.8
```

Out[ ]=
$$1.8\, e^{-x^2}$$

◆ **Step 6.** Verify the answer:

```
In[ ]:= yDIVP = FullSimplify[D[yIVP, x]]
Out[ ]=
        -3.6 ℮^-x^2 x
```

◆ Substituting y and y' to the initially given ODE, we get:

```
In[ ]:= FullSimplify[expr /. {y[x] → yIVP, y'[x] → yDIVP}]
Out[ ]=
        0.
```

◆ Check the answer using **DSolve** command:

```
In[ ]:= FullSimplify[DSolve[{y'[x] == -2 x * y[x], y[0] == 1.8}, y[x], x]]
Out[ ]=
        {{y[x] → 1.8 ℮^-x^2}}
```

Exact ODEs & Integrating Factors

A 1st order ODE $M(x, y) + N(x, y)\, y' = 0$ written as

$$M(x, y)\, dx + N(x, y)\, dy = 0$$

is **an exact differential equation.** It can be written as the differential of some function $u(x, y)$.

$$\frac{du}{dx}\, dx + \frac{du}{dy}\, dy = du$$

Comparing the ODE and the differential form, we see that:

$$du = 0 \quad \Longrightarrow \quad \frac{\partial u}{\partial x} = M \qquad \frac{\partial u}{\partial y} = N$$

Let's do some partial derivatives manupulation to get:

$$\frac{\partial M}{\partial y} = \frac{\partial^2 u}{\partial y\, \partial x} \qquad \frac{\partial N}{\partial x} = \frac{\partial^2 u}{\partial x\, \partial y}$$

Consequently, **the condition for the exactness of ODE** is when the following partial derivatives are equal:

$$\frac{\partial M}{\partial y} = \frac{\partial N}{\partial x}$$

Finally, by integration we obtain **an implicit solution** to an ODE as a function of $u(x, y)$:

$$u(x, y) = c$$

The function $u(x, y)$ can be found by the following systematic way; EITHER by integrating with respect to x, where $k(y)$ is the constant of integration.

$$u = \int M\, dx + k(y)$$

OR by integrating with respect to y, where $l(x)$ is the constant of integration as well:

$$u = \int N\, dy + l(x)$$

Example 1.3: An Exact ODE

$$\cos(x + y)\, dx + \left(3\, y^2 + 2\, y + \cos(x + y)\right) dy = 0$$

- **Step 1. Test for exactness.** By looking at the equation, we see that $M = \cos(x + y)$ and $N = 3\, y^2 + 2\, y + \cos(x + y)$. But instead of M & N, we use P & Q, because the capital letter **N** is **protected by Mathematica.**

In[ ]:=
```
ClearAll["Global`*"]
```

In[ ]:=
```
P[x_, y_] := Cos[x + y];
Q[x_, y_] := 3 y^2 + 2 y + Cos[x + y];
```

- **NOTE:** The variable cannot be named "**N**" because the Wolfram language has a built-in symbol described below.

In[ ]:=
```
? N
```

Out[ ]=

Symbol

N[*expr*] gives the numerical value of *expr*.

N[*expr*, *n*] attempts to give a result with *n*-digit precision.

- Let's check if the given ODE is exact:

In[ ]:=
```
D[P[x, y], y] == D[Q[x, y], x]    (**Check for exactness **)
```

Out[ ]=
```
True
```

- The given **ODE is exact.**
- **Step 2.** Find the general solution $u(x, y)$ by integrating with respect to x:

$$u = \int P\, dx + k(y)$$

```
In[•]:= u = Integrate[P[x, y], x] + k[y]
Out[•]=
```

$$k[y] + \text{Cos}[y]\,\text{Sin}[x] + \text{Cos}[x]\,\text{Sin}[y]$$

- where k[y] is a yet-to-be-determined function.
- **Step 3.** Let's solve for $k(y)$:

```
In[•]:= ODEofK = Simplify[D[u, y] == Q[x, y]]
Out[•]=
```

$$y\,(2 + 3\,y) == k'[y]$$

```
In[•]:= KSoln = DSolve[ODEofK, k[y], y]
Out[•]=
```

$$\{\{k[y] \to y^2 + y^3 + c_1\}\}$$

```
In[•]:= u
Out[•]=
```

$$k[y] + \text{Cos}[y]\,\text{Sin}[x] + \text{Cos}[x]\,\text{Sin}[y]$$

- **Step 4.** Substitute the value of k[y] to the given ODE:

```
In[•]:= u /. KSoln[[1]]
Out[•]=
```

$$y^2 + y^3 + c_1 + \text{Cos}[y]\,\text{Sin}[x] + \text{Cos}[x]\,\text{Sin}[y]$$

```
In[•]:= FullSimplify[u /. KSoln[[1]]]
Out[•]=
```

$$y^2 + y^3 + c_1 + \text{Sin}[x + y]$$

- So, the **general solution to the ODE is:** $u(x, y) = \sin(x + y) + y^2 + y^3 + c_1 = \text{constant}$

$$u(x, y) = \sin(x + y) + y^2 + y^3 = c$$

- **Step 5.** Check the obtained solution:

```
In[•]:= uSoln = y^2 + y^3 + c1 + Sin[x + y];
        D[uSoln, x]
Out[•]=
```

$$\text{Cos}[x + y]$$

```
In[•]:= D[uSoln, y]
Out[•]=
```

$$2\,y + 3\,y^2 + \text{Cos}[x + y]$$

- So, the solution is **correct.**

Non-Exact ODEs and Integrating Factors

What to do if the equation is not exact?

In a case of Nonexactness, the ODE can be solved by reducing the equation to the exact form. It is done by the integrating factors. The nonexact ODE is given by the following form:

$$P(x, y)\, dx + Q(x, y)\, dy = 0$$

If the equation is multiplied by a function F both sides, the result is

$$FP\, dx + FQ\, dy = 0$$

This function $F(x, y)$ is called **an integrating factor.**

How to find integrating factor?

As discussed before, **the condition for the exactness of ODE** is when the following partial derivatives are equal:

$$\frac{\partial M}{\partial y} = \frac{\partial N}{\partial x}$$

Thus, the condition for the exactness when the integrating factor is present is:

$$\frac{\partial}{\partial y}(FP) = \frac{\partial}{\partial x}(FQ)$$

By the product rule, with subscripts denoting partial derivatives, this gives

$$F_y P + FP_y = F_x Q + FQ_x$$

Because the integration factor depends only on **one variable** (either x or y), it simplifies easily.

Let's assume that the integrating factor depends on the x only. Also, let's denote the derivative of F_x as $F' = \frac{\partial F}{\partial x}$. Then it leads to

$$FP_y = F' Q + FQ_x$$

Simplifying it, we get the formula for the Integrating Factor $F(x)$:

$$F(x) = \exp \int R(x)\, dx \qquad \text{where} \qquad R(x) = \frac{1}{Q}\left(\frac{\partial P}{\partial y} - \frac{\partial Q}{\partial x}\right)$$

After similar mathematical manipilations, the formula for the Integrating Factor $F(y)$ is found.

$$F^*(y) = exp \int R^*(y)\, dy \qquad \text{where} \qquad R^*(y) = \frac{1}{P}\left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y}\right)$$

Example 1.4: A Non-exact ODE with IVP

$$(e^{x+y} + ye^y)\, dx + (xe^y - 1)\, dy = 0,\ y(0) = -1$$

- **Step 1. Test for exactness.** By looking at the equation, we see that $P = e^{x+y} + y\, e^y$ and $Q = xe^y - 1$.

```
ClearAll["Global`*"]
```

In[•]:=
```
P[x_, y_] := Exp[x + y] + y * Exp[y];
Q[x_, y_] := x * Exp[y] - 1;
```

In[•]:=
```
FullSimplify[D[P[x, y], y] == D[Q[x, y], x] ]
```

Out[•]=
$$e^y \left(e^x + y\right) == 0$$

- The result shows that the given **ODE is NOT exact.**
- **Step 2. Finding the Integrating Factor.** First, assume that the Integrating Factor depends only on *x*.

In[•]:=
```
Rx = FullSimplify[1/Q[x, y] * (D[P[x, y], y] - D[Q[x, y], x])]
```

Out[•]=
$$\frac{e^y \left(e^x + y\right)}{-1 + e^y x}$$

- We see that the ***R* contains both on *x*** and ***y*.** Therefore, the **first assumption is wrong.**
- Now, let's assume that ***F* depends on *y*.**

In[•]:=
```
Ry = FullSimplify[1/P[x, y] * (D[Q[x, y], x] - D[P[x, y], y])]
```

Out[•]=
$$-1$$

- The **second is assumption is correct** and the Integrating Factor **depends only on *y*, *F(y)*.**

In[•]:=
```
Fy = FullSimplify[Exp[Integrate[Ry, y]]]
```

Out[•]=
$$e^{-y}$$

- Let's let's redefine *P[x,y]* and *Q[x,y]* after multiplying the integrating factor of e^{-y} to both sides of the given ODE:

```
ClearAll[P, Q, x, y];
```

```
In[•]:= P[x_, y_] := (Exp[x + y] + y * Exp[y]) * Exp[-y];
        Q[x_, y_] := (x * Exp[y] - 1) * Exp[-y];
```

- Check the obtained equation for exactness: $(e^x + y)\,dx + (-e^{-y} + x)\,dy = 0$

```
In[•]:= FullSimplify[D[P[x, y], y] == D[Q[x, y], x]]
Out[•]=
        True
```

- Indeed, **it is exact.**
- **Step 3. General Solution.** As shown before, the general solution to the ODE can be found by the following formula, where $k(y)$ is the constant of integration.

$$u = \int P * F(y)\,dx + k(y)$$

```
In[•]:= u = Integrate[P[x, y], x] + k[y]
Out[•]=
        ℯ^x + x y + k[y]
```

- where k[y] is a yet-to-be-determined function.
- Let's solve for $k(y)$:

```
In[•]:= ODEofK = Simplify[D[u, y] == Q[x, y]]
Out[•]=
        ℯ^-y + k'[y] == 0

In[•]:= KSoln = DSolve[ODEofK, k[y], y]
Out[•]=
        {{k[y] → ℯ^-y + C[1]}}

In[•]:= u
Out[•]=
        ℯ^x + x y + k[y]
```

- Thus we have:

```
In[•]:= u /. KSoln[[1]]
Out[•]=
        ℯ^x + ℯ^-y + x y + C[1]

In[•]:= uSoln = FullSimplify[u /. KSoln[[1]]]
Out[•]=
        ℯ^x + ℯ^-y + x y + C[1]
```

- Hence, the **general solution is**

$$u(x, y) = e^x + e^{-y} + xy = c$$

- **Step 4.** Find the Particular solution with $y(0) = -1$:

```
In[ ]:= u0 = uSoln /. x → 0
Out[ ]=
1 + ℯ^-y + 𝕔1

In[ ]:= Solve[u0 == -1, 𝕔1]
Out[ ]=
{{𝕔1 → -ℯ^-y (1 + 2 ℯ^y)}}

In[ ]:= uIVP = Simplify[uSoln /. 𝕔1 → -ℯ^-y (1 + 2 ℯ^y)]
Out[ ]=
-2 + ℯ^x + x y
```

- **Step 5.** Check the obtained solution:

```
In[ ]:= D[uSoln, x]
Out[ ]=
ℯ^x + y

In[ ]:= D[uSoln, y]
Out[ ]=
-ℯ^-y + x
```

- It can be seen that $D[u, x]\,dx + D[u, y]\,dy = 0$ recovers the given ODE. Since $u = const.$, we have $du = D[u, x]\,dx + D[u, y]\,dy = 0$.

First-Order Linear Equations

A first-order ODE (in interval a < x < b) written in the standard form as follows

$$y' + p(x)\,y = r(x)$$

is called **a Linear ODE.** If the $r(x)$ equals to 0, the equation becomes **a Homogeneous Linear ODE.**

$$y' + p(x)\,y = 0$$

It is easily noticed that the ODE is separable, so by separating variable, the solution to the equation is

$$y(x) = \mathbb{c}\,e^{-\int p(x)\,dx} \qquad \mathbb{c} = \pm e^{c^*} \qquad \text{when } y \gtrless 0$$

And **the trivial solution y(x) = 0** for all x in the mentioned interval.

In a case the equation is **a Nonhomogeneous Linear ODE,** another method is used.

$$y' + p(x)\,y = r(x)$$

Here the ODE has a pleasant property that the integrating factor depends only on the x.

$$F\,y' + p\,F\,y = r\,F$$

After some mathematical manipulations (refer to the textbook), the general solution to the nonhomogeneous linear ODE is obtained.

$$y(x) = e^{-h}\left(\int e^{h}\,r\,dx + c\right) \qquad \text{where} \qquad h = \int p(x)\,dx$$

$$y(x) = e^{-h}\int e^{h}\,r\,dx + c e^{-h}$$

Example 1.5: First-Order ODE, IVP

$$y' + y\tan x = \sin 2x,\ y(0) = 1$$

- **Step 1.** From the standard form, here $p = \tan x$, $r = \sin 2x$.

```
ClearAll["Global`*"]
p = Tan[x];
r = Sin[2 x];
```

- We can introduce p & r as functions as was done in the previous example, but we don't have to.
- **Step 2.** Find h using the formula above.

In[ ]:=
```
h = Integrate[p, x]
```
Out[ ]=
```
-Log[Cos[x]]
```

- **Step 3.** Find the general solution to the given ODE. *ysoln0* is the **first term** and *ysoln1* is the **second term** in the general solution.

In[ ]:=
```
ysoln0 = Exp[-h] * Integrate[Exp[h] * r, x]
```
Out[ ]=
```
-2 Cos[x]^2
```

In[ ]:=
```
ysoln1 = Exp[-h] * c1
```
Out[ ]=
```
c1 Cos[x]
```

In[ ]:=
```
ysolnGen = ysoln0 + ysoln1
```
Out[ ]=
```
c1 Cos[x] - 2 Cos[x]^2
```

- **Step 4.** Find the particular solution by the initial data: $y(0) = 1$.

```
In[ ]:= ysolnGen /. x → 0
Out[ ]=
-2 + c1
```

- Solve for ***c1:***

```
In[ ]:= Solve[((c1 Cos[x] - 2 Cos[x]^2) /. x → 0) == 1, c1]
Out[ ]=
{{c1 → 3}}
```

```
In[ ]:= ysoln = (ysoln0 + ysoln1) /. c1 → 3
Out[ ]=
3 Cos[x] - 2 Cos[x]^2
```

- **Step 5.** Verify the solution.

```
In[ ]:= FullSimplify[D[ysoln, x] + p * ysoln == r]
Out[ ]=
True
```

Bernoulli Equation

Many ODEs with a huge importance in engineering are nonlinear that can transform into a linear ODE. One of the useful one is **the Bernoulli Equation.**

$$y' + p(x)\, y = g(x)\, y^{a} \qquad (a \text{ is any real number})$$

When $a = 0$, the Bernoulli equation is **a linear 1st order ODE**, which we have solved in the previous section.

When $a = 1$, the Bernoulli equation is **a separable, linear, 1st order, homogeneous ODE**, which is even simpler to solve.

When a is neither 0 nor 1, we have **a nonlinear ODE** of y(x).

The trick to solve the Bernoulli equation is to introduce the following variable transformation:

$$u(x) = [y(x)]^{1-a}$$

Using the $u(x)$ transformation variable, we get the linear ODE, which we know how to solve.

$$u' - (1 - a)\, p\, u = (1 - a)\, g$$

Example 1.6: Logistic Equation

$$y' = Ay - By^2$$

The given ODE is a Bernoulli equation is known as the **Logistic Equation** Equation).

```
In[•]:= ClearAll["Global`*"]
```

- **Step 1.** Find the $u(x)$ transformation variable. From the equation, we see that a is equal to 2.

```
In[•]:= u[y] = y[x]^(1-a) /. a → 2
```

```
Out[•]=
        1
      y[x]
```

```
In[•]:= D[u[y], x]
```

```
Out[•]=
      -y'[x]
      ------
       2
      y[x]
```

```
In[•]:= FullSimplify[D[u[y], x] /. y'[x] → {A * y[x] - B * y[x]^2}]
```

```
Out[•]=
        A
     {B - ----}
        y[x]
```

- **Step 2.** We found earlier that $u(x) = \frac{1}{y(x)}$. Hence, using it, $u'(x)$ becomes $u'(x) = B - A\,u(x)$.
- Now we have a linear ODE of form: $u' + A\,u = B$.
- **Step 3.** Solve the obtained linear ODE. It is **nonhomogeneous**. Thus, we use the same method as in **Example 1.3**.

```
In[•]:= p = A;
        r = B;
```

```
In[•]:= h = Integrate[p, x]
```

```
Out[•]=
      A x
```

```
In[•]:= usoln0 = Exp[-h] * Integrate[Exp[h] * r, x]
```

```
Out[•]=
      B
      -
      A
```

```
In[•]:= usoln1 = Exp[-h] * c1
```

```
Out[•]=
          -A x
      c1 ℯ
```

In[ ]:=
```
usolnGen = ysoln0 + ysoln1
```

Out[ ]=

ysoln0 + ysoln1

- **Step 4.** Since $u(x) = \frac{1}{y(x)}$, the **general solution** $y(x)$ as follows:

In[ ]:=
```
FullSimplify[Solve[((u[x]) /. u[x] → usolnGen) == 1/y[x], y[x]]]
```

Out[ ]=

$$\left\{\left\{y[x] \to \frac{1}{ysoln0 + ysoln1}\right\}\right\}$$

- **Step 5.** Also, directly from the ODE, it is seen that $y(x) = 0$ **for all** x is the solution to the equation as well **(a trivial solution).**
- **Step 6.** Always verify the solution.

In[ ]:=
```
ysoln = A/(B + A c1 e^(-A x))
```

Out[ ]=

$$\frac{A}{B + A\, c1\, e^{-A x}}$$

In[ ]:=
```
FullSimplify[D[ysoln, x]] == FullSimplify[A * ysoln - B * ysoln^2]
```

Out[ ]=
```
True
```

Summary

After completing this chapter, you should be able to

- solve several types of first-order ODEs step-by-step using Wolfram Mathematica.
- develop SOPs to solve first-order ODEs.
- develop the habit of always checking your solutions for quality assurance.

Week 2: Second-Order ODEs (Part 1)

How to solve 2nd-Order ODEs Step-by-step?

Table of Contents

Commands list

- DSolve[*eqn, u, x*]
- expr[[i]] or Part[*expr, i*]
- Log[*z*]
- D[*f, x*]
- Chop[*expr*]

Homogeneous Linear ODEs of Second Order

The standard form of the **Linear Second Order** ODE is as follows:

$$y'' + p(x)\,y' + q(x)\,y = r(x)$$

If $r(x)$ term is equal to **0**:

$$y'' + p(x)\,y' + q(x)\,y = 0$$

the ODE is called **Homogeneous.** If $r(x) \neq 0$, then it is called **Nonhomogeneous**.

The linear homogeneous second order ODEs have a rich solution structure that relies on the **Superposition Principle.**

The ***superposition principle*** or ***linearity principle*** means that we can obtain further solutions from the given ones by adding them or multiplying them with any constants.

$$y = c_1\,y_1 + c_2\,y_2 \qquad (c_1,\, c_2 \text{ arbitrary contants})$$

Note: This principle works only for **Homogeneous** AND **Linear** ODEs.

For a second-order homogeneous linear ODE, the **Initial Value Problem** consists of **t initial conditions:**

$$y(x_0) = K_0 \qquad y'(x_0) = K_1$$

The **General Solution** to the ODE is

$$y = c_1\,y_1 + c_2\,y_2$$

Here y_1 and y_2 are **not proportional** and c_1 and c_2 are arbitrary constants. This pair of linearly independent solutions is called **a basis of solutions.**

1.1

Example 2.1: Solve Second-Order ODE using DSolve

$$(x^2 - x)\,y'' - x\,y' + y = 0$$

- **Step 1.** Use the **DSolve** function directly, including the equation for the function **y[x]**, with independent variable x

In[ ]:=
```
ClearAll["Global`*"]
sol = DSolve[(x^2 - x) * y''[x] - x * y'[x] + y[x] == 0, y[x], x]
```

Out[ ]=
```
{{y[x] → x c1 + c2 (-1 - x Log[x])}}
```

- The solution to **y[x]** is written to **"ysol"** variable. Here the double square brackets is the short form [[ ]] for the **Part** function, which is used to get parts of lists.
- In short, the program gets the 1st part of the expression "sol", and writes it to the new variable "ysol".

In[ ]:=
```
ysol = y[x] /. sol[[1]]; ysol
```

Out[ ]=

$$x\, c_1 + c_2\, (-1 - x \operatorname{Log}[x])$$

In[ ]:=
```
? Part
```

Out[ ]=

Symbol

expr[[*i*]] or Part[*expr*, *i*] gives the i^{th} part of *expr*.

expr[[−*i*]] counts from the end.

expr[[*i*, *j*, …]] or Part[*expr*, *i*, *j*, …] is equivalent to *expr*[[*i*]][[*j*]] ….

expr[[{i_1, i_2, …}]] gives a list of the parts i_1, i_2, … of *expr*.

expr[[*m* ;; *n*]] gives parts *m* through *n*.

expr[[*m* ;; *n* ;; *s*]] gives parts *m* through *n* in steps of *s*.

expr[["*key*"]] gives the value associated with the key "*key*" in an association *expr*.

expr[[Key[*k*]]] gives the value associated with an arbitrary key *k* in the association *expr*.

- The new function called **GeneralSol[x_]** takes the solution to **y[x]** from the variable **"ysol"**. It is done so in the next step, we can plot the graph of the obtained solution.

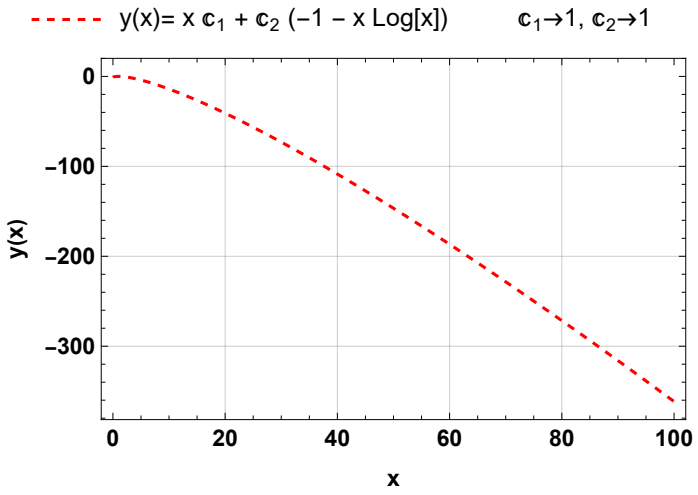
In[ ]:=
```
GeneralSol[x_] := ysol; GeneralSol[x]
```

Out[ ]=

$$x\, c_1 + c_2\, (-1 - x \operatorname{Log}[x])$$

In[ ]:=
```
Plot[GeneralSol[x] /. {C[1] → 1, C[2] → 1}, {x, 0, 100},
 Frame → True, FrameLabel → {"x", "y(x)"}, GridLines → Automatic,
 BaseStyle → {FontWeight → "Bold", Black, FontSize → 12}, PlotStyle → {Dashed, Red},
 PlotLegends → Placed[{"y(x)= x c₁ + c₂ (-1 - x Log[x])          c₁→1, c₂→1  "}, Above]]
```

Out[ ]=



- From the solution, it is seen that the solution perfectly matches the form $y = c_1 y_1 + c_2 y_2$, thus **a basis of solutions** is the following: $y_1 = x$ and $y_2 = -1 - x \ln(x)$.
- **Note:** In Wolfram Mathematica, the function **Log[x]** gives the natural logarithm of x.

```
In[ ]:= ? Log
```

```
Out[ ]=
```

Symbol

Log[*z*] gives the natural logarithm of *z* (logarithm to base e).

Log[*b*, *z*] gives the logarithm to base *b*.

- **Step 2.** Check the obtained solution by comparing Left-Hand-Side (**LHS**) and Right-Hand-Side (**RHS**).

```
In[ ]:= LHS = FullSimplify[
          (x^2 - x) * D[GeneralSol[x], {x, 2}] - x * D[GeneralSol[x], {x, 1}] + GeneralSol[x]]
```

```
Out[ ]=
0
```

```
In[ ]:= RHS = 0
```

```
Out[ ]=
0
```

```
In[ ]:= LHS == RHS
```

```
Out[ ]=
True
```

2

Homogeneous Linear ODEs with Constant Coefficients

Now let's consider the homogeneous linear second-order ODEs with constant coefficients a and b:

$$y'' + a\,y' + b\,y = 0$$

These ODEs have a huge implications in the mechanical and electrical vibrations, as we will see further.

To solve the homogeneous linear second-order ODEs, we need to solve the **characteristic equation** (or auxiliary equation)

$$\lambda^2 + a\,\lambda + b = 0$$

Because the characteristic equation is in the quadratic form, it may have three different kind of

roots, depending on the sign of the discrimanant $a^2 - 4\,b$. These 3 cases are as follows:

> **(Case I)** *Two real roots if* $a^2 - 4\,b > 0$,
>
> **(Case II)** *A real double root if* $a^2 - 4\,b = 0$,
>
> **(Case III)** *Complex conjugate roots if* $a^2 - 4\,b < 0$.

Depending on the case, the basis of solutions and the general solution to the ODE is summarized in the following table:

Case	Roots of (2)	Basis of (1)	General Solution of (1)
I	Distinct real $\lambda_1,\ \lambda_1$	$e^{\lambda_1 x},\ e^{\lambda_2 x}$	$y = c_1 e^{\lambda_1 x} + c_2 e^{\lambda_2 x}$
II	Real double root $\lambda = -\frac{1}{2} a$	$e^{-ax/2},\ x\,e^{-ax/2}$	$y = (c_1 + c_2 x)\, e^{-ax/2}$
III	Complex conjugate $\lambda_1 = -\frac{1}{2} a + i\omega,$ $\lambda_2 = -\frac{1}{2} a - i\omega$	$e^{-ax/2} \cos \omega x$ $e^{-ax/2} \sin \omega x$	$y = e^{-ax/2}\,(A \cos \omega x + B \sin \omega x)$

2.1

Example 2.2: Case I with IVP

> $y'' + y' - 2y = 0, \qquad y(0) = 4,\ y'(0) = -5$

- **Step 1.** Solve the characteristic equation and determine what case the ODE refers to.

In[ ]:=
```
ClearAll["Global`*"]
roots = Solve[λ^2 + λ - 2 == 0, λ] (** Note that we have to use == , not = **)
```

Out[ ]=
```
{{λ → -2}, {λ → 1}}
```

- **Step 2.** Find the general solution. We got two distinct real roots, so we proceed with **Case I.**

In[ ]:=
```
λ1 = λ /. roots[[1]]; λ2 = λ /. roots[[2]]; {λ1 , λ2 }
(** Double squared brackets [[]] get the ith element from the list**)
```

Out[ ]=
```
{-2, 1}
```

```
In[ ]:= GeneralSol[x_] := c1 * Exp[λ1 * x] + c2 * Exp[λ2 * x]; GeneralSol[x]
Out[ ]=
c1 ℯ^(-2 x) + c2 ℯ^x
```

◆ **Step 3.** Find the particular solution using the initial conditions: $y(0) = 4, y'(0) = -5$

```
In[ ]:= cond1 = GeneralSol[0] == 4;
        cond2 = (D[GeneralSol[x], x] /. x → 0) == -5;
```

◆ Solve for the arbitrary constants.

```
In[ ]:= soln = Solve[{cond1, cond2}, {c1, c2}]
Out[ ]=
{{c1 → 3, c2 → 1}}
```

◆ Obtain the particular solution.

```
In[ ]:= GeneralSol[x] /. soln
Out[ ]=
{3 ℯ^(-2 x) + ℯ^x}
```

◆ **Step 4.** Verify the solution.

```
In[ ]:= ivpSoln[x_] := 3 ℯ^(-2 x) + ℯ^x;
```

◆ Check for the initial conditions.

```
In[ ]:= ivpSoln[0]
Out[ ]=
4
```

```
In[ ]:= D[ivpSoln[x], {x, 1}] /. x → 0
Out[ ]=
-5
```

◆ Check that the solution satisfies the given ODE $y'' + y' - 2y = 0$.

```
In[ ]:= LHS = D[ivpSoln[x], {x, 2}] + D[ivpSoln[x], {x, 1}] - 2 * ivpSoln[x]
Out[ ]=
6 ℯ^(-2 x) + 2 ℯ^x - 2 (3 ℯ^(-2 x) + ℯ^x)
```

```
In[ ]:= RHS = 0
Out[ ]=
0
```

```
In[ ]:= FullSimplify[LHS == RHS]
Out[ ]=
True
```

◆ So the solution satisfies both the initial conditions and the ODE check.

◆ **Step 5.** Verify the solution by **DSolve** (Not Required).

```
In[•]:= ClearAll[y]; DSolve[y''[x] + y'[x] - 2 y[x] == 0, y[x], x]
```

Out[•]=

$$\{\{y[x] \to e^{-2x} c_1 + e^{x} c_2\}\}$$

```
In[•]:= yp = DSolve[{y''[x] + y'[x] - 2 y[x] == 0, y[0] == 4, y'[0] == -5}, y[x], x]
```

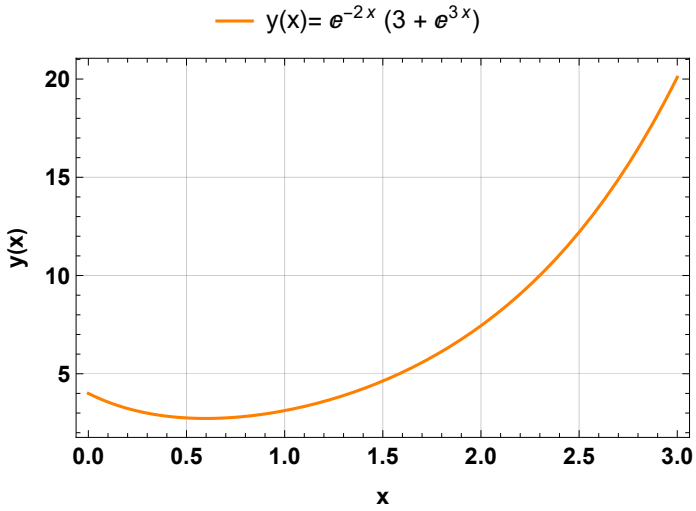
Out[•]=

$$\{\{y[x] \to e^{-2x}\left(3 + e^{3x}\right)\}\}$$

- Using the DSolve function yields in the same result.
- Let's also take a look at the graph of the solution:

```
In[•]:= Plot[y[x] /. yp, {x, 0, 3}, Frame → True, FrameLabel → {"x", "y(x)"},
  GridLines → Automatic, BaseStyle → {FontWeight → "Bold", Black, FontSize → 12},
  PlotStyle → {Orange}, PlotLegends → Placed[{"y(x) = e^-2x (3 + e^3x)"}, Above]]
```

Out[•]=



2.2

Example 2.3: Case II with IVP

$$y'' + y' + 0.25\,y = 0, \qquad y(0) = 3.0,\ y'(0) = -3.5$$

```
In[•]:= ClearAll["Global`*"]
```

- **Step 1.** Solve the characteristic equation and determine what case the ODE refers to.

```
In[•]:= roots = Solve[λ^2 + λ + 0.25 == 0, λ] (** Note that we have to use == , not = **)
```

Out[•]=

$$\{\{\lambda \to -0.5\}, \{\lambda \to -0.5\}\}$$

- **Step 2.** Find the general solution. We got a real double root, so we proceed with **Case II.**

```
In[•]:= λ1 = λ /. roots[[1]]; λ2 = λ /. roots[[2]]; {λ1, λ2}
        (** Double squared brackets [[]] get the ith element from the list**)
Out[•]=
        {-0.5, -0.5}

In[•]:= GeneralSol[x_] := (c1 + c2 * x) * Exp[λ1 * x]; GeneralSol[x]
Out[•]=
        e^(-0.5 x) (c1 + c2 x)
```

◆ **Step 3.** Find the particular solution using the initial conditions: $y(0) = 3.0,\ y'(0) = -3.5$

```
In[•]:= cond1 = GeneralSol[0] == 3.0;
        cond2 = (D[GeneralSol[x], x] /. x → 0) == -3.5;
```

◆ Solve for the arbitrary constants.

```
In[•]:= soln = Solve[{cond1, cond2}, {c1, c2}]
Out[•]=
        {{c1 → 3., c2 → -2.}}
```

◆ Obtain the particular solution.

```
In[•]:= GeneralSol[x] /. soln
Out[•]=
```

$$\left\{e^{-0.5\,x}\,(3. - 2.\,x)\right\}$$

◆ **Step 4.** Verify the solution.

```
In[•]:= ivpSoln[x_] := e^(-0.5` x) (3.` - 2.` x); ivpSoln[x]
Out[•]=
        e^(-0.5 x) (3. - 2. x)
```

◆ Check for the initial conditions.

```
In[•]:= ivpSoln[0]
Out[•]=
        3.

In[•]:= D[ivpSoln[x], {x, 1}] /. x → 0
Out[•]=
        -3.5
```

◆ Check that the solution satisfies the given ODE $y'' + y' + 0.25\,y = 0$.

```
In[•]:= LHS = D[ivpSoln[x], {x, 2}] + D[ivpSoln[x], {x, 1}] + 0.25 * ivpSoln[x]
Out[•]=
        0.

In[•]:= RHS = 0
Out[•]=
        0
```

In[ ]:=
```
LHS == RHS
```

Out[ ]=
```
True
```

- So the solution satisfies both the initial conditions and the ODE check.
- **Step 5.** Verify the solution by DSolve (Not Required).

In[ ]:=
```
ClearAll[y]; DSolve[y''[x] + y'[x] + 0.25 y[x] == 0, y[x], x]
```

Out[ ]=
$$\{\{y[x] \to e^{-0.5\,x}\,C_1 + e^{-0.5\,x}\,x\,C_2\}\}$$

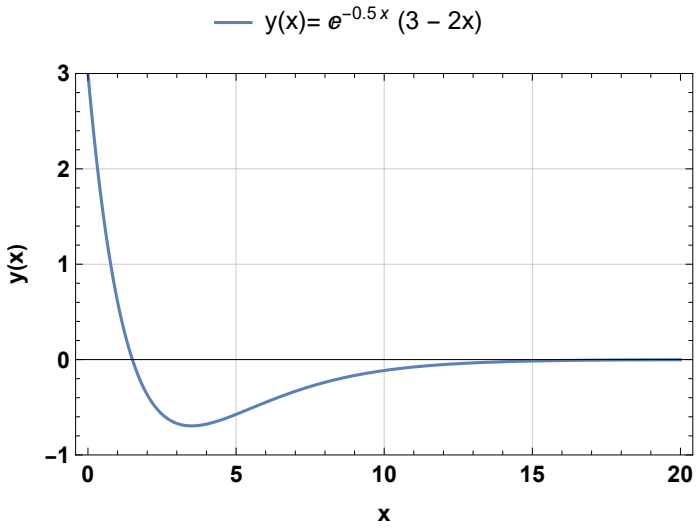
In[ ]:=
```
yp =
 FullSimplify[DSolve[{y''[x] + y'[x] + 0.25 y[x] == 0, y[0] == 3.0, y'[0] == -3.5}, y[x], x]]
```

Out[ ]=
$$\{\{y[x] \to e^{-0.5\,x}\,(3. - 2.\,x)\}\}$$

- Using the DSolve function yields in the same result.
- Let's also take a look at the graph of the solution.

In[ ]:=
```
Plot[y[x] /. yp, {x, 0, 20}, Frame → True, FrameLabel → {"x", "y(x)"}, GridLines → Automatic,
 BaseStyle → {FontWeight → "Bold", Black, FontSize → 12}, PlotRange → {-1, 3},
 PlotStyle → Automatic, PlotLegends → Placed[{"y(x) = e^(-0.5 x) (3 - 2x)"}, Above]]
```

Out[ ]=



2.3

Example 2.4: Case III with IVP

$$y'' + 0.4\,y' + 9.04\,y = 0, \qquad y(0) = 0,\ y'(0) = 3$$

In[ ]:=
```
ClearAll["Global`*"]
```

- **Step 1.** Solve the characteristic equation and determine what case the ODE refers to.

```
In[•]:= a = 0.4; b = 9.04;
        roots = Solve[λ^2 + a * λ + b == 0, λ] (** Note that we have to use == , not = **)
Out[•]=
        {{λ → -0.2 - 3. i}, {λ → -0.2 + 3. i}}
```

- **Step 2.** Find the general solution. We got two complex roots, so we proceed with **Case III**.

In this case, the roots of the characteristic equation are complex numbers that give the complex solutions of the ODE. However, it can be shown that we can obtain a basis of real solutions:

$$y_1 = e^{-ax/2}\cos\omega x \quad \text{and} \quad y_2 = e^{-ax/2}\sin\omega x \quad \text{where} \quad \omega = \sqrt{b - \frac{a^2}{4}}$$

```
In[•]:= ω = Sqrt[b - a^2 / 4]
Out[•]=
        3.
```

```
In[•]:= ClearAll[A, B];
        GeneralSol[x_] := (A * Cos[ω * x] + B * Sin[ω * x] ) * Exp[(-a / 2) * x];
        GeneralSol[x]
Out[•]=
        e^(-0.2 x) (A Cos[3. x] + B Sin[3. x])
```

- **Step 3.** Find the particular solution using the initial conditions: $y(0) = 0,\ y'(0) = 3$

```
In[•]:= cond1 = GeneralSol[0] == 0;
        cond2 = (D[GeneralSol[x], x] /. x → 0) == 3;
```

- Solve for the constants A and B.

```
In[•]:= soln = Solve[{cond1, cond2}, {A, B}]
Out[•]=
        {{A → 0., B → 1.}}
```

- Obtain the particular solution.

```
In[•]:= GeneralSol[x] /. soln
Out[•]=
```

$$\left\{e^{-0.2\,x}\,(0. + 1.\,\text{Sin}[3.\,x])\right\}$$

```
In[•]:= FullSimplify[GeneralSol[x] /. soln]
Out[•]=
```

$$\left\{1.\,e^{-0.2\,x}\,\text{Sin}[3.\,x]\right\}$$

- The solution is $y = e^{-0.2\,x}\sin(3\,x)$.
- **Step 4.** Verify the solution.

In[•]:=
```
ivpSoln[x_] := e^(-0.2` x) (0.` + 1.` Sin[3.` x]);
```

- Check for the initial conditions.

In[•]:=
```
ivpSoln[0]
```

Out[•]=
```
0.
```

In[•]:=
```
D[ivpSoln[x], {x, 1}] /. x → 0
```

Out[•]=
```
3.
```

- Check that the solution satisfies the given ODE $y'' + 0.4\,y' + 9.04\,y = 0$.

In[•]:=
```
LHS = FullSimplify[D[ivpSoln[x], {x, 2}] + 0.4 * D[ivpSoln[x], {x, 1}] + 9.04 * ivpSoln[x]]
```

Out[•]=
```
-1.72085 × 10^-15 e^(-0.2 x) Sin[3. x]
```

In[•]:=
```
Chop[LHS]
```

Out[•]=
```
0
```

In[•]:=
```
RHS = 0
```

Out[•]=
```
0
```

In[•]:=
```
Chop[LHS] == RHS
```

Out[•]=
```
True
```

- So the solution satisfies both the initial conditions and the ODE check.
- **Step 5.** Verify the solution by **DSolve** (Not Required).

In[•]:=
```
ClearAll[y]; DSolve[y''[x] + 0.4 y'[x] + 9.04 y[x] == 0, y[x], x]
```

Out[•]=
```
{{y[x] → e^(-0.2 x) C[2] Cos[3. x] + e^(-0.2 x) C[1] Sin[3. x]}}
```

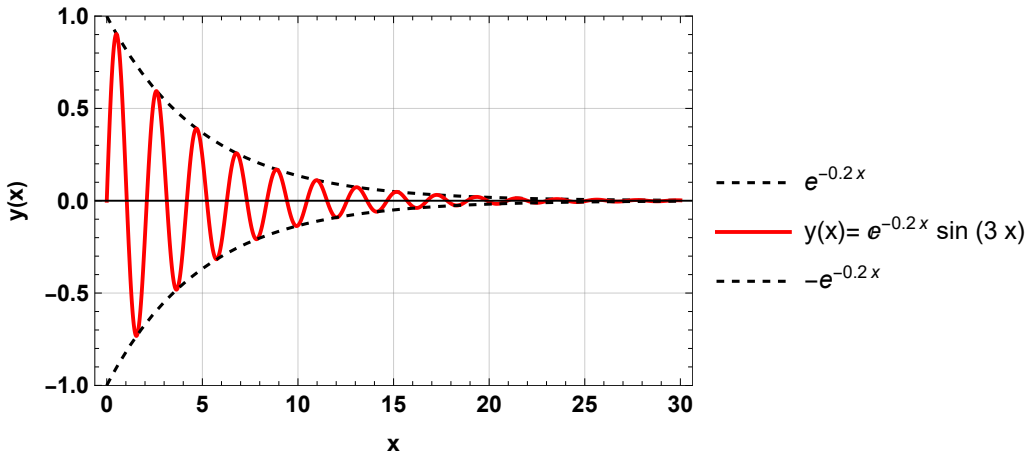
In[•]:=
```
yp = DSolve[{y''[x] + 0.4 y'[x] + 9.04 y[x] == 0, y[0] == 0, y'[0] == 3}, y[x], x]
```

Out[•]=
```
{{y[x] → 1. e^(-0.2 x) Sin[3. x]}}
```

- Using the DSolve function yields in the same result.
- Let's also take a look at the graph of the solution.

```
In[ ]:= Plot[{1.` e^(-0.2` x), ivpSoln[x], -1.` e^(-0.2` x)}, {x, 0, 30}, Frame → True,
 PlotStyle → {{Black, Dashed}, {Red, Thick}, {Black, Dashed}}, Frame → True,
 FrameLabel → {"x", "y(x)"}, BaseStyle → {FontWeight → "Bold", Black, FontSize → 12},
 PlotStyle → {Black}, GridLines → Automatic,
 PlotLegends → {"e^-0.2x", "y(x)= e^-0.2x sin (3 x)", "-e^-0.2x"},
 AxesStyle → Directive[RGBColor[0.`, 0.`, 0.`], AbsoluteThickness[1]],
 Method → {"DefaultBoundaryStyle" → Automatic, "DefaultMeshStyle" → AbsolutePointSize[6],
   "ScalingFunctions" → None}, PlotRange → {-1.0, 1.0}]
```

Out[ ]=



- The solution oscillates between $e^{-0.2x}$ and $-e^{-0.2x}$ functions.

3

Modeling of Free Oscillations of Mass-Spring System

The motion of the mechanical mass-spring system is determined by Newton's second law:

$$\text{Mass} \times \text{Acceleration} = m\,y'' = \text{Force}$$

There are two possible scenarios for the mass-spring system motion.

3.3

First Case. Undamped System.

The damping in the system is negligible. In this case the ODE of the **Undamped System** is as follows:

$$m\,y'' + k\,y = 0$$

where m is an object mass, k is the spring constant. This is a homogeneous linear ODE with constant coefficients, whose general solution is obtained easily

$$y(t) = A \cos \omega_0 t + B \sin \omega_0 t \qquad \omega_0 = \sqrt{\frac{k}{m}}$$

An alternative representation that shows physical characteristics of amplitude and phase shift is

$$y(t) = C \cos(\omega_0 t - \delta) \qquad C = \sqrt{A^2 + B^2} \qquad \tan \delta = \frac{B}{A}$$

2.1, 2.5

Example 2.5: Harmonic Oscillation of an Undamped Mass–Spring System

If a mass–spring system with an iron ball of weight W = 98 nt (about 22 lb) can be regarded as undamped, and the spring is such that the ball stretches it 1.09 m (about 43 in.), how many cycles per minute will the system execute? What will its motion be if we pull the ball down from rest by 16 cm (about 6 in.) and let it start with zero initial velocity?

- **Step 1.** Set up the model and determine the suitable ODE .
- Find the spring constant from Hooke's law.

In[ ]:=
```
ClearAll["Global`*"]
W = 98;
l = 1.09;
k = W / l
```

Out[ ]=
```
89.9083
```

- Find the mass of the object.

In[ ]:=
```
g = 9.81;
m = W / g
```

Out[ ]=
```
9.98981
```

- Find the frequency.

In[ ]:=
```
ω₀ = Sqrt[k / m]
```

Out[ ]=
```
3.
```

In[ ]:=
```
f = ω₀ / (2 Pi) (**In [Hz]**)
```

Out[ ]=
```
0.477465
```

In[ ]:=
```
fcpm = Round[f * 60] (**In [cycles per minute]**)
```
Out[ ]=

29

- Find the coefficients A and B using the initial conditions: $y(0) = 0.16$, $y'(0) = \omega_0 B = 0$

In[ ]:=
```
y[t_] := A * Cos[ω₀ * t] + B * Sin[ω₀ * t]
```

In[ ]:=
```
y0 = y[0]
```
Out[ ]=
```
0. + 1. A
```

In[ ]:=
```
Solve[y0 == 0.16, A]
```
Out[ ]=

{{A → 0.16}}

In[ ]:=
```
Solve[{ω₀ * B == 0}, B]
```
Out[ ]=

{{B → 0.}}

In[ ]:=
```
y[t] /. {A → 0.16, B → 0}
```
Out[ ]=

0.16 Cos[3. t]

- **Step 2.** Verify the solution.

In[ ]:=
```
ySoln[t_] := 0.16 * Cos[3 * t]
```

In[ ]:=
```
ySoln[0] == 0.16
```
Out[ ]=
```
True
```

In[ ]:=
```
D[ySoln[t] /. t → 0, t] == 0
```
Out[ ]=
```
True
```

In[ ]:=
```
LHS = FullSimplify[m * D[ySoln[t], {t, 2}] + k * ySoln[t]]
```
Out[ ]=
```
1.77636 × 10^-15 Cos[3 t]
```

In[ ]:=
```
RHS = 0
```
Out[ ]=
```
0
```

In[ ]:=
```
Chop[LHS] == RHS
```
Out[ ]=

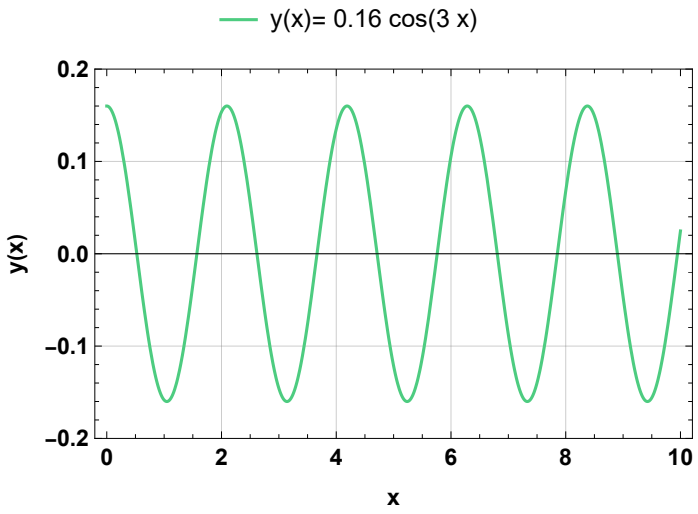
True

- So the solution satisfies both the initial conditions and the ODE check.
- **Step 3.** Verify the solution by **DSolve** (Not Required).

```
In[ ]:= ClearAll[y]; DSolve[m * y''[x] + k * y[x] == 0, y[x], x]
Out[ ]=
{{y[x] → 1. c1 Cos[3. x] + 1. c2 Sin[3. x]}}

In[ ]:= yp = DSolve[{m * y''[x] + k * y[x] == 0, y[0] == 0.16, y'[0] == 0}, y[x], x]
Out[ ]=
{{y[x] → 0.16 Cos[3. x]}}
```

- Using the DSolve function yields in the same result.
- Let's also take a look at the graph of the solution.

```
In[ ]:= Plot[y[x] /. yp, {x, 0, 10}, Frame → True, FrameLabel → {"x", "y(x)"},
  GridLines → Automatic, BaseStyle → {FontWeight → "Bold", Black, FontSize → 12},
  PlotRange → {-0.2, 0.2}, PlotStyle → RGBColor[0.3, 0.8, 0.5],
  PlotLegends → Placed[{"y(x)= 0.16 cos(3 x)"}, Above]]
Out[ ]=
```



Second Case. Damped System.

The system has a considerable damping. In this case the ODE of the **Damped Syst** follows:

$$m\,y'' + c\,y' + k\,y = 0$$

here c is called the ***damping constant***. This is a homogeneous linear ODE with constant coefficients. We can obtain the general solution by solving the characteristic equation as discussed before

$$\lambda^2 + \frac{c}{m}\,\lambda + \frac{k}{m} = 0$$

Again there are three cases with three different kind of roots, depending on the sign of the discrimanant $\left(\frac{c}{m}\right)^2 - 4\,\frac{k}{m}$.

Case I	$c^2 > 4\,m\,k$	*Distinct real roots* $\lambda_1,\ \lambda_2$.	**(Overdampling)**
Case II	$c^2 = 4\,m\,k$	*A real double root.*	**(Critical dampling)**
Case III	$c^2 < 4\,m\,k$	*Complex conjugate roots.*	**(Underdampling)**

As before, the solution to the ODE in each case is summarized below:

Case I. Overdamping

$$y(t) = c_1\,e^{-(\alpha-\beta)\,t} + c_2\,e^{-(\alpha+\beta)\,t} \qquad \alpha = \frac{c}{2\,m} \qquad \beta = \frac{1}{2\,m}\,\sqrt{c^2 - 4\,m\,k}$$

Case II. Critical damping

$$y(t) = (c_1 + c_2\,t)\,e^{-\alpha\,t} \qquad \alpha = \frac{c}{2\,m}$$

Case III. Underdamping

$$y(t) = e^{-\alpha\,t}(A\cos\omega^*\,t + B\sin\omega^*\,t) = C\,e^{-\alpha\,t}\cos(\omega^*\,t - \delta)$$

$$C^2 = A^2 + B^2 \qquad \delta = B/A \qquad \alpha = c/(2\,m)$$

$$\omega^* = \frac{1}{2\,m}\,\sqrt{4\,m\,k - c^2} = \sqrt{\frac{k}{m} - \frac{c^2}{4\,m^2}}$$

3.2

Example 2.6: The Three Cases of Damped Motion

If a mass–spring system in the Example 1 with an iron ball of mass m = 10 kg is regarded as damped, and the spring has a spring constant k = 90 N/m. We pull the ball down from rest by 16 cm (about 6 in.) and let it start with zero initial velocity as before. How does the motion change if we change the damping constant c from one to another of the

following three values?

(I) c = 100 kg/sec
(II) c = 60 kg/sec
(III) c = 10 kg/sec

```
In[•]:= ClearAll["Global`*"]
```

(I) $c = 100$ kg / sec

```
In[•]:= m = 10;
        k = 90;
        c = 100;

In[•]:= LHS = m * y'' + c * y' + k * y
        RHS = 0;

Out[•]=
        90 y + 100 y′ + 10 y′′
```

- Solve the characteristic equation.

```
In[•]:= roots = Solve[λ^2 + c/m * λ + k/m == 0, λ]

Out[•]=
        {{λ → -9}, {λ → -1}}
```

- There are two distinct roots, so we proceed with **Case I**, ***overdamping***. This gives the general solution:

```
In[•]:= λ1 = λ /. roots[[1]]; λ2 = λ /. roots[[2]]; {λ1, λ2}
        (** Double squared brackets [[]] get the ith element from the list**)

Out[•]=
        {-9, -1}

In[•]:= GeneralSol[x_] := c1 * Exp[λ1 * x] + c2 * Exp[λ2 * x]; GeneralSol[x]

Out[•]=
        c1 ℯ^(-9 x) + c2 ℯ^(-x)
```

- Find the particular solution using the initial conditions: $y(0) = 0.16$, $y'(0) = \omega_0 B = 0$

```
In[•]:= cond1 = GeneralSol[0] == 0.16;
        cond2 = (D[GeneralSol[x], x] /. x → 0) == 0;
```

- Solve for the arbitrary constants.

```
In[•]:= soln = Solve[{cond1, cond2}, {c1, c2}]

Out[•]=
        {{c1 → -0.02, c2 → 0.18}}
```

- Obtain the particular solution.

In[•]:=
```
yp1 = GeneralSol[x] /. soln
```
Out[•]=
$$\{-0.02\, e^{-9x} + 0.18\, e^{-x}\}$$

- Check the particular solution.

In[•]:=
```
ivpSoln[x_] := -0.02` ℯ^-9x + 0.18` ℯ^-x;
```
- Check for the initial conditions.

In[•]:=
```
ivpSoln[0]
```
Out[•]=
```
0.16
```

In[•]:=
```
D[ivpSoln[x], {x, 1}] /. x → 0
```
Out[•]=
```
0.
```

- Check that the solution satisfies the given ODE ***my'' + cy' + ky = 0***.

In[•]:=
```
FullSimplify[
 LHS /. {y'' → D[ivpSoln[x], {x, 2}], y' → D[ivpSoln[x], {x, 1}], y → ivpSoln[x]}]
```
Out[•]=
$$-2.88658 \times 10^{-15}\, ℯ^{-9x}$$

In[•]:=
```
Chop[FullSimplify[
   LHS /. {y'' → D[ivpSoln[x], {x, 2}], y' → D[ivpSoln[x], {x, 1}], y → ivpSoln[x]}]] == RHS
```
Out[•]=
```
True
```

- So the solution satisfies both the initial conditions and the ODE check.

(II) *c* = 60 kg / sec

In[•]:=
```
m = 10;
k = 90;
c = 60;
```

In[•]:=
```
LHS = m * y'' + c * y' + k * y
RHS = 0;
```
Out[•]=
$$90\, y + 60\, y' + 10\, y''$$

- Solve the characteristic equation.

In[•]:=
```
roots = Solve[λ^2 + c/m * λ + k/m == 0, λ]
```
Out[•]=
```
{{λ → -3}, {λ → -3}}
```

◆ There is a real double root, so we proceed with **Case II**, ***critical damping***. This gives the general solution.

```
In[•]:= λ1 = λ /. roots[[1]]; λ2 = λ /. roots[[2]]; {λ1, λ2}
        (** Double squared brackets [[]] get the ith element from the list**)
Out[•]=
        {-3, -3}

In[•]:= GeneralSol[x_] := (c1 + c2 * x) * Exp[λ1 * x]; GeneralSol[x]
Out[•]=
        e^(-3 x) (c1 + c2 x)
```

◆ Find the particular solution using the initial conditions: $y(0) = 0.16\,,\ y'(0) = \omega_0 B = 0$.

```
In[•]:= cond1 = GeneralSol[0] == 0.16;
        cond2 = (D[GeneralSol[x], x] /. x → 0) == 0;
```

◆ Solve for the arbitrary constants.

```
In[•]:= soln = Solve[{cond1, cond2}, {c1, c2}]
Out[•]=
        {{c1 → 0.16, c2 → 0.48}}
```

◆ Obtain the particular solution.

```
In[•]:= yp2 = GeneralSol[x] /. soln
Out[•]=
```

$$\left\{e^{-3\,x}\,(0.16 + 0.48\,x)\right\}$$

◆ Check the particular solution.

```
In[•]:= ivpSoln[x_] := e^(-3 x) (0.16` + 0.48` x);
```

◆ Check for the initial conditions.

```
In[•]:= ivpSoln[0]
Out[•]=
        0.16

In[•]:= D[ivpSoln[x], {x, 1}] /. x → 0
Out[•]=
        0.
```

◆ Check that the solution satisfies the given ODE $my'' + cy' + ky = 0$.

```
In[•]:= FullSimplify[
         LHS /. {y'' → D[ivpSoln[x], {x, 2}], y' → D[ivpSoln[x], {x, 1}], y → ivpSoln[x]}]
Out[•]=
        7.10543 × 10^-15 e^(-3 x) x

In[•]:= Chop[FullSimplify[
           LHS /. {y'' → D[ivpSoln[x], {x, 2}], y' → D[ivpSoln[x], {x, 1}], y → ivpSoln[x]}]] == RHS
```

Out[ ]=

```
True
```

- So the solution satisfies both the initial conditions and the ODE check.

(III) $c = 10$ kg / sec

In[ ]:=
```
m = 10;
k = 90;
c = 10;
```

In[ ]:=
```
LHS = m * y'' + c * y' + k * y
RHS = 0;
```

Out[ ]=

$$90\,y + 10\,y' + 10\,y''$$

- Solve the characteristic equation.

In[ ]:=
```
roots = Solve[λ^2 + c/m * λ + k/m == 0, λ]
```

Out[ ]=

$$\left\{\left\{\lambda \to \frac{1}{2}\left(-1 - i\sqrt{35}\right)\right\}, \left\{\lambda \to \frac{1}{2}\left(-1 + i\sqrt{35}\right)\right\}\right\}$$

- Find the general solution. We got two complex roots, so we proceed with **Case III**.

In this case, the roots of the characteristic equation are complex numbers that give the complex solutions of the ODE. However, it can be shown that we can obtain a basis of real solutions:

$$y_1 = e^{-ax/2}\cos\omega x \quad \text{and} \quad y_2 = e^{-ax/2}\sin\omega x \quad \text{where} \quad \omega^* = \sqrt{\frac{k}{m} - \frac{c^2}{4m^2}}$$

In[ ]:=
```
ω* = Sqrt[k/m - c^2/(4 m^2)]
```

Out[ ]=

$$\frac{\sqrt{35}}{2}$$

In[ ]:=
```
ClearAll[A, B];
GeneralSol[x_] := (A * Cos[ω* * x] + B * Sin[ω* * x]) * Exp[(-c / (2 m)) * x];
GeneralSol[x]
```

Out[ ]=

$$e^{-x/2}\left(A\,\text{Cos}\left[\frac{\sqrt{35}\,x}{2}\right] + B\,\text{Sin}\left[\frac{\sqrt{35}\,x}{2}\right]\right)$$

- Find the particular solution using the initial conditions: $y(0) = 0.16$, $y'(0) = \omega_0 B = 0$.

In[ ]:=
```
cond1 = GeneralSol[0] == 0.16;
cond2 = (D[GeneralSol[x], x] /. x → 0) == 0;
```

◆ Solve for the arbitrary constants.

```
In[ ]:= soln = Solve[{cond1, cond2}, {A, B}]
Out[ ]=
{{A → 0.16, B → 0.0270449}}
```

◆ Obtain the particular solution.

```
In[ ]:= yp3 = GeneralSol[x] /. soln
```

Out[ ]=

$$\left\{e^{-x/2}\left(0.16\,\text{Cos}\left[\frac{\sqrt{35}\,x}{2}\right]+0.0270449\,\text{Sin}\left[\frac{\sqrt{35}\,x}{2}\right]\right)\right\}$$

◆ Check the particular solution.

```
In[ ]:= ivpSoln[x_] := e^(-x/2) (0.16` Cos[Sqrt[35] x/2] + 0.02704493615131253` Sin[Sqrt[35] x/2]);
```

◆ Check for the initial conditions.

```
In[ ]:= ivpSoln[0]
Out[ ]=
0.16

In[ ]:= D[ivpSoln[x], {x, 1}] /. x → 0
Out[ ]=
0.
```

◆ Check that the solution satisfies the given ODE $my'' + cy' + ky = 0$.

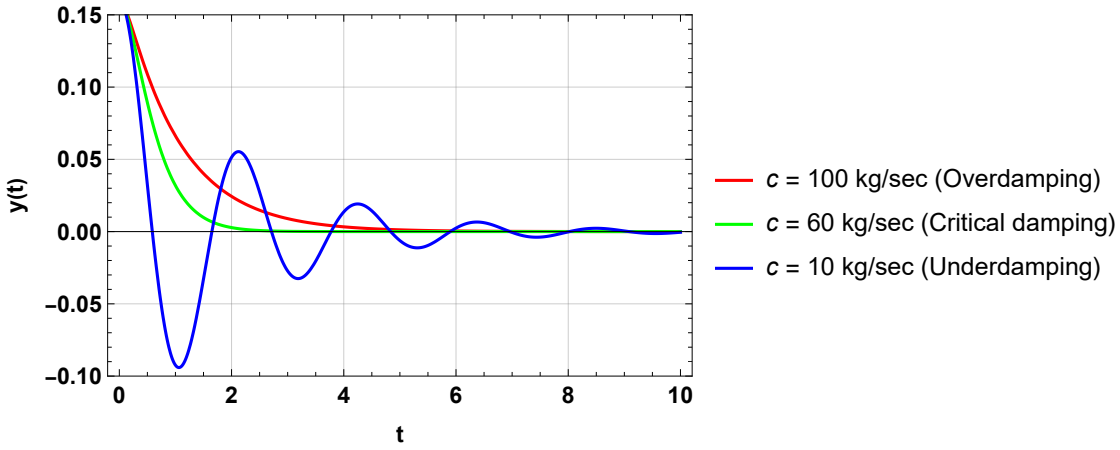
```
In[ ]:= FullSimplify[
 LHS /. {y'' → D[ivpSoln[x], {x, 2}], y' → D[ivpSoln[x], {x, 1}], y → ivpSoln[x]}]
Out[ ]=
e^(-x/2) (-1.77636 × 10^-15 Cos[Sqrt[35] x/2] - 4.44089 × 10^-16 Sin[Sqrt[35] x/2])

In[ ]:= Chop[FullSimplify[
    LHS /. {y'' → D[ivpSoln[x], {x, 2}], y' → D[ivpSoln[x], {x, 1}], y → ivpSoln[x]}]] == RHS
Out[ ]=
True
```

◆ So the solution satisfies both the initial conditions and the ODE check.

◆ Let's plot three curves at the same graph.

In[ ]:=
```
Plot[{yp1, yp2, yp3}, {x, 0, 10}, Frame → True, FrameLabel → {"t", "y(t)"},
 GridLines → Automatic, BaseStyle → {FontWeight → "Bold", Black, FontSize → 12},
 PlotRange → {-0.1, 0.15}, PlotStyle → {Red, Green, Blue},
 PlotLegends → {"c = 100 kg/sec (Overdamping)",
   "c = 60 kg/sec (Critical damping)", "c = 10 kg/sec (Underdamping)"}]
```

Out[ ]=



4

Wolfram Demonstrations Project: Unforced, Damped, Simple Harmonic Motion

This Demonstration illustrates unforced, damped, simple harmonic motion using the standard mass and spring setup. By manipulating the mass, Hooke's constant, and the damping coefficient parameters, you can observe the phenomenon of critical dampening.

The source code below was developed by John Erickson, Chicago State University (2009).

Open content licensed under CC BY-NC-SA.

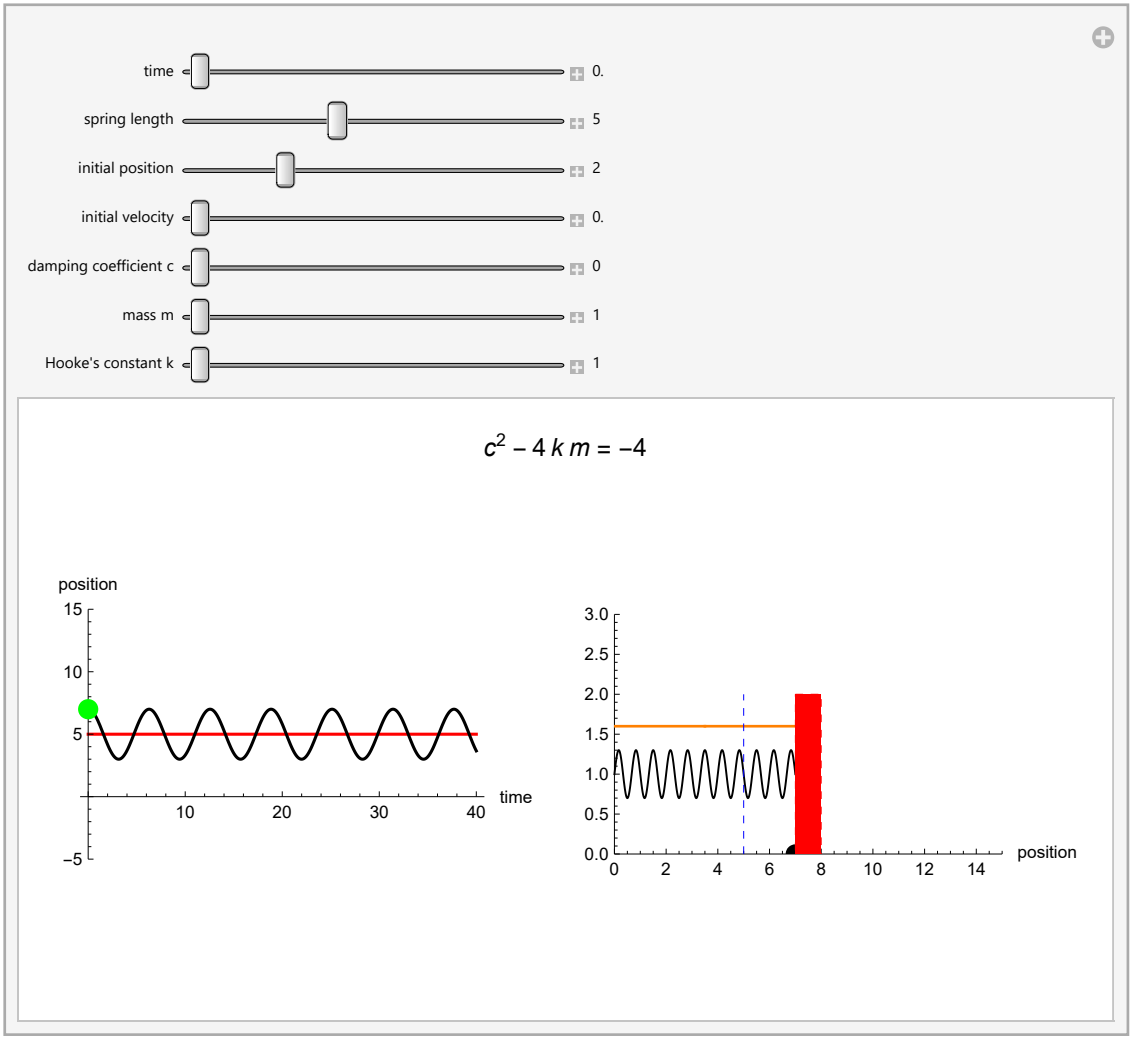
John Erickson, Chicago State University
"Unforced, Damped, Simple Harmonic Motion"
https://demonstrations.wolfram.com/UnforcedDampedSimpleHarmonicMotion/
Wolfram Demonstrations Project; Published: March 10, 2009; Accessed on July 26, 2022.

In[ ]:=
```
ClearAll["Global`*"]
```

In[ ]:=
```
solPlotDamp[springLength_, initEquibPos_, initVel_, mm_, kk_, cc_, tt_] :=
 Module[{x, t},
  sol = x[t] /. DSolve[{x''[t] + cc/mm x'[t] + kk/mm x[t] == 0,
      x[0] == initEquibPos, x'[0] == initVel}, x[t], t][[1]];

  Column[{Text@TraditionalForm@Row[{c^2 - 4 k m, " = ", cc^2 - 4 kk mm}],
    GraphicsGrid[{{Plot[{springLength, springLength + sol}, {t, 0, 40}, PlotRange →
        {-5, 15}, PlotStyle → {Red, Black}, AxesLabel → {"time", "position"},
       Epilog → {Black, PointSize[.03], Point[{0, springLength + sol /. t → tt}],
         Green, PointSize[.05], Point[{tt, springLength + sol /. t → tt}]}],
      Plot[1 + .3 Sin[π s (5 - sol /. t → tt)], {s, 0, springLength + sol /. t → tt},
       PlotRange → {{0, 15}, {0, 3}}, PlotStyle → {Black, Thickness[.005 kk]},
       AxesLabel → {"position", None}, Epilog → {{Orange, Thickness[.03 cc],
          Line[{{0, 1.6}, {(springLength + sol /. t → tt)/2, 1.6}}],
          Thickness[.06 cc], Line[{{(springLength + sol /. t → tt)/2, 1.6},
            {springLength + sol /. t → tt, 1.6}}]},
         Blue, Dashed, Line[{{springLength, 0}, {springLength, 2}}],
         Black, PointSize[.05], Point[{springLength + sol /. t → tt, 0}],
         Red, Rectangle[{springLength + sol /. t → tt, 0},
          {springLength + mm + sol /. t → tt, 2}]}]}, ImageSize → {540, 270}]}, Center]
 ]
```

In[ ]:=
```
Manipulate[solPlotDamp[springLength, initEquibPos, initVel, mm, kk, cc, tt],
 {{tt, 0.0, "time"}, 0, 40, Appearance → "Labeled"},
 {{springLength, 5, "spring length"}, 3, 8, Appearance → "Labeled"},
 {{initEquibPos, 2, "initial position"}, 1, 5, Appearance → "Labeled"},
 {{initVel, 0.0, "initial velocity"}, 0, 1, Appearance → "Labeled"},
 {{cc, 0, "damping coefficient c"}, 0, 2, Appearance → "Labeled"},
 {{mm, 1, "mass m"}, 1, 4, Appearance → "Labeled"},
 {{kk, 1, "Hooke's constant k"}, 1, 3, Appearance → "Labeled"},
 ControlPlacement → Top, SaveDefinitions → True, SynchronousUpdating → False]
```

Out[ ]=



5

Summary

After completing this chapter, you should be able to

- solve 2nd-order linear homogeneous ODEs step-by-step using Wolfram Mathematica.

- develop SOPs to solve 2nd-order linear homogeneous ODEs.
- develop the habit of always checking your solutions for quality assurance.
- learn and use information, tools, and technology to solve engineering math problems.

Week 3: Second-Order ODEs (Part 2)

How to Solve Second-Order ODEs Step-by-step?

Table of Contents

Commands list

- Sqrt[*z*]
- Exp[*z*]
- Collect[*expr,x*]
- Chop[*expr*]
- Plot[*f, {x, x_min, x_max}*]

Nonhomogeneous Linear ODEs of Second Order

The standard form of the **Nonhomogeneous Linear Second Order** ODE is as follows:

$$y'' + p(x)\,y' + q(x)\,y = r(x)$$

The **general solution** of the nonhomogeneous ODE on the open interval I is

$$y(x) = y_h(x) + y_p(x)$$

here $y_h(x) = c_1\,y_1 + c_2\,y_2$ is the general solution of the homogeneous ODE on the same I interval. We learned how to solve it earlier (by solving the characteristic equation).

Case	Roots of (2)	Basis of (1)	General Solution of (1)
I	Distinct real $\lambda_1,\ \lambda_1$	$e^{\lambda_1 x},\ e^{\lambda_2 x}$	$y = c_1 e^{\lambda_1 x} + c_2 e^{\lambda_2 x}$
II	Real double root $\lambda = -\frac{1}{2} a$	$e^{-ax/2},\ x e^{-ax/2}$	$y = (c_1 + c_2 x) e^{-ax/2}$
III	Complex conjugate $\lambda_1 = -\frac{1}{2} a + i\omega,$ $\lambda_2 = -\frac{1}{2} a - i\omega$	$e^{-ax/2} \cos \omega x$ $e^{-ax/2} \sin \omega x$	$y = e^{-ax/2}$ (A *cos* ωx + B *sin* ωx)

The $y_p(x)$ is any solution on I containing no arbitrary constants. It can be found by using the **Method of Undetermined Coefficients.** The method is suitable for linear ODEs with constant coefficients a and b:

$$y'' + a\,y' + b\,y = r(x)$$

Term in $r(x)$	Choice for $y_p(x)$
$ke^{\gamma x}$	$Ce^{\gamma x}$
$kx^n \quad (n = 0,1,\cdots)$	$K_n x^n + K_{n-1} x^{n-1} + \cdots + K_1 x + K_0$
$k \cos \omega x$ $k \sin \omega x$	} $K \cos \omega x + M \sin \omega x$
$ke^{\alpha x} \cos \omega x$ $ke^{\alpha x} \sin \omega x$	} $e^{\alpha x} (K \cos \omega x + M \sin \omega x)$

Note 1: If a term in your choice for $y_p(x)$ happens to be a solution of the homogeneous ODE, use the **Modification Rule** (multiply this term by x or by x^2).

Note 2: If a term in your choice for $y_p(x)$ happens to be a sum of functions in the first column of the Table above, then for $y_p(x)$ choose the sum of the functions in the corresponding lines of the second column **(Sum Rule).**

Example 3.1. Method of Undetermined Coefficients

$$y'' + y = 0.001\,x^2 \qquad y(0) = 0,\ y'(0) = 1.5$$

```
In[•]:= ClearAll["Global`*"]

In[•]:= LHSOp[y_, x_] = y''[x] + y[x]
Out[•]=
        y[x] + y''[x]

In[•]:= rhsFunc[x_] = 0.001 x^2
Out[•]=
        0.001 x^2
```

- **Step 1.** Solve the corresponding homogeneous ODE to obtain the general solution o $y_h(x)$.
- To do so, let's solve the characteristic equation.

```
In[•]:= a = 0; b = 1;
        roots = Solve[λ^2 + a * λ + b == 0, λ]
Out[•]=
        {{λ → -i}, {λ → i}}
```

- We got two complex roots, so we proceed with **Case III.**

In this case, the roots of the characteristic equation are complex numbers that give the complex solutions of the ODE. However, it can be shown that we can obtain a basis of real solutions:

$$y_1 = e^{-ax/2} \cos \omega x \quad \text{and} \quad y_2 = e^{-ax/2} \sin \omega x \quad \text{where} \quad \omega = \sqrt{b - \frac{a^2}{4}}$$

```
In[•]:= ω = Sqrt[b - a^2 / 4]
Out[•]=
        1

In[•]:= ClearAll[A, B];
        yh[x_] := (A * Cos[ω * x] + B * Sin[ω * x] ) * Exp[(-a / 2) * x];
        yh[x]
Out[•]=
        A Cos[x] + B Sin[x]
```

- **Step 2.** Applying the **method of undetermined coefficients**, find a solution $y_p(x)$.
- Since the $r(x)$ term is in the form of $k\,x^n$ for $(n = 0, 1, \ldots)$, the corresponding $y_p(x)$ choice (second row in the Table) is $y_p = K_2\,x^2 + K_1\,x + K_0$.
- K_2, K_1, K_0 are coefficients to be determined

```
In[•]:= yp[x_] = K2 * x^2 + K1 * x + K0
Out[•]=
        K0 + K1 x + K2 x^2
```

- Let's plug the assumed solution $y_p(x)$ into the LHS:

In[ ]:=
```
LHS = LHSOp[yp, x]
```
Out[ ]=
$$K0 + 2\,K2 + K1\,x + K2\,x^2$$

In[ ]:=
```
RHS = rhsFunc[x]
```
Out[ ]=
$$0.001\,x^2$$

- Next, equate coefficients of x and x^2 because the coefficient of each power of x must be the same on both sides. Hence, LHS-RHS must be zero for all x.

In[ ]:=
```
Q = Collect[(LHS - RHS), {x, x^2}]
```
Out[ ]=
$$K0 + 2\,K2 + K1\,x + (-0.001 + K2)\,x^2$$

- The conditions that all coefficients must be 0 gives us 3 equations with 3 unknowns, which we can solve using **Solve**[ ].

In[ ]:=
```
eqn0 = K0 + 2 K2 == 0 (** constant term **);
eqn1 = K1 == 0 (**coefficient of x **);
eqn2 = (-0.001` + K2) == 0 (**coefficient of x^2 **);
```

In[ ]:=
```
coeffSoln = Solve[{eqn0, eqn1, eqn2}, {K2, K1, K0}]
```
Out[ ]=
$$\{\{K2 \to 0.001,\ K1 \to 0,\ K0 \to -0.002\}\}$$

- Let's substitute the coefficients to the solution.

In[ ]:=
```
ypSoln[x_] = yp[x] /. coeffSoln[[1]]
```
Out[ ]=
$$-0.002 + 0.001\,x^2$$

- Check the solution.

In[ ]:=
```
LHSCheck = LHSOp[ypSoln, x]
```
Out[ ]=
$$0. + 0.001\,x^2$$

In[ ]:=
```
FullSimplify[Chop[LHSCheck]]
```
Out[ ]=
$$0.001\,x^2$$

In[ ]:=
```
FullSimplify[Chop[LHSCheck]] == RHS
```
Out[ ]=
```
True
```

- Great! The same as the right hand side.
- **Step 3.** Then, the general solution to the initial nonhomogeneous ODE is:

```
In[•]:= ygeneral[x_] = yh[x] + ypSoln[x]
Out[•]=
-0.002 + 0.001 x^2 + A Cos[x] + B Sin[x]
```

- **Step 4.** Find the particular solution using the initial conditions: $y(0) = 0,\ y'(0) = 1.5$.
- Find the derivative $y' = \frac{dy}{dx}$.

```
In[•]:= dygeneral[x_] = D[ygeneral[x], {x, 1}]
Out[•]=
0.002 x + B Cos[x] - A Sin[x]

In[•]:= IC1 = ygeneral[0] == 0
Out[•]=
-0.002 + A == 0

In[•]:= IC2 = dygeneral[0] == 1.5
Out[•]=
0. + B == 1.5

In[•]:= valuesofcofficients = Solve[{IC1, IC2}, {A, B}]
Out[•]=
{{A → 0.002, B → 1.5}}
```

- Hence, the particular solution to the given ODE is:

```
In[•]:= yparticular[x_] = ygeneral[x] /. valuesofcofficients[[1]]
Out[•]=
```

$$-0.002 + 0.001\,x^2 + 0.002\,\text{Cos}[x] + 1.5\,\text{Sin}[x]$$

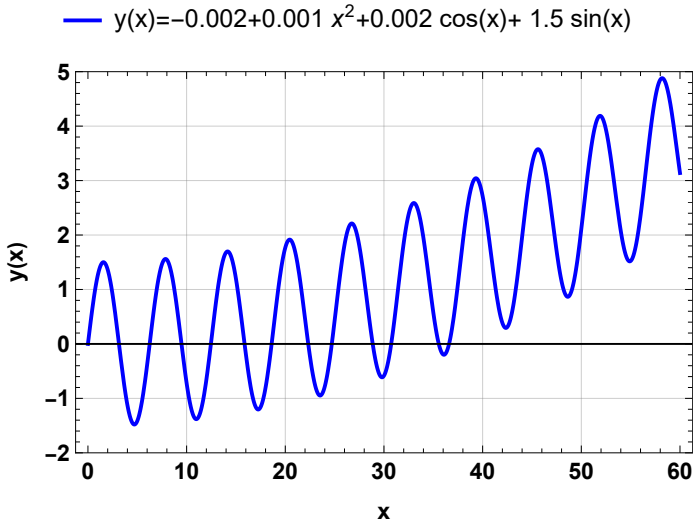
- **Step 5.** Verify the solution to the ODE $y'' + y = 0.001\,x^2$ with initial conditions: $y(0) = 0,\ y'(0) = 1.5$.

```
In[•]:= LHSOp[yparticular, x]
Out[•]=
0. + 0.001 x^2

In[•]:= FullSimplify[Chop[LHSOp[yparticular, x]]] == rhsFunc[x]
Out[•]=
True

In[•]:= yparticular[0]
Out[•]=
0.

In[•]:= D[yparticular[x], {x, 1}] /. {x → 0}
Out[•]=
1.5
```

- The solution satisfies both the ODE and initial conditions check!
- Let's take a look at the graph of the solution.

```
In[•]:= Plot[yparticular[x], {x, 0, 60}, Frame → True,
  PlotStyle → {{Blue, Thick}}, Frame → True, FrameLabel → {"x", "y(x)"},
  PlotLegends → Placed[{"y(x)=-0.002+0.001 x^2+0.002 cos(x)+ 1.5 sin(x)"}, Above],
  BaseStyle → {FontWeight → "Bold", Black, FontSize → 12}, GridLines → Automatic,
  AxesStyle → Directive[RGBColor[0.`, 0.`, 0.`], AbsoluteThickness[1]],
  Method → {"DefaultBoundaryStyle" → Automatic, "DefaultMeshStyle" → AbsolutePointSize[6],
    "ScalingFunctions" → None}, PlotRange → {-2, 5}]
```

Out[•]=



- **Step 6.** Solve the ODE using a built-in **DSolve** function (Not Required).

```
In[•]:= DSolve[LHSOp[y, x] == rhsFunc[x] , y[x], x] (** A general solution **)
```

Out[•]=

$$\{\{y[x] \to -0.002 + 0.001\, x^2 + 1.\, c_1 \text{Cos}[1.\, x] + 1.\, c_2 \text{Sin}[1.\, x]\}\}$$

```
In[•]:= DSolve[{LHSOp[y, x] == rhsFunc[x], y[0] == 0, y'[0] == 1.5} , y[x], x]
(** A particular solution **)
```

Out[•]=

$$\{\{y[x] \to -0.002 + 0.001\, x^2 + 0.002\, \text{Cos}[1.\, x] + 1.5\, \text{Sin}[1.\, x]\}\}$$

Example 3.2. Application of Modification Rule

$$y'' + 3\,y' + 2.25\,y = -10\,e^{-1.5\,x} \qquad y(0) = 1,\ y'(0) = 0$$

```
In[•]:= ClearAll["Global`*"]
```

```
In[•]:= LHSOp[y_, x_] = y''[x] + 3 y'[x] + 2.25 y[x]
```

Out[•]=

$$2.25\,y[x] + 3\,y'[x] + y''[x]$$

```
In[•]:= rhsFunc[x_] = -10 Exp[-1.5 x]
Out[•]=
-10 e^(-1.5 x)
```

- **Step 1.** Solve the corresponding homogeneous ODE to obtain the general solution o $y_h(x)$.
- To do so, let's solve the characteristic equation.

```
In[•]:= a = 3; b = 2.25;
roots = Solve[λ^2 + a * λ + b == 0, λ]
Out[•]=
{{λ → -1.5}, {λ → -1.5}}
```

- We got a real double root, so we proceed with **Case II.**

```
In[•]:= λ1 = λ /. roots[[1]]; λ2 = λ /. roots[[2]]; {λ1 , λ2 }
Out[•]=
{-1.5, -1.5}
```

```
In[•]:= yh[x_] := (c1 + c2 * x) * Exp[λ1 * x]; yh[x]
Out[•]=
e^(-1.5 x) (c1 + c2 x)
```

- **Step 2.** Applying the **method of undetermined coefficients**, find a solution $y_p(x)$.
- Since the $r(x)$ term is in the form of $k\, e^{\gamma x}$, the corresponding $y_p(x)$ choice (first row in the Table) is $y_p = K_0\, e^{-1.5\,x}$. K_0 is a coefficient to be determined.
- **Warning!** Here we need to use the **Modification Rule**, because $y_p(x)$ term happens to be the solution to the corresponding homogeneous equation. Thus, **multiply** the term by x and get the correct expression of form $y_p = K_0\ x^2\, e^{-1.5\,x}$.

```
In[•]:= yp[x_] = K0 * x^2 * Exp[-1.5 x]
Out[•]=
e^(-1.5 x) K0 x^2
```

- Let's plug the assumed solution $y_p(x)$ into the LHS.

```
In[•]:= LHS = LHSOp[yp, x]
Out[•]=
2 e^(-1.5 x) K0 - 6. e^(-1.5 x) K0 x + 4.5 e^(-1.5 x) K0 x^2 + 3 (2 e^(-1.5 x) K0 x - 1.5 e^(-1.5 x) K0 x^2)
```

```
In[•]:= RHS = rhsFunc[x]
Out[•]=
-10 e^(-1.5 x)
```

```
In[•]:= LHS - RHS
Out[•]=
10 e^(-1.5 x) + 2 e^(-1.5 x) K0 - 6. e^(-1.5 x) K0 x + 4.5 e^(-1.5 x) K0 x^2 + 3 (2 e^(-1.5 x) K0 x - 1.5 e^(-1.5 x) K0 x^2)
```

- Next, equate coefficients of x and x^2 because the coefficient of each power of x must be the same on both sides. Hence, LHS-RHS must be zero for all x.

```
In[•]:= Q = Collect[(LHS - RHS), {Exp[-1.5 x], x * Exp[-1.5 x], x^2 * Exp[-1.5 x]}]
Out[•]=
e^(-1.5 x) (10 + 2 K0)
```

- The condition that all coefficients must be 0 gives us only one equation with the unknown K_0, which we can solve using **Solve**[ ].

```
In[•]:= eqn0 = 10 + 2 K0 == 0
Out[•]=
10 + 2 K0 == 0

In[•]:= coeffSoln = Solve[{eqn0}, {K0}]
Out[•]=
{{K0 → -5}}
```

- Let's substitute the coefficient to the solution.

```
In[•]:= ypSoln[x_] = yp[x] /. coeffSoln[[1]]
Out[•]=
-5 e^(-1.5 x) x^2
```

- Check the solution.

```
In[•]:= LHSCheck = LHSOp[ypSoln, x]
Out[•]=
-10 e^(-1.5 x) + 30. e^(-1.5 x) x - 22.5 e^(-1.5 x) x^2 + 3 (-10 e^(-1.5 x) x + 7.5 e^(-1.5 x) x^2)

In[•]:= FullSimplify[Chop[LHSCheck]]
Out[•]=
-10. e^(-1.5 x)

In[•]:= FullSimplify[Chop[LHSCheck]] == RHS
Out[•]=
True
```

- Great! The same as the right hand side.
- **Step 3.** Then, the general solution to the initial nonhomogeneous ODE is:

```
In[•]:= ygeneral[x_] = yh[x] + ypSoln[x]
Out[•]=
-5 e^(-1.5 x) x^2 + e^(-1.5 x) (c1 + c2 x)
```

- **Step 4.** Find the particular solution using the initial conditions: $y(0) = 1,\ y'(0) = 0$.
- Find the derivative $y' = \frac{dy}{dx}$.

```
In[•]:= dygeneral[x_] = D[ygeneral[x], {x, 1}]
Out[•]=
c2 e^-1.5 x - 10 e^-1.5 x x + 7.5 e^-1.5 x x^2 - 1.5 e^-1.5 x (c1 + c2 x)

In[•]:= IC1 = ygeneral[0] == 1
Out[•]=
0. + 1. c1 == 1

In[•]:= IC2 = dygeneral[0] == 0
Out[•]=
0. - 1.5 c1 + 1. c2 == 0

In[•]:= valuesofcofficients = Solve[{IC1, IC2}, {c1, c2}]
Out[•]=
{{c1 → 1., c2 → 1.5}}
```

- Hence, the particular solution to the given ODE is:

```
In[•]:= yparticular[x_] = ygeneral[x] /. valuesofcofficients[[1]]
Out[•]=
-5 e^-1.5 x x^2 + e^-1.5 x (1. + 1.5 x)
```

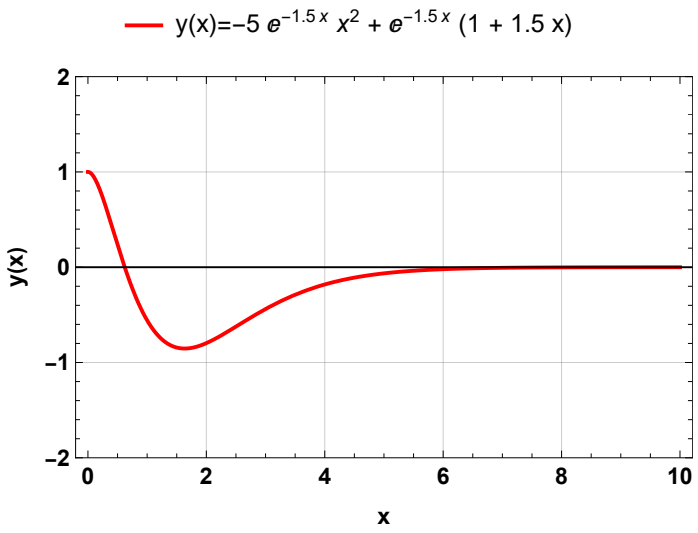
- **Step 5.** Verify the solution to the ODE $y'' + 3y' + 2.25y = -10e^{-1.5x}$ with initial conditions: $y(0) = 1,\ y'(0) = 0$.

```
In[•]:= LHSOp[yparticular, x]
Out[•]=
-14.5 e^-1.5 x + 30. e^-1.5 x x - 11.25 e^-1.5 x x^2 + 2.25 e^-1.5 x (1. + 1.5 x) +
 3 (1.5 e^-1.5 x - 10 e^-1.5 x x + 7.5 e^-1.5 x x^2 - 1.5 e^-1.5 x (1. + 1.5 x)) +
 2.25 (-5 e^-1.5 x x^2 + e^-1.5 x (1. + 1.5 x))

In[•]:= FullSimplify[Chop[LHSOp[yparticular, x]]] == rhsFunc[x]
Out[•]=
True

In[•]:= yparticular[0]
Out[•]=
1.

In[•]:= D[yparticular[x], {x, 1}] /. {x → 0}
Out[•]=
0.
```

- The solution satisfies both the ODE and initial conditions check!
- Let's take a look at the graph of the solution.

```
In[•]:= Plot[yparticular[x], {x, 0, 10}, Frame → True,
 PlotStyle → {{Red, Thick}}, Frame → True, FrameLabel → {"x", "y(x)"},
 PlotLegends → Placed[{"y(x)=-5 ℯ^-1.5 x x^2 + ℯ^-1.5 x (1 + 1.5 x)"}, Above],
 BaseStyle → {FontWeight → "Bold", Black, FontSize → 12}, GridLines → Automatic,
 AxesStyle → Directive[RGBColor[0.`, 0.`, 0.`], AbsoluteThickness[1]],
 Method → {"DefaultBoundaryStyle" → Automatic, "DefaultMeshStyle" → AbsolutePointSize[6],
   "ScalingFunctions" → None}, PlotRange → {-2, 2}]
```

Out[•]=



◆ **Step 6.** Solve the ODE using a built-in **DSolve** function (Not Required).

```
In[•]:= DSolve[LHSOp[y, x] == rhsFunc[x] , y[x], x] (** A general solution **)
```

Out[•]=

$$\{\{y[x] \to -5.\, e^{-1.5\,x}\, x^2 + e^{-1.5\,x}\, C_1 + e^{-1.5\,x}\, x\, C_2\}\}$$

```
In[•]:= DSolve[{LHSOp[y, x] == rhsFunc[x], y[0] == 1, y'[0] == 0} , y[x], x]
 (** A particular solution **)
```

Out[•]=

$$\{\{y[x] \to -5.\, e^{-1.5\,x}\,(-0.2 - 0.3\,x + 1.\,x^2)\}\}$$

```
In[•]:= FullSimplify[yparticular[x]]
 (** The solution that we obtained through a step-by-step SOP **)
```

Out[•]=

$$-5.\, e^{-1.5\,x}\,(-0.2 + (-0.3 + x)\,x)$$

Example 3.3. Application of Sum Rule

$$y'' + 2\,y' + 0.75\,y = 2\cos(x) - 0.25\sin(x) + 0.09\,x \qquad y(0) = 2.78,\ y'(0) = -0.43$$

```
In[•]:= ClearAll["Global`*"]
```

◆ Create expressions for the LHS operator and the RHS function.

```
In[•]:= LHSOp[y_, x_] = y''[x] + 2 y'[x] + 0.75 y[x]
Out[•]=
0.75 y[x] + 2 y'[x] + y''[x]

In[•]:= rhsFunc[x_] = 2 Cos[x] - 0.25 Sin[x] + 0.09 x
Out[•]=
0.09 x + 2 Cos[x] - 0.25 Sin[x]
```

- **Step 1.** Solve the corresponding homogeneous ODE to obtain the general solution o $y_h(x)$.
- To do so, let's solve the characteristic equation.

```
In[•]:= a = 2; b = 0.75;
        roots = Solve[λ^2 + a * λ + b == 0, λ]
Out[•]=
{{λ → -1.5}, {λ → -0.5}}
```

- We got two distinct roots, so we proceed with **Case I** to obtain the general solution.

```
In[•]:= λ1 = λ /. roots[[1]]; λ2 = λ /. roots[[2]]; {λ1, λ2}
Out[•]=
{-1.5, -0.5}

In[•]:= yh[x_] := c1 * Exp[λ1 * x] + c2 * Exp[λ2 * x]; yh[x]
Out[•]=
c1 ℯ^(-1.5 x) + c2 ℯ^(-0.5 x)
```

- **Step 2.** Applying the **method of undetermined coefficients**, find a solution $y_p(x)$.
- Since the $r(x)$ term is the sum of several functions

$$r(x) = 2\cos(x) - 0.25\sin(x) + 0.09\,x$$

- The corresponding $y_p(x)$ choice (from the Undetermined Coefficients Table) is $y_p = K\cos(x) + M\sin(x) + K_1 x + K_0$ (based on the **Sum Rule**).
- K, M, K_1, K_0 are coefficients to be determined. Don't also forget to check with the modification rule.

```
In[•]:= yp[x_] = M1 * Cos[x] + M2 * Sin[x] + K1 * x + K0
Out[•]=
K0 + K1 x + M1 Cos[x] + M2 Sin[x]
```

- **Note:** In Wolfram Language the variable cannot be named "K", because it is already built-in symbol, so use M_1, M_2 instead.

In[ ]:=
```
? K
```

Out[ ]=

Symbol

K is a default generic name for a summation index in a symbolic sum.

- Let's plug the assumed solution $y_p(x)$ into the LHS.

In[ ]:=
```
LHS = LHSOp[yp, x]
```

Out[ ]=
```
-M1 Cos[x] - M2 Sin[x] + 2 (K1 + M2 Cos[x] - M1 Sin[x]) + 0.75 (K0 + K1 x + M1 Cos[x] + M2 Sin[x])
```

In[ ]:=
```
RHS = rhsFunc[x]
```

Out[ ]=
```
0.09 x + 2 Cos[x] - 0.25 Sin[x]
```

In[ ]:=
```
LHS - RHS
```

Out[ ]=
```
-0.09 x - 2 Cos[x] - M1 Cos[x] + 0.25 Sin[x] - M2 Sin[x] +
 2 (K1 + M2 Cos[x] - M1 Sin[x]) + 0.75 (K0 + K1 x + M1 Cos[x] + M2 Sin[x])
```

- Next, equate coefficients of x and x^2 because the coefficient of each power of x must be the same on both sides. Hence, LHS-RHS must be zero for all x.

In[ ]:=
```
Q = Collect[(LHS - RHS), {x, Cos[x], Sin[x]}]
```

Out[ ]=
```
0.75 K0 + 2 K1 + (-0.09 + 0.75 K1) x + (-2 - 0.25 M1 + 2 M2) Cos[x] + (0.25 - 2 M1 - 0.25 M2) Sin[x]
```

- The condition that all coefficients must be 0 gives us 4 equations with 4 unknowns, which we can solve using **Solve**[ ].

In[ ]:=
```
eqn0 = 0.75` K0 + 2 K1 == 0;
eqn1 = -0.09` + 0.75` K1 == 0;
eqn2 = -2 - 0.25` M1 + 2 M2 == 0;
eqn3 = 0.25` - 2 M1 - 0.25` M2 == 0;
```

In[ ]:=
```
coeffSoln = Solve[{eqn0, eqn1, eqn2, eqn3}, {K0, K1, M1, M2}]
```

Out[ ]=
```
{{K0 → -0.32, K1 → 0.12, M1 → 0., M2 → 1.}}
```

- Let's substitute the coefficients to the solution.

In[ ]:=
```
ypSoln[x_] = yp[x] /. coeffSoln[[1]]
```

Out[ ]=
```
-0.32 + 0.12 x + 1. Sin[x]
```

- Check the solution.

```
In[•]:= LHSCheck = LHSOp[ypSoln, x]
Out[•]=
2 (0.12 + 1. Cos[x]) - 1. Sin[x] + 0.75 (-0.32 + 0.12 x + 1. Sin[x])

In[•]:= FullSimplify[Chop[LHSCheck]]
Out[•]=
2.77556 × 10^-17 + 0.09 x + 2. Cos[x] - 0.25 Sin[x]

In[•]:= Chop[FullSimplify[Chop[LHSCheck]]]
Out[•]=
0.09 x + 2. Cos[x] - 0.25 Sin[x]

In[•]:= Chop[FullSimplify[Chop[LHSCheck]]] == RHS
Out[•]=
True
```

- Excellent! The same as the right hand side.
- **Step 3.** Then, the general solution to the initial nonhomogeneous ODE is:

```
In[•]:= ygeneral[x_] = yh[x] + ypSoln[x]
Out[•]=
-0.32 + c1 ℯ^(-1.5 x) + c2 ℯ^(-0.5 x) + 0.12 x + 1. Sin[x]
```

- **Step 4.** Find the particular solution using the initial conditions: $y(0) = 2$, $y'(0) = -0.43$.
- Find the derivative $y' = \frac{dy}{dx}$.

```
In[•]:= dygeneral[x_] = D[ygeneral[x], {x, 1}]
Out[•]=
0.12 - 1.5 c1 ℯ^(-1.5 x) - 0.5 c2 ℯ^(-0.5 x) + 1. Cos[x]

In[•]:= IC1 = ygeneral[0] == 2.78
Out[•]=
-0.32 + 1. c1 + 1. c2 == 2.78

In[•]:= IC2 = dygeneral[0] == -0.43
Out[•]=
1.12 - 1.5 c1 - 0.5 c2 == -0.43

In[•]:= valuesofcofficients = Solve[{IC1, IC2}, {c1, c2}]
Out[•]=
{{c1 → 2.22045 × 10^-16, c2 → 3.1}}
```

- Hence, the particular solution to the given ODE is:

```
In[•]:= yparticular[x_] = ygeneral[x] /. valuesofcofficients[[1]]
Out[•]=
-0.32 + 2.22045 × 10^-16 ℯ^(-1.5 x) + 3.1 ℯ^(-0.5 x) + 0.12 x + 1. Sin[x]
```

```
In[ ]:= yparticular[x_] = Chop[ ygeneral[x] /. valuesofcofficients[[1]]]
Out[ ]=
```

$$-0.32 + 3.1\, e^{-0.5\,x} + 0.12\,x + 1.\,\text{Sin}[x]$$

- **Note:** In doing numerical computations, it is inevitable that you will sometimes end up with results that are less precise than you want. Particularly when you get numerical results that are very close to zero, you may well want to assume that the results should be exactly zero. The function Chop allows you to replace approximate real numbers that are close to zero by the exact integer 0.
- **Chop[expr] = To replace all approximate real numbers in expr with magnitude less than 10^{-10} by 0.**
- **Step 5.** Verify the solution to the ODE:

$$y'' + 2\,y' + 0.75\,y = 2\cos(x) - 0.25\sin(x) + 0.09\,x$$

- with initial conditions $y(0) = 2.78,\ y'(0) = -0.43$.

```
In[ ]:= LHSOp[yparticular, x]
Out[ ]=
```

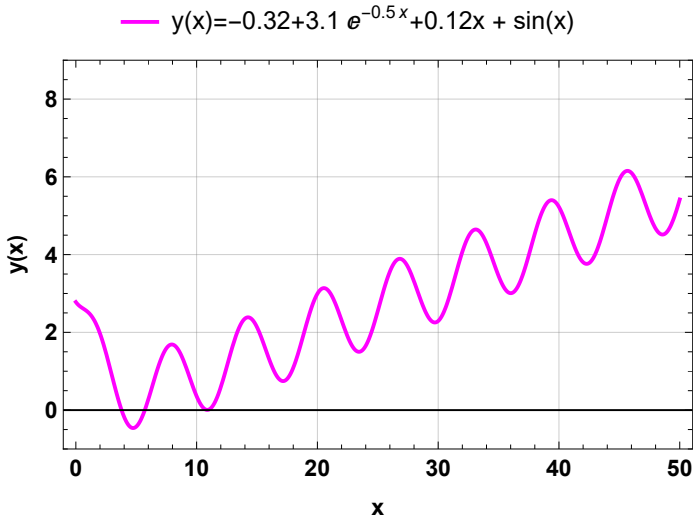
$$0.775\, e^{-0.5\,x} + 2\left(0.12 - 1.55\, e^{-0.5\,x} + 1.\,\text{Cos}[x]\right) - 1.\,\text{Sin}[x] + 0.75\left(-0.32 + 3.1\, e^{-0.5\,x} + 0.12\,x + 1.\,\text{Sin}[x]\right)$$

```
In[ ]:= Chop[FullSimplify[Chop[LHSOp[yparticular, x]]]] == rhsFunc[x]
Out[ ]= True

In[ ]:= yparticular[0]
Out[ ]= 2.78

In[ ]:= D[yparticular[x], {x, 1}] /. {x → 0}
Out[ ]= -0.43
```

- The solution satisfies both the ODE and initial conditions check!
- Let's take a look at the graph of the solution.

In[ ]:=
```
Plot[yparticular[x], {x, 0, 50}, Frame → True,
 PlotStyle → {{Magenta, Thick}, {Black, Thick}},
 PlotLegends → Placed[{"y(x)=-0.32+3.1 ℯ^(-0.5 x)+0.12x + sin(x)"}, Above],
 Frame → True, FrameLabel → {"x", "y(x)"},
 BaseStyle → {FontWeight → "Bold", Black, FontSize → 12}, GridLines → Automatic,
 AxesStyle → Directive[RGBColor[0.`, 0.`, 0.`], AbsoluteThickness[1]],
 Method → {"DefaultBoundaryStyle" → Automatic, "DefaultMeshStyle" → AbsolutePointSize[6],
   "ScalingFunctions" → None}, PlotRange → {-1, 9}]
```

Out[ ]=



◆ **Step 6.** Solve the ODE using a built-in **DSolve** function (Not Required).

In[ ]:=
```
DsolveSoln0 = DSolve[LHSOp[y, x] == rhsFunc[x], y[x], x] (** A general solution **)
```

Out[ ]=

$$\left\{\left\{y[x] \to e^{-1.5\,x}\,c_1 + e^{-0.5\,x}\,c_2 - 0.06\,e^{-1.11022\times10^{-16}\,x}\left(6. - 0.666667\,e^{1.11022\times10^{-16}\,x} - 3.\,x + 1.\,e^{1.11022\times10^{-16}\,x}\,x - \left(16.6667 + 2.77556\times10^{-15}\,i\right)\text{Cos}[x] + \left(16.6667 + 9.25186\times10^{-16}\,i\right)e^{1.11022\times10^{-16}\,x}\,\text{Cos}[x] - \left(25. - 1.85037\times10^{-15}\,i\right)\text{Sin}[x] + \left(8.33333 - 2.77556\times10^{-15}\,i\right)e^{1.11022\times10^{-16}\,x}\,\text{Sin}[x]\right)\right\}\right\}$$

In[ ]:=
```
DsolveSoln1 = DSolve[{LHSOp[y, x] == rhsFunc[x], y[0] == 2.78, y'[0] == -0.43} , y[x], x]
 (** A particular solution **)
```

Out[ ]=

$$\left\{\left\{y[x] \to -0.06\,e^{-2.\,x}\left(\left(5.73615\times10^{-15} - 1.85037\times10^{-15}\,i\right)e^{0.5\,x} - \left(51.6667 - 3.70074\times10^{-15}\,i\right)e^{1.5\,x} + 5.33333\,e^{2.\,x} - 2.\,e^{2.\,x}\,x + \left(3.55271\times10^{-15} - 1.85037\times10^{-15}\,i\right)e^{2.\,x}\,\text{Cos}[x] - \left(16.6667 + 9.25186\times10^{-16}\,i\right)e^{2.\,x}\,\text{Sin}[x]\right)\right\}\right\}$$

In[ ]:=
```
Chop[FullSimplify[DsolveSoln1]]
```

Out[ ]=

$$\{\{y[x] \to -0.32 + 3.1\, e^{-0.5\,x} + 0.12\,x + 1.\,\text{Sin}[x]\}\}$$

In[ ]:=
```
yparticular[x] (** The solution that we obtained through a step-by-step SOP **)
```

Out[ ]=

$$-0.32 + 3.1\, e^{-0.5\,x} + 0.12\,x + 1.\,\text{Sin}[x]$$

Example 3.4. Another example of the Method of Undetermined Coefficients

$$y'' + 2\,y' + 5\,y = 1.25\,\exp(0.5\,x) + 40\,\cos(4\,x) - 55\,\sin(4\,x) \qquad y(0) = 0.2,\ y'(0) = 60.1$$

In[ ]:=
```
ClearAll["Global`*"]
```

In[ ]:=
```
LHSOp[y_, x_] = y''[x] + 2 y'[x] + 5 y[x]
```

Out[ ]=

$$5\,y[x] + 2\,y'[x] + y''[x]$$

In[ ]:=
```
rhsFunc[x_] = 1.25 Exp[0.5 x] + 40 Cos[4 x] - 55 Sin[4 x]
```

Out[ ]=

$$1.25\, e^{0.5\,x} + 40\,\text{Cos}[4\,x] - 55\,\text{Sin}[4\,x]$$

- **Step 1.** Solve the corresponding homogeneous ODE to obtain the general solution o $y_h(x)$.
- To do so, let's solve the characteristic equation.

In[ ]:=
```
a = 2; b = 5;
roots = Solve[λ^2 + a * λ + b == 0, λ]
```

Out[ ]=

$$\{\{\lambda \to -1 - 2\,i\},\ \{\lambda \to -1 + 2\,i\}\}$$

- We got two complex roots, so we proceed with **Case III.**

In this case, the roots of the characteristic equation are complex numbers that give the complex solutions of the ODE. However, it can be shown that we can obtain a basis of real solutions:

$$y_1 = e^{-ax/2}\cos\omega\,x \quad \text{and} \quad y_2 = e^{-ax/2}\sin\omega\,x \quad \text{where} \quad \omega = \sqrt{b - \frac{a^2}{4}}$$

In[ ]:=
```
ω = Sqrt[b - a^2 / 4]
```

Out[ ]=

$$2$$

```
In[•]:= ClearAll[A, B];
        yh[x_] := (A * Cos[ω * x] + B * Sin[ω * x] ) * Exp[(-a / 2) * x];
        yh[x]
Out[•]=
        ℮^-x (A Cos[2 x] + B Sin[2 x])
```

- **Step 2.** Applying the **method of undetermined coefficients**, find a solution $y_p(x)$.
- Since the $r(x)$ term is the sum of several functions:

$$r(x) = 1.25\exp(0.5\,x) + 40\cos(4\,x) - 55\sin(4\,x)$$

- The corresponding $y_p(x)$ choice (from the Undetermined Coefficients Table) is $y_p = C\exp(0.5\,x) + K\cos(4\,x) + M\sin(4\,x)$ (based on the **Sum Rule**).
- C, K, M are coefficients to be determined. Don't also forget to check with the modification rule.

```
In[•]:= yp[x_] = M0 * Exp[0.5 x] + M1 * Cos[4 x] + M2 * Sin[4 x]
Out[•]=
        ℮^(0.5 x) M0 + M1 Cos[4 x] + M2 Sin[4 x]
```

- Let's plug the assumed solution $y_p(x)$ into the LHS.

```
In[•]:= LHS = LHSOp[yp, x]
Out[•]=
        0.25 ℮^(0.5 x) M0 - 16 M1 Cos[4 x] - 16 M2 Sin[4 x] +
          2 (0.5 ℮^(0.5 x) M0 + 4 M2 Cos[4 x] - 4 M1 Sin[4 x]) + 5 (℮^(0.5 x) M0 + M1 Cos[4 x] + M2 Sin[4 x])

In[•]:= RHS = rhsFunc[x]
Out[•]=
        1.25 ℮^(0.5 x) + 40 Cos[4 x] - 55 Sin[4 x]

In[•]:= LHS - RHS
Out[•]=
        -1.25 ℮^(0.5 x) + 0.25 ℮^(0.5 x) M0 - 40 Cos[4 x] - 16 M1 Cos[4 x] + 55 Sin[4 x] - 16 M2 Sin[4 x] +
          2 (0.5 ℮^(0.5 x) M0 + 4 M2 Cos[4 x] - 4 M1 Sin[4 x]) + 5 (℮^(0.5 x) M0 + M1 Cos[4 x] + M2 Sin[4 x])
```

- Next, equate coefficients of x and x^2 because the coefficient of each power of x must be the same on both sides. Hence, LHS-RHS must be zero for all x.

```
In[•]:= Q = Collect[(LHS - RHS), {Exp[0.5 x], Cos[4 x], Sin[4 x]}]
Out[•]=
        ℮^(0.5 x) (-1.25 + 6.25 M0) + (-40 - 11 M1 + 8 M2) Cos[4 x] + (55 - 8 M1 - 11 M2) Sin[4 x]
```

- The condition that all coefficients must be zero gives us 3 equations in 3 unknowns, which we can solve using **Solve**[ ].

```
In[•]:= eqn0 = -1.25` + 6.25` M0 == 0;
        eqn1 = -40 - 11 M1 + 8 M2 == 0;
        eqn2 = 55 - 8 M1 - 11 M2 == 0;

In[•]:= coeffSoln = Solve[{eqn0, eqn1, eqn2}, {M0, M1, M2}]
Out[•]=
        {{M0 → 0.2, M1 → 0, M2 → 5}}
```

- Let's substitute the coefficients to the solution.

```
In[•]:= ypSoln[x_] = yp[x] /. coeffSoln[[1]]
Out[•]=
        0.2 ℯ^(0.5 x) + 5 Sin[4 x]
```

- Check the solution.

```
In[•]:= LHSCheck = LHSOp[ypSoln, x]
Out[•]=
        0.05 ℯ^(0.5 x) + 2 (0.1 ℯ^(0.5 x) + 20 Cos[4 x]) - 80 Sin[4 x] + 5 (0.2 ℯ^(0.5 x) + 5 Sin[4 x])

In[•]:= FullSimplify[Chop[LHSCheck]]
Out[•]=
        1.25 ℯ^(0.5 x) + 40. Cos[4 x] - 55. Sin[4 x]

In[•]:= FullSimplify[Chop[LHSCheck]] == RHS
Out[•]=
        True
```

- Great! The same as the right hand side.
- **Step 3.** Then, the general solution to the initial nonhomogeneous ODE is:

```
In[•]:= ygeneral[x_] = yh[x] + ypSoln[x]
Out[•]=
        0.2 ℯ^(0.5 x) + ℯ^(-x) (A Cos[2 x] + B Sin[2 x]) + 5 Sin[4 x]
```

- **Step 4.** Find the particular solution using the initial conditions: $y(0) = 0.2,\ y'(0) = 60.1$.
- Find the derivative $y' = \frac{dy}{dx}$.

```
In[•]:= dygeneral[x_] = D[ygeneral[x], {x, 1}]
Out[•]=
        0.1 ℯ^(0.5 x) + 20 Cos[4 x] + ℯ^(-x) (2 B Cos[2 x] - 2 A Sin[2 x]) - ℯ^(-x) (A Cos[2 x] + B Sin[2 x])

In[•]:= IC1 = ygeneral[0] == 0.2
Out[•]=
        0.2 + A == 0.2

In[•]:= IC2 = dygeneral[0] == 60.1
Out[•]=
        20.1 - A + 2 B == 60.1
```

```
In[ ]:= valuesofcofficients = Solve[{IC1, IC2}, {A, B}]
Out[ ]=
{{A → 0., B → 20.}}
```

- Hence, the particular solution to the given ODE is:

```
In[ ]:= yparticular[x_] = ygeneral[x] /. valuesofcofficients[[1]]
Out[ ]=
0.2 e^(0.5 x) + e^(-x) (0. + 20. Sin[2 x]) + 5 Sin[4 x]

In[ ]:= yparticular[x_] = Chop[ygeneral[x] /. valuesofcofficients[[1]]]
Out[ ]=
0.2 e^(0.5 x) + 20. e^(-x) Sin[2 x] + 5 Sin[4 x]
```

- **Step 5.** Verify the solution to the ODE:

$$y'' + 2y' + 5y = 1.25\exp(0.5x) + 40\cos(4x) - 55\sin(4x)$$

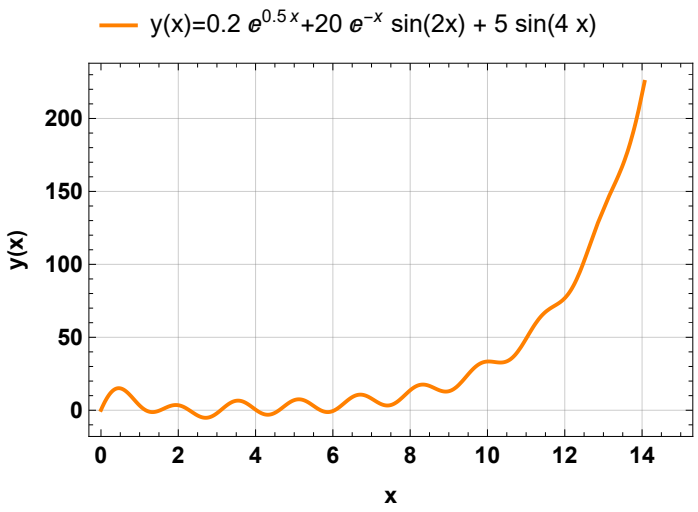
- with initial conditions: $y(0) = 0.2,\ y'(0) = 60.1$.

```
In[ ]:= LHSOp[yparticular, x]
Out[ ]=
0.05 e^(0.5 x) - 80. e^(-x) Cos[2 x] - 60. e^(-x) Sin[2 x] +
 2 (0.1 e^(0.5 x) + 40. e^(-x) Cos[2 x] + 20 Cos[4 x] - 20. e^(-x) Sin[2 x]) -
 80 Sin[4 x] + 5 (0.2 e^(0.5 x) + 20. e^(-x) Sin[2 x] + 5 Sin[4 x])

In[ ]:= FullSimplify[Chop[LHSOp[yparticular, x]]] == rhsFunc[x]
Out[ ]=
True

In[ ]:= yparticular[0]
Out[ ]=
0.2

In[ ]:= D[yparticular[x], {x, 1}] /. {x → 0}
Out[ ]=
60.1
```

- The solution satisfies both the ODE and initial conditions check!
- Let's take a look at the graph of the solution.

```
In[ ]:= Plot[yparticular[x], {x, 0, 15}, Frame → True,
 PlotStyle → {{Orange, Thick}, {Black, Thick}}, Frame → True, FrameLabel → {"x", "y(x)"},
 PlotLegends → Placed[{"y(x)=0.2 ℯ^(0.5 x)+20 ℯ^-x sin(2x) + 5 sin(4 x)"}, Above],
 BaseStyle → {FontWeight → "Bold", Black, FontSize → 12}, GridLines → Automatic,
 AxesStyle → Directive[RGBColor[0.`, 0.`, 0.`], AbsoluteThickness[1]],
 Method → {"DefaultBoundaryStyle" → Automatic,
   "DefaultMeshStyle" → AbsolutePointSize[6], "ScalingFunctions" → None}]
```

Out[ ]=



◆ **Step 6.** Solve the ODE using a built-in **DSolve** function (Not Required).

```
In[ ]:= DsolveSoln0 = DSolve[LHSOp[y, x] == rhsFunc[x], y[x], x] (** A general solution **)
```

Out[ ]=

$$\left\{\left\{y[x] \to e^{(-1.+2.\,i)\,x}\,C_1 + e^{(-1.-2.\,i)\,x}\,C_2 + (0.1 - 0.075\,i)\,e^{-2.\,x}\left((1.28 + 0.96\,i)\,e^{2.5\,x} - \left(2.84217\times 10^{-15} + 2.13163\times 10^{-15}\,i\right)e^{2.\,x}\,\text{Cos}[4.\,x] + (32. + 24.\,i)\,e^{2.\,x}\,\text{Sin}[4.\,x]\right)\right\}\right\}$$

```
In[ ]:= DsolveSoln1 = DSolve[{LHSOp[y, x] == rhsFunc[x], y[0] == 0.2, y'[0] == 60.1} , y[x], x]
 (** A particular solution **)
```

Out[ ]=

$$\left\{\left\{y[x] \to (10. + 0.\,i)\,e^{(-4.-2.\,i)\,x}\left((0. - 1.\,i)\,e^{(3.+4.\,i)\,x} + \left(1.70974\times 10^{-16} + 1.\,i\right)e^{3.\,x} + \left(0.02 + 4.16334\times 10^{-18}\,i\right)e^{(4.5+2.\,i)\,x} - \left(4.44089\times 10^{-17} + 0.\,i\right)e^{(4.+2.\,i)\,x}\,\text{Cos}[4.\,x] + \left(0.5 + 4.44089\times 10^{-17}\,i\right)e^{(4.+2.\,i)\,x}\,\text{Sin}[4.\,x]\right)\right\}\right\}$$

```
In[ ]:= FullSimplify[Chop[DsolveSoln1[[1]]]]
```

Out[ ]=

$$\left\{y[x] \to (0. + 10.\,i)\,e^{(-1.-2.\,i)\,x} - (0. + 10.\,i)\,e^{(-1.+2.\,i)\,x} + 0.2\,e^{0.5\,x} + 5.\,\text{Sin}[4.\,x]\right\}$$

```
In[ ]:= yfromDsolve = y[x] /. FullSimplify[Chop[DsolveSoln1[[1]]]] (**A result from DSolve **)
```

Out[ ]=

$$(0. + 10.\,i)\,e^{(-1.-2.\,i)\,x} - (0. + 10.\,i)\,e^{(-1.+2.\,i)\,x} + 0.2\,e^{0.5\,x} + 5.\,\text{Sin}[4.\,x]$$

In[ ]:=
```
yparticular[x] (** The solution that we obtained through a step-by-step SOP **)
```

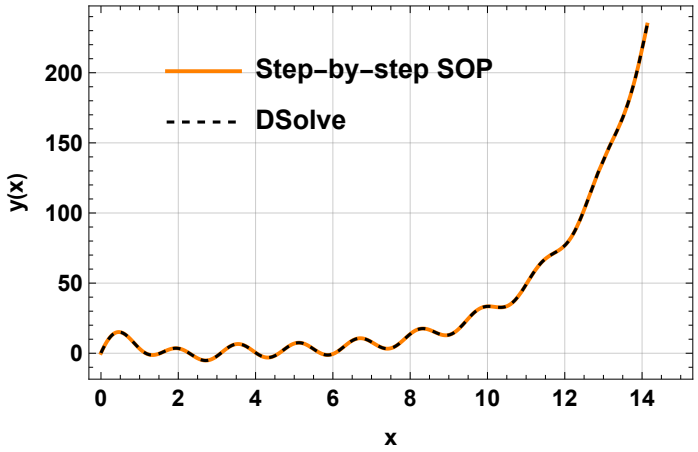
Out[ ]=

$$0.2\, e^{0.5\,x} + 20.\, e^{-x} \sin[2\,x] + 5 \sin[4\,x]$$

- Let' compare the solution that we obtained through a step-by-step **SOP** (standard opera ing procedures) to that from **DSolve** by plotting them together.

In[ ]:=
```
Plot[{yparticular[x], yfromDsolve}, {x, 0, 15}, Frame → True,
 PlotStyle → {{Orange, Thick}, {Black, Dashed}}, Frame → True, FrameLabel → {"x", "y(x)"},
 BaseStyle → {FontWeight → "Bold", Black, FontSize → 12}, GridLines → Automatic,
 AxesStyle → Directive[RGBColor[0.`, 0.`, 0.`], AbsoluteThickness[1]],
 Method → {"DefaultBoundaryStyle" → Automatic,
   "DefaultMeshStyle" → AbsolutePointSize[6], "ScalingFunctions" → None},
 PlotLegends → Placed[{"Step-by-step SOP", "DSolve"}, {0.4, 0.75}]]
```

Out[ ]=



- Exactly the same! Well done.

Summary

After completing this chapter, you should be able to

- solve 2nd-order linear non-homogeneous ODEs step-by-step by the method of undetermined coefficients using Wolfram Mathematica.
- develop SOPs for the method of undetermined coefficients.
- develop the habit of always checking your solutions for quality assurance.
- develop your attention-to-detail skills in solving problems.

Week 4: Second-Order ODEs (Part 3)

Forced Oscillations & Resonance

Table of Contents

Commands list

- Collect[*expr, x*]
- *expr*[[*i*]] or Part[*expr, i*]
- Solve[*expr, vars*]
- Plot[*f, {x, x_min, x_max}*]

Modeling: Forced Oscillations

The oscillations in the presence of the external force is described by the **Nonhomogeneous Linear Second Order** ODE as follows:

$$m\,y''(t) + c\,y'(t) + k\,y(t) = F_0 \cos \omega t$$

here $r(t) = F_0 \cos \omega t$ is called a **driving force.**

m is the mass of the object that undergoes the oscillations.
c is the damping constant.
k is the spring constant.
$y(t)$ is displacement as a function of time, t.

Solve the Nonhomogeneous ODE

In[ ]:=
```
ClearAll["Global`*"]
```

- To start with, let's write the governing equation in the standard form:

$$y''(t) + \frac{c}{m} y'(t) + \frac{k}{m} y(t) = \frac{F_0}{m} \cos \omega t$$

- Define the equations for the LHS and RHS.

```
In[•]:= LHSOp[y_, x_] = y''[t] + (c / m) * y'[t] + (k / m) * y[t]
```

Out[•]=

$$\frac{k\,y[t]}{m} + \frac{c\,y'[t]}{m} + y''[t]$$

```
In[•]:= rhsFunc[x_] = (F0 / m) * Cos[ω * t]
```

Out[•]=

$$\frac{F0\,\text{Cos}[t\,\omega]}{m}$$

- **Step 1.** Solve the corresponding homogeneous ODE to obtain the general solution o $y_h(t)$.
- To do so, let's solve the characteristic equation.

```
In[•]:= a = c / m; b = k / m;
        roots = Solve[λ^2 + a * λ + b == 0, λ]
```

Out[•]=

$$\left\{\left\{\lambda \to \frac{1}{2}\left(-\frac{c}{m} - \frac{\sqrt{c^2 - 4\,k\,m}}{m}\right)\right\}, \left\{\lambda \to \frac{1}{2}\left(-\frac{c}{m} + \frac{\sqrt{c^2 - 4\,k\,m}}{m}\right)\right\}\right\}$$

```
In[•]:= Discriminant[λ^2 + a * λ + b, λ]
```

Out[•]=

$$\frac{c^2 - 4\,k\,m}{m^2}$$

Case	Roots of (2)	Basis of (1)	General Solution of (1)
I	Distinct real $\lambda_1,\ \lambda_1$	$e^{\lambda_1 x},\ e^{\lambda_2 x}$	$y = c_1 e^{\lambda_1 x} + c_2 e^{\lambda_2 x}$
II	Real double root $\lambda = -\frac{1}{2} a$	$e^{-ax/2},\ x e^{-ax/2}$	$y = (c_1 + c_2 x) e^{-ax/2}$
III	Complex conjugate $\lambda_1 = -\frac{1}{2} a + i\omega,$ $\lambda_2 = -\frac{1}{2} a - i\omega$	$e^{-ax/2} \cos \omega x$ $e^{-ax/2} \sin \omega x$	$y = e^{-ax/2}$ (A *cos* ωx + B *sin* ωx)

- **Step 2.** Applying the **method of undetermined coefficients**, find a solution $y_p(t)$.

Term in r (x)	Choice for $y_p(x)$
$ke^{\gamma x}$	$Ce^{\gamma x}$
$kx^n \quad (n = 0,1,\cdots)$	$K_n x^n + K_{n-1} x^{n-1} + \cdots + K_1 x + K_0$
$k \cos \omega x$ $k \sin \omega x$	$K \cos \omega x + M \sin \omega x$
$ke^{\alpha x} \cos \omega x$ $ke^{\alpha x} \sin \omega x$	$e^{\alpha x} (K \cos \omega x + M \sin \omega x)$

- Since the $r(t)$ term is in the form of $k \cos \omega t$, the corresponding $y_p(t)$ choice (second row in the Table) is $y_p = K_1 \cos \omega t + K_2 \sin \omega t$.
- K_1, K_2 are coefficients to be determined.

In[ ]:=
```
yp[t_] = K1 * Cos[ω * t] + K2 * Sin[ω * t]
```

- Let's plug the assumed solution $y_p(t)$ into the LHS.

In[ ]:=
```
yp[t_] = K1 * Cos[ω * t] + K2 * Sin[ω * t]
```

Out[ ]=
```
K1 Cos[t ω] + K2 Sin[t ω]
```

In[ ]:=
```
LHS = LHSOp[yp, t]
```

Out[ ]=
```
-K1 ω^2 Cos[t ω] - K2 ω^2 Sin[t ω] + (k (K1 Cos[t ω] + K2 Sin[t ω]))/m + (c (K2 ω Cos[t ω] - K1 ω Sin[t ω]))/m
```

```
In[ ]:= RHS = rhsFunc[t]
```

Out[ ]=

$$\frac{F0\,\text{Cos}[t\,\omega]}{m}$$

```
In[ ]:= LHS - RHS
```

Out[ ]=

$$-\frac{F0\,\text{Cos}[t\,\omega]}{m} - K1\,\omega^2\,\text{Cos}[t\,\omega] - K2\,\omega^2\,\text{Sin}[t\,\omega] + \frac{k\,(K1\,\text{Cos}[t\,\omega] + K2\,\text{Sin}[t\,\omega])}{m} + \frac{c\,(K2\,\omega\,\text{Cos}[t\,\omega] - K1\,\omega\,\text{Sin}[t\,\omega])}{m}$$

- Next, gather coefficients of **cos** $\omega\,t$ and **sin** $\omega\,t$. Notice that we are working with LHS-RHS at this point.
- LHS-RHS must be zero for all t, which means the coefficients of **cos** $\omega\,t$ and **sin** $\omega\,t$ must be zero independently.

```
In[ ]:= Q = Collect[(LHS - RHS), {Cos[ω * t], Sin[ω * t]}]
```

Out[ ]=

$$\left(-\frac{F0}{m} + \frac{k\,K1}{m} + \frac{c\,K2\,\omega}{m} - K1\,\omega^2\right)\text{Cos}[t\,\omega] + \left(\frac{k\,K2}{m} - \frac{c\,K1\,\omega}{m} - K2\,\omega^2\right)\text{Sin}[t\,\omega]$$

- The conditions that all coefficients must be 0 gives us two equations with two unknowns, which we can solve using **Solve**[ ].

```
In[ ]:= eqn1 = (-F0/m + k K1/m + c K2 ω/m - K1 ω^2) == 0 (**coefficient of Cos[ω*t] **);
        eqn2 = (k K2/m - c K1 ω/m - K2 ω^2) == 0 (**coefficient of Sin[ω*t] **);
```

```
In[ ]:= coeffSoln = Solve[{eqn1, eqn2}, {K1, K2}]
```

Out[ ]=

$$\left\{\left\{K1 \to \frac{F0\,(k - m\,\omega^2)}{k^2 + c^2\,\omega^2 - 2\,k\,m\,\omega^2 + m^2\,\omega^4},\ K2 \to \frac{c\,F0\,\omega}{k^2 + c^2\,\omega^2 - 2\,k\,m\,\omega^2 + m^2\,\omega^4}\right\}\right\}$$

```
In[ ]:= FullSimplify[coeffSoln /. {k → (m * ω0^2)}]
```

Out[ ]=

$$\left\{\left\{K1 \to \frac{F0\,m\,(-\omega^2 + \omega_0^2)}{\omega^2\,(c^2 + m^2\,\omega^2) + m^2\,\omega_0^2\,(-2\,\omega^2 + \omega_0^2)},\ K2 \to \frac{c\,F0\,\omega}{\omega^2\,(c^2 + m^2\,\omega^2) + m^2\,\omega_0^2\,(-2\,\omega^2 + \omega_0^2)}\right\}\right\}$$

- Let's substitute the coefficients to the solution.

```
In[ ]:= ypSoln[t_] = yp[t] /. coeffSoln[[1]]
```

Out[ ]=

$$\frac{F0\,(k - m\,\omega^2)\,\text{Cos}[t\,\omega]}{k^2 + c^2\,\omega^2 - 2\,k\,m\,\omega^2 + m^2\,\omega^4} + \frac{c\,F0\,\omega\,\text{Sin}[t\,\omega]}{k^2 + c^2\,\omega^2 - 2\,k\,m\,\omega^2 + m^2\,\omega^4}$$

- Check the solution.

In[ ]:=
```
LHSCheck = LHSOp[ypSoln, t]
```

Out[ ]=

$$-\frac{F0\,\omega^2\,\left(k-m\,\omega^2\right)\,\text{Cos}[t\,\omega]}{k^2+c^2\,\omega^2-2\,k\,m\,\omega^2+m^2\,\omega^4}-\frac{c\,F0\,\omega^3\,\text{Sin}[t\,\omega]}{k^2+c^2\,\omega^2-2\,k\,m\,\omega^2+m^2\,\omega^4}+\frac{k\left(\frac{F0\,\left(k-m\,\omega^2\right)\,\text{Cos}[t\,\omega]}{k^2+c^2\,\omega^2-2\,k\,m\,\omega^2+m^2\,\omega^4}+\frac{c\,F0\,\omega\,\text{Sin}[t\,\omega]}{k^2+c^2\,\omega^2-2\,k\,m\,\omega^2+m^2\,\omega^4}\right)}{m}+\frac{c\left(\frac{c\,F0\,\omega^2\,\text{Cos}[t\,\omega]}{k^2+c^2\,\omega^2-2\,k\,m\,\omega^2+m^2\,\omega^4}-\frac{F0\,\omega\,\left(k-m\,\omega^2\right)\,\text{Sin}[t\,\omega]}{k^2+c^2\,\omega^2-2\,k\,m\,\omega^2+m^2\,\omega^4}\right)}{m}$$

In[ ]:=
```
FullSimplify[LHSCheck - RHS ]
```

Out[ ]=
```
0
```

In[ ]:=
```
FullSimplify[LHSCheck] == RHS
```

Out[ ]=
```
True
```

- Great! The same as the right hand side.
- **Step 3.** Then, the general solution to the initial nonhomogeneous ODE is:

In[ ]:=
```
ygeneral[t_] = yh[t] + ypSoln[t]
```

Out[ ]=

$$\frac{F0\left(k-m\,\omega^2\right)\,Cos[t\,\omega]}{k^2+c^2\,\omega^2-2\,k\,m\,\omega^2+m^2\,\omega^4}+\frac{c\,F0\,\omega\,Sin[t\,\omega]}{k^2+c^2\,\omega^2-2\,k\,m\,\omega^2+m^2\,\omega^4}+yh[t]$$

- **yh[t]** is the solution to the corresponding homogeneous equation based on three cases depending on the discriminant sign.

Find the maximum amplitude of Damped Forced Oscillations

From the previous chapter, we know that the solution $y_p(t)$ for the nonhomogeneous ODE is as follows:

$$y_p = F_0\,\frac{m\left(\omega_0^2-\omega^2\right)}{m^2\left(\omega_0^2-\omega^2\right)+\omega^2\,c^2}\,\cos(\omega\,t)+F_0\,\frac{\omega\,c}{m^2\left(\omega_0^2-\omega^2\right)+\omega^2\,c^2}\,\sin(\omega\,t)$$

After a sufficiently long time the output of a damped vibrating system under a purely sinusoidal driving force will practically be a harmonic oscillation whose frequency is that of the input. It is called a **Steady State solution**, when $y(t) = y_p(t)$.

$$y_p(t) = a\cos(\omega t) + b\sin(\omega t) = C^* \cos(\omega t - \eta)$$

Amplitude

$$C^* = \sqrt{a^2 + b^2} = \frac{F_0}{\sqrt{m^2(\omega_0^2 - \omega^2) + \omega^2 c^2}}$$

Phase angle η

$$\tan\eta = \frac{b}{a} = \frac{\omega c}{m(\omega_0^2 - \omega^2)}$$

Example 4.1. Amplitude of the Steady State solution. Practical Resonance

The amplitude $C^*(\omega)$ has a maximum at a certain ω value. Find its location, then its size.

```
In[•]:= ClearAll["Global`*"]
```

- Define $C^*(\omega)$ as a function:

```
In[•]:= Ampl[ω_] := F0 / Sqrt[m^2 (w0^2 - ω^2)^2 + ω^2 c^2];
```

- Find its maximum by taking the first derivative.

```
In[•]:= Solve[D[Ampl[ω], ω] == 0, ω]  (** 1st derivative=0, solve for ω **)
```

```
Out[•]= {{ω → 0}, {ω → -Sqrt[-c^2 + 2 m^2 w0^2] / (Sqrt[2] m)}, {ω → Sqrt[-c^2 + 2 m^2 w0^2] / (Sqrt[2] m)}}
```

- Then $C^*(\omega_{max})$ is equal to:

```
In[•]:= C0 = FullSimplify[Ampl[Sqrt[-c^2 + 2 m^2 w0^2] / (Sqrt[2] m)]]  (** C*(ωmax) **)
```

```
Out[•]= 2 F0 / Sqrt[-c^4/m^2 + 4 c^2 w0^2]
```

- $(\omega_{max})^2$ is equal to:

In[ ]:=
```
FullSimplify[(Sqrt[-c^2 + 2 m^2 w0^2]/(Sqrt[2] m))^2] (** ω_max^2 **)
```

Out[ ]=

$$-\frac{c^2}{2\,m^2}+w0^2$$

- Let's plot the **amplification** $\frac{C^*}{F_0}$ as a function of ω to see how the amplitude changes when the damping terms vary. The data about the mass, spring constant, driving force is randomly assigned.

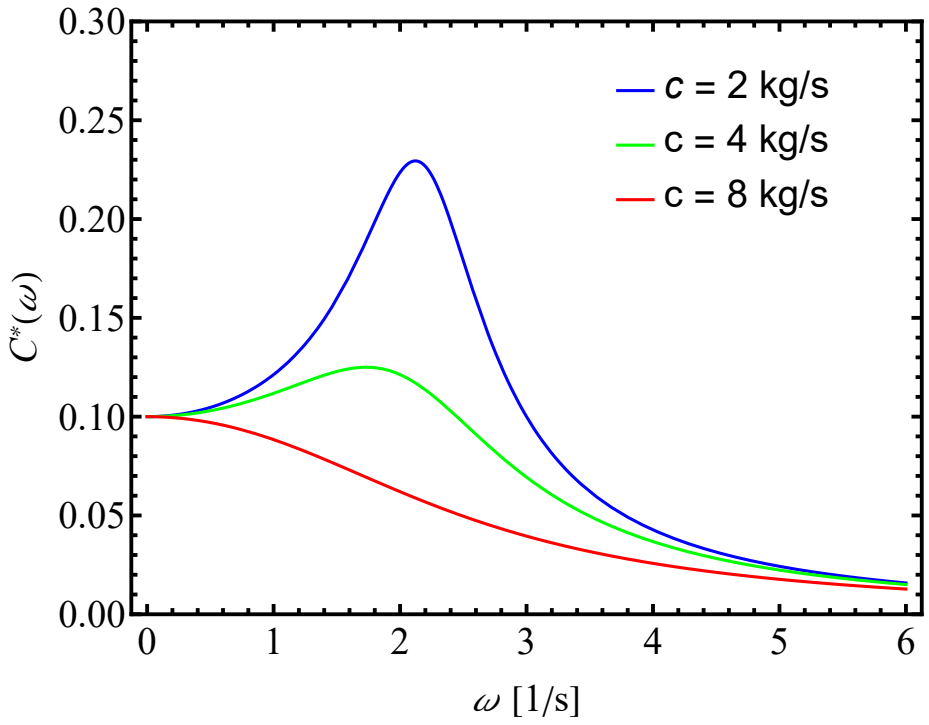
In[ ]:=
```
C0 = Ampl[ω]/F0 /. {F0 → 10, m → 2, w0 → Sqrt[5]}
```

Out[ ]=

$$\frac{1}{\sqrt{c^2\,\omega^2+4\,\left(5-\omega^2\right)^2}}$$

In[ ]:=
```
fC0[c_, w_] := 1/Sqrt[c^2 ω^2 + 4 (5 - ω^2)^2];

fontsize = 20;
fig01 = Plot[{fC0[2, ω], fC0[4, ω], fC0[8, ω]}, {ω, 0, 6},
  PlotRange → {0, 0.3}, PlotStyle → {Blue, Green, Red}, Background → White,
  BaseStyle → {FontFamily → "Times New Roman", fontsize},
  Frame → True,
  FrameLabel → {"ω [1/s]", "C*(ω)"},
  FrameStyle → Directive[Black, Thick], AxesOrigin → {0, 0},
  PlotLegends → Placed[LineLegend[Automatic, {Text[Style["c = 2 kg/s", fontsize]],
      Text[Style["c = 4 kg/s", fontsize]], Text[Style["c = 8 kg/s", fontsize]]},
    Spacings → 0.2, LegendLayout → {"Column", 1}], {0.75, 0.8}],
  ImageSize → 480, AspectRatio → 3 / 4]
```

Out[ ]=



In[ ]:=
```
Export["fig01.pdf", fig01,
  "AllowRasterization" → True, ImageSize → 480, ImageResolution → 600] ;
```

In[ ]:=
```
SystemOpen["fig01.pdf"]
```

- From the graph, we can conclude that the **biggest practical resonance** happens w the **damping term** is the **smallest**.
- Please note that this is the figure of the **publication quality**.

Summary

After completing this chapter, you should be able to

- develop standard operating procedures to solve second-order linear non-homogeneous ODEs step-by-step using Wolfram Mathematica.
- model simple physical situations encountered in engineering using differential equations.
- learn and use information, tools, and technology to solve engineering math problems.
- analyze results graphically and create figures of publication quality.

Week 5: Laplace Transforms

Basics of Laplace Transforms

Table of Contents

Commands list

- LaplaceTransform[*f*[*t*],*t*,*s*]
- InverseLaplaceTransform[*F*[s],*s*,*t*]
- Integrate[*f*, *x*]
- Limit[*f* , $x \to x^*$]
- HeavisideTheta[*x*]
- UnitStep[*x*]
- DiracDelta[*x*]

Basics of Laplace Transforms

Laplace tranforms are essential tools in solving the engineering problems since they they make solving linear ODEs , IVPs, as well as systems of linear ODEs, much easier.

If $f(t)$ is a function defined for all $t \geqq 0$, its **Laplace transform** is the integral of $f(t)$ times e^{-st} from $t = 0$ to ∞. It is a function of s, say, $F(s)$, and is denoted by $\mathcal{L}(f)$; thus

$$F(s) = \mathcal{L}(f) = \int_0^\infty e^{-st} f(t)\, dt$$

Not only is the result $F(s)$ called the Laplace transform, but the operation just described, which yields $F(s)$ from a given $f(t)$, is also called the **Laplace transform**. It is an "**integral transform**" with "**kernel**": $k(s, t) = e^{-st}$.

$$F(s) = \int_0^\infty k(s, t) f(t)\, dt$$

Built-in Functions in Wolfram Mathematica

Laplace transforms are typically used to transform **differential and part equations** to **algebraic equations**, solve and then inverse **transform back** to a solution.

Laplace transforms are also extensively used in **control theory** and **signal processing** as a way to represent and manipulate linear systems in the form of transfer functions and transfer matrices. The Laplace transform and its inverse are then a way to transform **between domain and frequency domain**.

LaplaceTransform[f[t], t, s] gives the symbolic Laplace transform of *f[t]* in the variable *t* and returns a transform *F[s]* in the variable *s*.

InverseLaplaceTransform[F[s], s, t] gives the symbolic inverse Laplace transform o *f[t]* in the variable *s* and returns a transform *F[s]* in the variable *t*.

In[1]:=
```
? LaplaceTransform
```

Out[1]=

Symbol

LaplaceTransform[$f[t]$, t, s] gives the symbolic Laplace transform of $f[t]$ in the variable t and returns a transform $F[s]$ in the variable s.

LaplaceTransform[$f[t]$, t, $\hat{s}$] gives the numeric Laplace transform at the numerical value $\hat{s}$.

LaplaceTransform[$f[t_1, \ldots, t_n]$, $\{t_1, \ldots, t_n\}$, $\{s_1, \ldots, s_n\}$]

gives the multidimensional Laplace transform of $f[t_1, \ldots, t_n]$.

In[2]:=
```
? InverseLaplaceTransform
```

Out[2]=

Symbol

InverseLaplaceTransform[$F[s]$, s, t] gives the symbolic inverse Laplace transform of $F[s]$ in the variable s as $f[t]$ in the variable t.

InverseLaplaceTransform[$F[s]$, s, $\hat{t}$] gives the numeric inverse Laplace transform at the numerical value $\hat{t}$.

InverseLaplaceTransform[$F[s_1, \ldots, s_n]$, $\{s_1, s_2, \ldots\}$, $\{t_1, t_2, \ldots\}$]

gives the multidimensional inverse Laplace transform of $F[s_1, \ldots, s_n]$.

In[3]:=
```
ClearAll["Global`*"]
```

- Define a function ***f(t)*** in the variable ***t.***

In[4]:=
```
f[t_] := Exp[a * t] * Cos[w * t];
```

- Find the **Laplace transform** using the built-in function.

In[5]:=
```
LaplaceTransform[f[t], t, s]
```

Out[5]=
$$\frac{-a+s}{(a-s)^2+w^2}$$

- Define a function ***F(s)*** in the variable ***s.***

In[6]:=
```
F[s_] := (-a + s)/((a - s)^2 + w^2);
```

- Find the **Inverse of the Laplace transform** using the built-in function.

In[7]:=
```
InverseLaplaceTransform[F[s], s, t]
```

Out[7]=
$$e^{at}\,\text{Cos}[tw]$$

Laplace Transform by Integration

In[8]:=
```
ClearAll["Global`*"]
```

- Define a **kernel** function.

In[9]:=
```
kernel[s_, t_] := Exp[-s * t];
```

- Define a function *f(t)* in the variable *t*.

In[10]:=
```
f[t_] := Exp[a * t];
```

- Perform the integration.

In[11]:=
```
Integrate[kernel[s, t] * f[t], {t, 0, +Infinity}]
```

Out[11]=

$$\frac{1}{-a+s} \text{ if } \text{Re}[a] < \text{Re}[s]$$

In[12]:=
```
Integrate[kernel[s, t] * f[t], {t, 0, T}]
```

Out[12]=

$$\frac{-1+e^{(a-s)\,T}}{a-s}$$

- Take a limit of the integral as *T* approaches the infinity.

In[13]:=
```
Limit[(-1 + e^((a-s) T))/(a - s), T → Infinity]
```

Out[13]=

$$\frac{1}{-a+s} \text{ if } a < s$$

In[14]:=
```
Limit[(-1 + e^((a-s) T))/(a - s), T → Infinity, Assumptions → {Re[a] < Re[s]}]
```

Out[14]=

$$\frac{1}{-a+s}$$

Linearity of the Laplace Transform

The Laplace transform is a ***linear operation***; that is, for any functions *f(t)* and *g(t)* whose transforms exist and any constants *a* and *b* the transform of $af(t) + bg(t)$ exists, and

$$\mathcal{L}\{af(t) + bg(t)\} = a\mathcal{L}\{f(t)\} + b\mathcal{L}\{g(t)\}$$

- Let's verify the linearity of the Laplace transform.

In[15]:=
```
ClearAll["Global`*"]
```

- Define the RHS of the above equation. For that, find the Laplace transfroms of $a\mathcal{L}\{f(t)\}$ and $b\mathcal{L}\{g(t)\}$.

In[16]:=
```
RHS1 = LaplaceTransform[c1 * Exp[a * t], t, s]
```

Out[16]=

$$\frac{c1}{-a+s}$$

In[17]:=
```
RHS2 = LaplaceTransform[c2 * Exp[b * t], t, s]
```

Out[17]=

$$\frac{c2}{-b+s}$$

In[18]:=
```
RHS = RHS1 + RHS2
```

Out[18]=

$$\frac{c1}{-a+s}+\frac{c2}{-b+s}$$

- Define the LHS of the above equation.

In[19]:=
```
LHS = LaplaceTransform[c1 * Exp[a * t] + c2 * Exp[b * t], t, s]
```

Out[19]=

$$\frac{c1}{-a+s}+\frac{c2}{-b+s}$$

- Check for equality.

In[20]:=
```
LHS == RHS
```

Out[20]=

```
True
```

Laplace Transform of Derivatives

In[21]:=
```
ClearAll["Global`*"]
```

- Find the Laplace Transform of the **1st** derivative of the function ***f(t).***

In[22]:=
```
LaplaceTransform[f'[t], t, s]
```

Out[22]=

```
-f[0] + s LaplaceTransform[f[t], t, s]
```

- Find the Laplace Transform of the **2nd** derivative of the function ***f(t).***

In[23]:=
```
LaplaceTransform[f''[t], t, s]
```

Out[23]=

```
-s f[0] + s^2 LaplaceTransform[f[t], t, s] - f'[0]
```

Unit Step Function and Dirac's Delta Fuction

Unit Step Function (Heaviside Function)

There are two ways to generate a unit step function in Wolfram Mathematica. You may use either the ***UnitStep[x]*** command or the ***HeavisideTheta[x]*** command.

- **HeavisideTheta[x]** returns 0 or 1 for all real numeric x other than 0. HeavisideTheta can be used in integrals, integral transforms, and differential equations.

In[24]:=
```
? HeavisideTheta
```

Out[24]=

Symbol

HeavisideTheta[x] represents the Heaviside theta function θ(x), equal to 0 for x < 0 and 1 for x > 0.

HeavisideTheta[x_1, x_2, ...] represents the multidimensional

Heaviside theta function, which is 1 only if all of the x_i are positive.

- **UnitStep[x]** represents the unit step function, equal to 0 for $x < 0$ and 1 for $x \geq 0$.

In[25]:=
```
? UnitStep
```

Out[25]=

Symbol

UnitStep[x] represents the unit step function, equal to 0 for x < 0 and 1 for x ≥ 0.

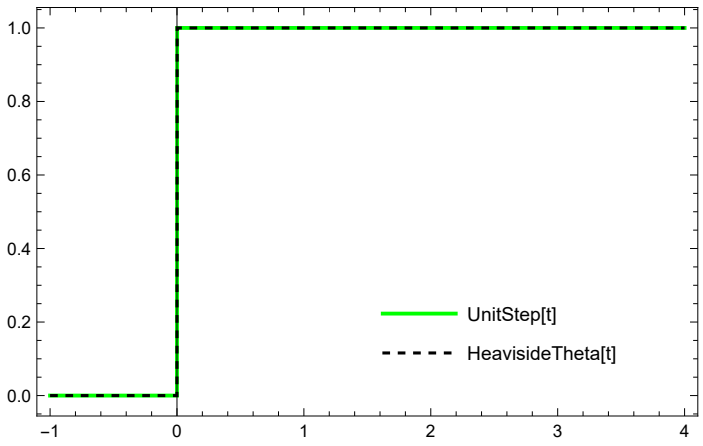
UnitStep[x_1, x_2, ...] represents the multidimensional

unit step function which is 1 only if none of the x_i are negative.

In[26]:=
```
ClearAll["Global`*"]
```

In[27]:=
```
Plot[{UnitStep[t], HeavisideTheta[t]}, {t, -1, 4},
 PlotStyle → {{Green, Thick}, {Black, Dashed}}, Exclusions → None, Frame → True,
 PlotLegends → Placed[{"UnitStep[t]", "HeavisideTheta[t]"}, {0.7, 0.2}],
 Background → White]
```

Out[27]=



What is the Laplace Transform of the Unit Step Function?

$$\mathcal{L}\{u(t-a)\} = \int_0^{+\infty} e^{-st}\, u(t-a)\, dt = \frac{e^{-as}}{s}$$

In[28]:=
```
LaplaceTransform[HeavisideTheta[t - a], t, s]
```

Out[28]=

$$\frac{\text{UnitStep}[-a] + e^{-as}\,\text{UnitStep}[a]}{s}$$

In[29]:=
```
LaplaceTransform[UnitStep[t - a], t, s]
```

Out[29]=

$$\frac{\text{UnitStep}[-a] + e^{-as}\,\text{UnitStep}[a]}{s}$$

In[30]:=
```
FullSimplify[LaplaceTransform[UnitStep[t - a], t, s], a ∈ Reals && a > 0]
```

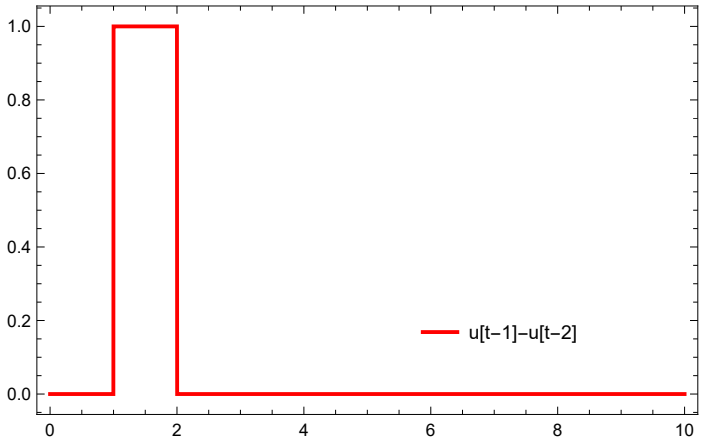
Out[30]=

$$\frac{e^{-as}}{s}$$

- In engineering, the Unit step function is mainly used in the problems that involve **switch on and off, shifts**.

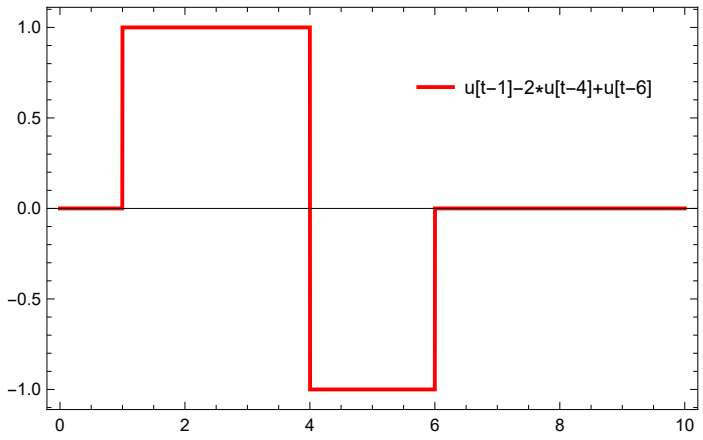
In[ ]:=
```
Plot[UnitStep[t - 1] - UnitStep[t - 2], {t, 0, 10}, PlotStyle → {Red, Thick},
 PlotLegends → Placed[{"u[t-1]-u[t-2]"}, {0.7, 0.2}], Exclusions → None, Frame → True]
```

Out[ ]=



In[32]:=
```
Plot[UnitStep[t - 1] - 2 * UnitStep[t - 4] + UnitStep[t - 6], {t, 0, 10},
 PlotStyle → {Red, Thick}, PlotLegends → Placed[{"u[t-1]-2*u[t-4]+u[t-6]"}, {0.75, 0.8}],
 Exclusions → None, Frame → True]
```

Out[32]=



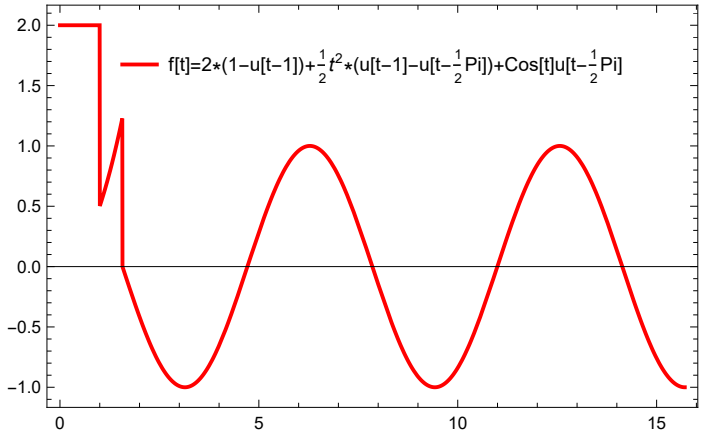
Example 5.1

$$f(t) = \begin{cases} 2 & 0 < t < 1 \\ \frac{1}{2} t^2 & 1 < t < \frac{1}{2}\pi \\ \cos(t) & t > \frac{1}{2}\pi \end{cases}$$

In[33]:=
```
ClearAll["Global`*"]
```

In[34]:=
```
Plot[2 * (1 - UnitStep[t - 1]) + 1/2 t^2 * (UnitStep[t - 1] - UnitStep[t - 1/2 Pi]) +
  Cos[t] UnitStep[t - 1/2 Pi], {t, 0, 5 Pi}, PlotStyle → {Red, Thick},
 PlotLegends → Placed[{"f[t]=2*(1-u[t-1])+1/2 t^2*(u[t-1]-u[t-1/2 Pi])+Cos[t]u[t-1/2 Pi]"},
  {0.5, 0.85}], Exclusions → None, Frame → True]
```

Out[34]=



- Let's define the given *f(t)* function as a function *y[t_]* of the variable *t*.

In[35]:=
```
y[t_] := 2 * (1 - UnitStep[t - 1]) +
  1/2 t^2 * (UnitStep[t - 1] - UnitStep[t - 1/2 Pi]) + Cos[t] UnitStep[t - 1/2 Pi]; y[t]
```

Out[35]=

$$2\,(1-\text{UnitStep}[-1+t])+\frac{1}{2}t^2\left(\text{UnitStep}[-1+t]-\text{UnitStep}\left[-\frac{\pi}{2}+t\right]\right)+\text{Cos}[t]\,\text{UnitStep}\left[-\frac{\pi}{2}+t\right]$$

- Find the Laplace transform of the function.

In[36]:=
```
Y[s_] := LaplaceTransform[y[t], t, s]; Y[s]
```

Out[36]=

$$\frac{1}{2}e^{-s}\left(\frac{2}{s^3}+\frac{2}{s^2}+\frac{1}{s}\right)-\frac{1}{2}e^{-\frac{\pi s}{2}}\left(\frac{2}{s^3}+\frac{\pi}{s^2}+\frac{\pi^2}{4s}\right)+\frac{2}{s}-\frac{2e^{-s}}{s}-\frac{e^{-\frac{\pi s}{2}}}{1+s^2}$$

- Find the Inverse of the Laplace transform of the function.

In[37]:=
```
y2[t_] := InverseLaplaceTransform[Y[s], s, t]; y2[t]
```

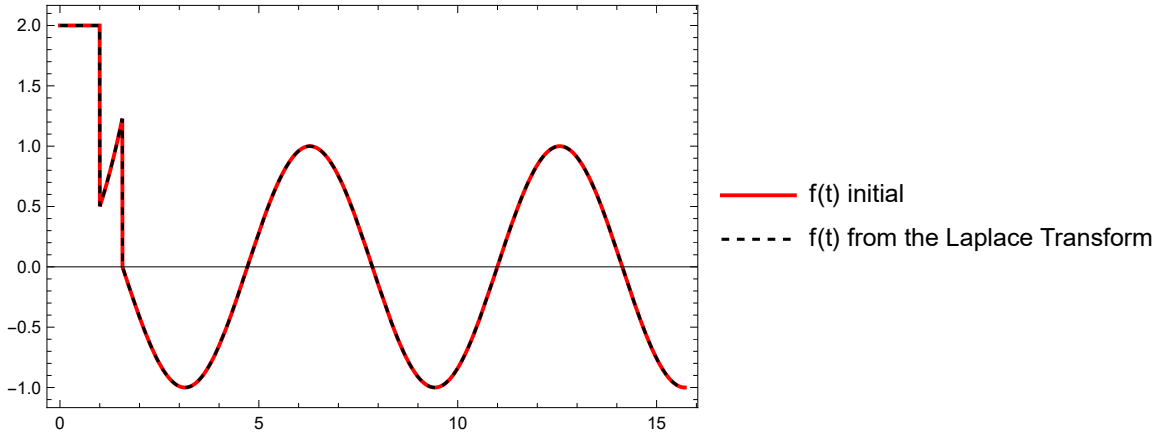
Out[37]=

$$2-\frac{3}{2}\text{HeavisideTheta}[-1+t]+(-1+t)\,\text{HeavisideTheta}[-1+t]+\frac{1}{2}(-1+t)^2\,\text{HeavisideTheta}[-1+t]-\frac{1}{8}\pi^2\,\text{HeavisideTheta}\left[-\frac{\pi}{2}+t\right]-\frac{1}{2}\pi\left(-\frac{\pi}{2}+t\right)\text{HeavisideTheta}\left[-\frac{\pi}{2}+t\right]-\frac{1}{2}\left(-\frac{\pi}{2}+t\right)^2\text{HeavisideTheta}\left[-\frac{\pi}{2}+t\right]+\text{Cos}[t]\,\text{HeavisideTheta}\left[-\frac{\pi}{2}+t\right]$$

- As we can see, the initial function and the expression obtained from the Laplace transform method yield the same result.

In[38]:=
```
Plot[{y[t], y2[t]}, {t, 0, 5 Pi},
 PlotStyle → {{Red, Thick}, {Black, Dashed}}, Exclusions → None, Frame → True,
 PlotLegends → {"f(t) initial", "f(t) from the Laplace Transform"}]
```

Out[38]=



Example 5.2

$$f(t) = \begin{cases} (1+t)^2 & 0 \le t < 1 \\ 1 + t^2 & t \ge 1 \end{cases}$$

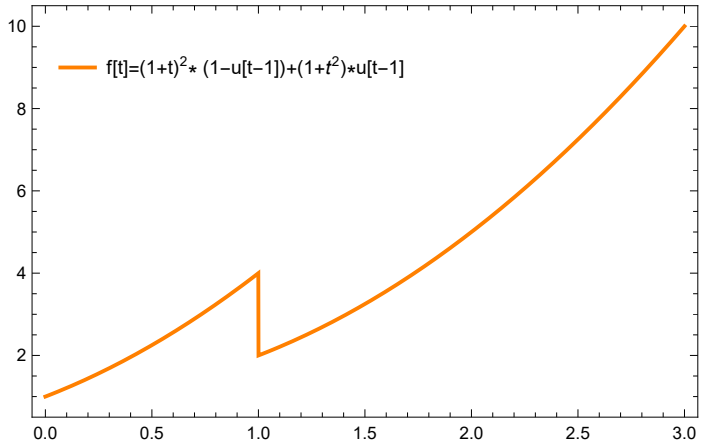
In[39]:=
```
ClearAll["Global`*"]
```

- Let's define the given *f(t)* function as a function of the variable *t*.

In[40]:=
```
f[t_] := (1 + t)^2 * (1 - HeavisideTheta[t - 1]) + (1 + t^2) * HeavisideTheta[t - 1];
```

In[41]:=
```
Plot[f[t], {t, 0, 3}, PlotStyle → {Orange, Thick},
 PlotLegends → Placed[{"f[t]=(1+t)^2* (1-u[t-1])+(1+t^2)*u[t-1]"}, {0.3, 0.85}],
 Exclusions → None, Frame → True]
```

Out[41]=



- Find the Laplace transform of the function.

In[42]:=
```
F[s_] := LaplaceTransform[f[t], t, s]; F[s]
```

Out[42]=

$$\frac{2}{s^3}+\frac{2}{s^2}+\frac{1}{s}-\frac{2\,e^{-s}\,(1+s)}{s^2}$$

- Find the Inverse of the Laplace transform of the function.

In[43]:=
```
f2[t_] := InverseLaplaceTransform[F[s], s, t]; f2[t]
```

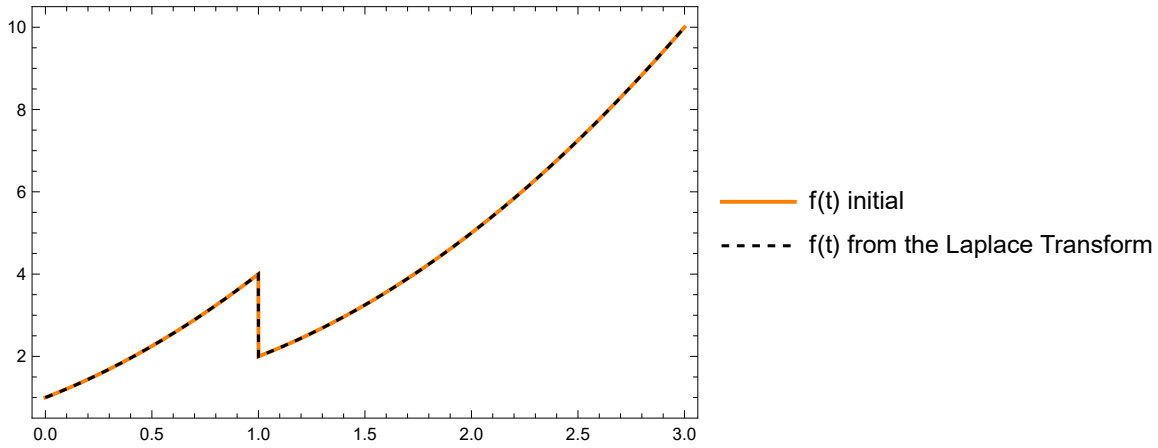
Out[43]=

$$(1+t)^2-2\,t\,\text{HeavisideTheta}[-1+t]$$

- As we can see, the initial function and the expression obtained from the Laplace transform method yield the same result.

In[44]:=
```
Plot[{f[t], f2[t]}, {t, 0, 3},
 PlotStyle → {{Orange, Thick}, {Black, Dashed}}, Exclusions → None, Frame → True,
 PlotLegends → {"f(t) initial", "f(t) from the Laplace Transform"}]
```

Out[44]=



Dirac's Delta Function (Impulse Function)

- DiracDelta[x] returns 0 for all real numeric x other than 0.
- DiracDelta can be used in integrals, integral transforms, and differential equations.
- Some transformations are done automatically when DiracDelta appears in a product of terms.
- ⚠ **Differentiate the Heaviside function to obtain DiracDelta:**

In[45]:=
```
D[HeavisideTheta[x], x]
```
Out[45]=
```
DiracDelta[x]
```

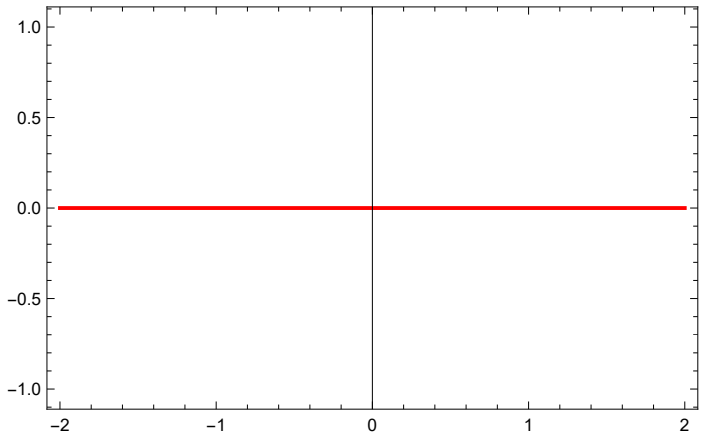
- ⚠ DiracDelta vanishes for nonzero arguments:

In[46]:=
```
DiracDelta[1 / 2]
```
Out[46]=
```
0
```

- ⚠ DiracDelta stays unevaluated for $x = 0$:

In[47]:=
```
DiracDelta[0]
```
Out[47]=
```
DiracDelta[0]
```

In[48]:=
```
Plot[DiracDelta[x], {x, -2, 2}, AxesOrigin → {0, -1},
 PlotStyle → {Red, Thick}, Exclusions → None, Frame → True]
```
Out[48]=



- Use DiracDelta in an integral:

In[49]:=
```
Integrate[DiracDelta[x] Cos[x], {x, -Infinity, Infinity}]
```
Out[49]=
```
1

1
```

Properties of Dirac's Delta

$$\delta(t-a) = \begin{cases} \infty, & t = a \\ 0, & t \neq a \end{cases}, \quad \int_{-\infty}^{+\infty} \delta(t-a)\, dt = 1$$

For $a > 0$, we have

$$\int_{0}^{+\infty} \delta(t-a)\, dt = 1$$

In[50]:=
```
Integrate[DiracDelta[t - a], {t, 0, Infinity}, Assumptions → {a ∈ Reals && a > 0}]
```
Out[50]=
```
1
```

For a continuous function $g(t)$, we have

$$\int_{0}^{+\infty} g(t)\, \delta(t-a)\, dt = g(a)$$

In[51]:=
```
Integrate[g[t] * DiracDelta[t - a], {t, 0, Infinity}, Assumptions → {a ∈ Reals && a > 0}]
```
Out[51]=
```
g[a]
```

What is the Laplace Transform of the Dirac's Delta Function?

$$\mathcal{L}\{\delta(t-a)\} = \int_{0}^{+\infty} e^{-st}\, \delta(t-a)\, dt = e^{-as}$$

- Laplace transform of $\delta(t-a)$:

In[52]:=
```
LaplaceTransform[DiracDelta[t - a], t, s]
```
Out[52]=
```
ℯ^(-a s) HeavisideTheta[a]
```

In[53]:=
```
FullSimplify[LaplaceTransform[DiracDelta[t - a], t, s], a ∈ Reals && a > 0]
```
Out[53]=
```
ℯ^(-a s)
```

- Laplace transform of $125\, \delta\left(t - \frac{1}{3}\pi\right)$:

In[54]:=
```
LaplaceTransform[125 * DiracDelta[t - Pi / 3], t, s]
```

Out[54]=

$$125\, e^{-\frac{\pi s}{3}}$$

$$125\, e^{-\frac{\pi s}{3}}$$

Summary

After completing this chapter, you should be able to

- perform Laplace and inverse Laplace transforms using Wolfram Mathematica
- analyze special functions such as the unit step function and the Dirac delta function
- learn and use information, tools, and technology to solve engineering math problems.

Week 6: Laplace Transforms (Part 2)

Applications of Laplace Transforms

Table of Contents

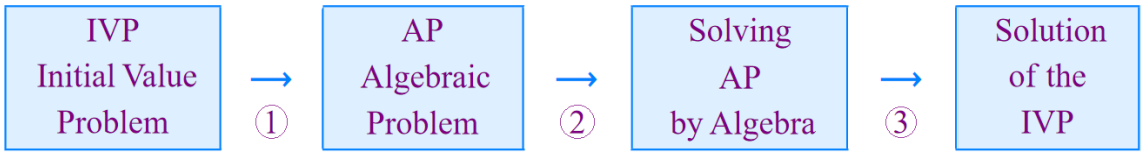
Commands list

- LaplaceTransform[*f*[*t*],*t*,*s*]
- InverseLaplaceTransform[*F*[s],*s*,*t*]
- HeavisideTheta[*x*]
- UnitStep[*x*]
- DiracDelta[*x*]
- Convolve[*f*, *g*, *x*, *y*]

Solving an IVP by Laplace Transforms: The SOP

The process of solving an ODE using Laplace Transform method consists of three steps shown below:

- **Step 1.** The given ODE is transformed into an algebraic equation, called the **subsidiar equation.**
- **Step 2.** The subsidiary equation is solved by purely algebraic manipulations.
- **Step 3.** The solution in Step 2 is transformed back, resulting in the solution of the give probem.



Example 6.1

$$y''(t) + 2y'(t) + 15y(t) = te^{-t} \qquad y(0) = 0,\ y'(0) = 1$$

This example and its sample solutions were developed by Prof. Katharine Long, Texas Tech University - Math Dept.

```
In[•]:= ClearAll["Global`*"]
```

- **Step 0.** Write the ODE as an equation, and the initial conditions as a set of substitution rules.

```
In[•]:= myODE = y''[t] + 2 y'[t] + 15 y[t] == t Exp[-t]
```

```
Out[•]=
15 y[t] + 2 y'[t] + y''[t] == e^-t t
```

```
In[•]:= IC = {y[0] → 0, y'[0] → 1}
```

```
Out[•]=
{y[0] → 0, y'[0] → 1}
```

- **Step 1.** Take Laplace transforms of both sides of the equation, and substitute the initial conditions into the equation.

```
In[•]:= ltODE = LaplaceTransform[myODE, t, s] /. IC
```

```
Out[•]=
-1 + 15 LaplaceTransform[y[t], t, s] +
  2 s LaplaceTransform[y[t], t, s] + s^2 LaplaceTransform[y[t], t, s] == 1/(1 + s)^2
```

- This equation will be easier to read if we write **Y(s)** for **$\mathcal{L}${y(t)}(s)**, which we can do using a substitution rule.

```
In[•]:= eqnForY = ltODE /. LaplaceTransform[y[t], t, s] → Y[s]
```

```
Out[•]=
-1 + 15 Y[s] + 2 s Y[s] + s^2 Y[s] == 1/(1 + s)^2
```

- **Step 2.** Solve the subsidiary equation by algebraic manipulations.

```
In[•]:= Solve[eqnForY, Y[s]]
```

```
Out[•]=
{{Y[s] → (2 + 2 s + s^2)/((1 + s)^2 (15 + 2 s + s^2))}}
```

In[ ]:=
```
YSoln[s_] := Y[s] /. Solve[eqnForY, Y[s]][[1]]; YSoln[s]
```

Out[ ]=

$$\frac{2 + 2\,s + s^2}{(1+s)^2\,\left(15 + 2\,s + s^2\right)}$$

- Now we have computed the Laplace transform of the solution. Take its inverse Laplace transform to get the solution.
- **Step 3.** The solution in Step 2, **Y(s),** is transformed back, resulting in the solution of the given problem.

In[ ]:=
```
InverseLaplaceTransform[YSoln[s], s, t]
```

Out[ ]=

$$\frac{1}{196}\,e^{-t}\left(14\,t + 13\,\sqrt{14}\,\text{Sin}\left[\sqrt{14}\,t\right]\right)$$

In[ ]:=
```
ySoln[t_] = InverseLaplaceTransform[YSoln[s], s, t]; ySoln[t]
```

Out[ ]=

$$\frac{1}{196}\,e^{-t}\left(14\,t + 13\,\sqrt{14}\,\text{Sin}\left[\sqrt{14}\,t\right]\right)$$

- **Step 4.** Verify the solution.

In[ ]:=
```
ODECheck = myODE /. y → ySoln
```

Out[ ]=

$$-\frac{1}{98}\,e^{-t}\left(14 + 182\,\text{Cos}\left[\sqrt{14}\,t\right]\right) - \frac{13\,e^{-t}\,\text{Sin}\left[\sqrt{14}\,t\right]}{\sqrt{14}} + \frac{4}{49}\,e^{-t}\left(14\,t + 13\,\sqrt{14}\,\text{Sin}\left[\sqrt{14}\,t\right]\right) + 2\left(\frac{1}{196}\,e^{-t}\left(14 + 182\,\text{Cos}\left[\sqrt{14}\,t\right]\right) - \frac{1}{196}\,e^{-t}\left(14\,t + 13\,\sqrt{14}\,\text{Sin}\left[\sqrt{14}\,t\right]\right)\right) == e^{-t}\,t$$

In[ ]:=
```
FullSimplify[ODECheck]
```

Out[ ]=
```
True
```

In[ ]:=
```
ICCheck = {ySoln[0] == y[0], ySoln'[0] == y'[0]} /. IC
```

Out[ ]=
```
{True, True}
```

- **Step 5.** Verify the solution by **DSolve** function (Not Required).
- A general solution:

In[ ]:=
```
DsolveSoln0 = DSolve[myODE, y[t], t]
```

Out[ ]=

$$\left\{\left\{y[t] \to e^{-t}\,c_2\,\text{Cos}\left[\sqrt{14}\,t\right] + e^{-t}\,c_1\,\text{Sin}\left[\sqrt{14}\,t\right] + \frac{1}{14}\,e^{-t}\,t\left(\text{Cos}\left[\sqrt{14}\,t\right]^2 + \text{Sin}\left[\sqrt{14}\,t\right]^2\right)\right\}\right\}$$

- A particular solution:

In[ ]:=
```
DsolveSoln1 = DSolve[{myODE, y[0] == 0, y'[0] == 1} , y[t], t]
```

Out[ ]=

$$\left\{\left\{y[t] \to \frac{1}{196}\, e^{-t} \left(14\, t \,\text{Cos}\left[\sqrt{14}\, t\right]^2 + 13\, \sqrt{14}\, \text{Sin}\left[\sqrt{14}\, t\right] + 14\, t\, \text{Sin}\left[\sqrt{14}\, t\right]^2\right)\right\}\right\}$$

In[ ]:=
```
FullSimplify[Chop[DsolveSoln1[[1]]]]
```

Out[ ]=

$$\left\{y[t] \to \frac{1}{196}\, e^{-t} \left(14\, t + 13\, \sqrt{14}\, \text{Sin}\left[\sqrt{14}\, t\right]\right)\right\}$$

- Result from DSolve:

In[ ]:=
```
yfromDsolve = y[t] /. FullSimplify[Chop[DsolveSoln1[[1]]]]
```

Out[ ]=

$$\frac{1}{196}\, e^{-t} \left(14\, t + 13\, \sqrt{14}\, \text{Sin}\left[\sqrt{14}\, t\right]\right)$$

- Solution from the method of Laplace transform:

In[ ]:=
```
ySoln[t]
```

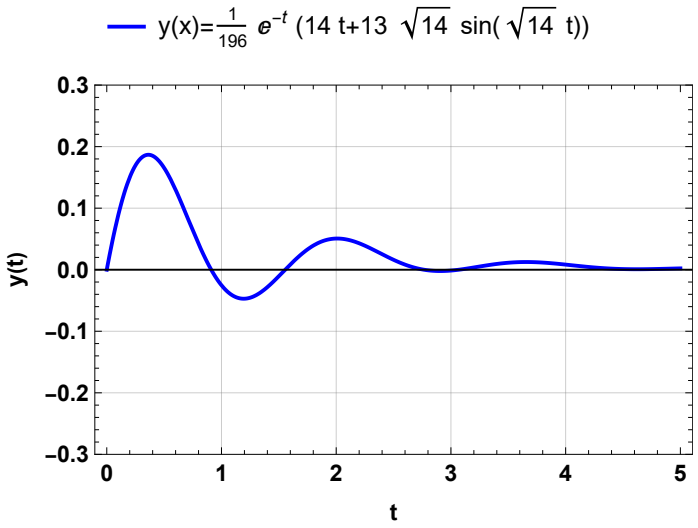
Out[ ]=

$$\frac{1}{196}\, e^{-t} \left(14\, t + 13\, \sqrt{14}\, \text{Sin}\left[\sqrt{14}\, t\right]\right)$$

- Also, let's take a look at the solution by plotting its graph:

```
In[ ]:= Plot[{ySoln[t]}, {t, 0, 5}, PlotRange → {-0.3, 0.3},
 PlotStyle → {Blue, Thick}, Frame → True, FrameLabel → {"t", "y(t)"},
 BaseStyle → {FontWeight → "Bold", Black, FontSize → 12}, GridLines → Automatic,
 PlotLegends → Placed[{"y(x) = 1/196 e^-t (14 t+13 √14 sin(√14 t))"}, Above],
 AxesStyle → Directive[RGBColor[0.`, 0.`, 0.`], AbsoluteThickness[1]],
 Method → {"DefaultBoundaryStyle" → Automatic,
   "DefaultMeshStyle" → AbsolutePointSize[6], "ScalingFunctions" → None}]
```

Out[ ]=



Example 6.2

$$y''(t) + 2\,y'(t) + 5\,y(t) = 1.25\exp(0.5\,t) + 40\cos(4\,t) - 55\sin(4\,t)$$

$$y(0) = 0.2,\ y'(0) = 60.1$$

```
In[ ]:= ClearAll["Global`*"]
```

- **Step 0.** Write the ODE as an equation, and the initial conditions as a set of substitution rules.

```
In[ ]:= LHSOp[y_, t_] = y''[t] + 2.0 y'[t] + 5.0 y[t]
```

Out[ ]=

```
5. y[t] + 2. y'[t] + y''[t]
```

```
In[ ]:= rhsFunc[t_] = 1.25 Exp[0.5 t] + 40.0 Cos[4.0 t] - 55.0 Sin[4.0 t]
```

Out[ ]=

```
1.25 ℯ^(0.5 t) + 40. Cos[4. t] - 55. Sin[4. t]
```

```
In[ ]:= myODE = LHSOp[y, t] == rhsFunc[t]
```

Out[ ]=

```
5. y[t] + 2. y'[t] + y''[t] == 1.25 ℯ^(0.5 t) + 40. Cos[4. t] - 55. Sin[4. t]
```

In[•]:=
```
IC = {y[0] → 0.2, y'[0] → 60.1}
```
Out[•]=
$$\{y[0] \to 0.2,\ y'[0] \to 60.1\}$$

- **Step 1.** Take Laplace transforms of both sides of the equation, and substitute the initial conditions into the equation.

In[•]:=
```
ltODE = LaplaceTransform[myODE, t, s] /. IC
```
Out[•]=
$$-60.1 - 0.2\,s + 5.\,\text{LaplaceTransform}[y[t], t, s] + s^2\,\text{LaplaceTransform}[y[t], t, s] + 2.\,(-0.2 + s\,\text{LaplaceTransform}[y[t], t, s]) == \frac{1.25}{-0.5 + s} - \frac{220.}{16. + s^2} + \frac{40.\,s}{16. + s^2}$$

- This equation will be easier to read if we write **Y(s)** for **$\mathcal{L}${y(t)}(s)**, which we can do using a substitution rule.

In[•]:=
```
eqnForY = ltODE /. LaplaceTransform[y[t], t, s] → Y[s]
```
Out[•]=
$$-60.1 - 0.2\,s + 5.\,Y[s] + s^2\,Y[s] + 2.\,(-0.2 + s\,Y[s]) == \frac{1.25}{-0.5 + s} - \frac{220.}{16. + s^2} + \frac{40.\,s}{16. + s^2}$$

- **Step 2.** Solve the subsidiary equation by algebraic manipulations.

In[•]:=
```
Solve[eqnForY, Y[s]]
```
Out[•]=
$$\left\{\left\{Y[s] \to \frac{60.5 + \frac{1.25}{-0.5+s} + 0.2\,s - \frac{220.}{16.+s^2} + \frac{40.\,s}{16.+s^2}}{5. + 2.\,s + s^2}\right\}\right\}$$

In[•]:=
```
YSoln[s_] := Y[s] /. Solve[eqnForY, Y[s]][[1]]; YSoln[s]
```
Out[•]=
$$\frac{60.5 + \frac{1.25}{-0.5+s} + 0.2\,s - \frac{220.}{16.+s^2} + \frac{40.\,s}{16.+s^2}}{5. + 2.\,s + s^2}$$

- Now we have computed the Laplace transform of the solution. Take its inverse Laplace transform to get the solution.
- **Step 3.** The solution in Step 2, **Y(s),** is transformed back, resulting in the solution of the given problem.

In[•]:=
```
InverseLaplaceTransform[YSoln[s], s, t]
```
Out[•]=
$$0.2\,e^{0.5\,t} + e^{(-1.-2.\,i)\,t}\left(\left(-1.19349 \times 10^{-15} + 10.\,i\right) - \left(1.19349 \times 10^{-15} + 10.\,i\right) e^{(0.+4.\,i)\,t}\right) + e^{(0.-4.\,i)\,t}\left(\left(-4.44089 \times 10^{-16} + 2.5\,i\right) - \left(4.44089 \times 10^{-16} + 2.5\,i\right) e^{(0.+8.\,i)\,t}\right)$$

In[•]:=
```
ySoln[t_] = FullSimplify[Chop[InverseLaplaceTransform[YSoln[s], s, t]]]; ySoln[t]
```
Out[•]=
$$0.2\,\text{Cosh}[0.5\,t] + 20.\,e^{-1.\,t}\,\text{Sin}[2.\,t] + 5.\,\text{Sin}[4.\,t] + 0.2\,\text{Sinh}[0.5\,t]$$

◆ **Step 4.** Verify the solution.

In[ ]:=
```
myODE
```
Out[ ]=
$$5.\,y[t]+2.\,y'[t]+y''[t]==1.25\,e^{0.5\,t}+40.\,\text{Cos}[4.\,t]-55.\,\text{Sin}[4.\,t]$$

In[ ]:=
```
LHS = LHSOp[ySoln, t]
```
Out[ ]=
$$-80.\,e^{-1.\,t}\,\text{Cos}[2.\,t]+0.05\,\text{Cosh}[0.5\,t]-60.\,e^{-1.\,t}\,\text{Sin}[2.\,t]-80.\,\text{Sin}[4.\,t]+2.\left(40.\,e^{-1.\,t}\,\text{Cos}[2.\,t]+20.\,\text{Cos}[4.\,t]+0.1\,\text{Cosh}[0.5\,t]-20.\,e^{-1.\,t}\,\text{Sin}[2.\,t]+0.1\,\text{Sinh}[0.5\,t]\right)+5.\left(0.2\,\text{Cosh}[0.5\,t]+20.\,e^{-1.\,t}\,\text{Sin}[2.\,t]+5.\,\text{Sin}[4.\,t]+0.2\,\text{Sinh}[0.5\,t]\right)+0.05\,\text{Sinh}[0.5\,t]$$

In[ ]:=
```
RHS = rhsFunc[t]
```
Out[ ]=
$$1.25\,e^{0.5\,t}+40.\,\text{Cos}[4.\,t]-55.\,\text{Sin}[4.\,t]$$

In[ ]:=
```
Chop[FullSimplify[LHS == RHS]]
```
Out[ ]=
```
True
```

In[ ]:=
```
IC
```
Out[ ]=
$$\{y[0]\to 0.2,\ y'[0]\to 60.1\}$$

In[ ]:=
```
{ySoln[0], ySoln'[0]}
```
Out[ ]=
$$\{0.2,\ 60.1\}$$

◆ **Step 5.** Verify the solution by **DSolve** function (Not Required).

◆ A general solution:

```
DsolveSoln0 = DSolve[LHSOp[y, x] == rhsFunc[x], y[x], x]
```
Out[ ]=
$$\left\{\left\{y[x]\to e^{-1.\,x}\,c_2\,\text{Cos}[2.\,x]+e^{-1.\,x}\,c_1\,\text{Sin}[2.\,x]+5.\,e^{-1.\,x}\left(0.+0.04\,e^{1.5\,x}\,\text{Cos}[2.\,x]^2+0.04\,e^{1.5\,x}\,\text{Sin}[2.\,x]^2+1.\,e^{1.\,x}\,\text{Cos}[2.\,x]^2\,\text{Sin}[4.\,x]+1.\,e^{1.\,x}\,\text{Sin}[2.\,x]^2\,\text{Sin}[4.\,x]\right)\right\}\right\}$$

◆ A particular solution:

```
DsolveSoln1 = DSolve[{LHSOp[y, x] == rhsFunc[x], y[0] == 0.2, y'[0] == 60.1} , y[x], x]
```
Out[ ]=
$$\left\{\left\{y[x]\to 5.\,e^{-1.\,x}\left(0.04\,e^{1.5\,x}\,\text{Cos}[2.\,x]^2+4.\,\text{Sin}[2.\,x]+0.04\,e^{1.5\,x}\,\text{Sin}[2.\,x]^2+1.\,e^{1.\,x}\,\text{Cos}[2.\,x]^2\,\text{Sin}[4.\,x]+1.\,e^{1.\,x}\,\text{Sin}[2.\,x]^2\,\text{Sin}[4.\,x]\right)\right\}\right\}$$

In[ ]:=
```
FullSimplify[Chop[DsolveSoln1[[1]]]]
```
Out[ ]=
$$\left\{y[x]\to 0.2\,e^{0.5\,x}+20.\,e^{-1.\,x}\,\text{Sin}[2.\,x]+5.\,\text{Sin}[4.\,x]\right\}$$

◆ Result from DSolve:

```
yfromDsolve = y[x] /. FullSimplify[Chop[DsolveSoln1[[1]]]]
```

Out[ ]=

$$0.2\, e^{0.5\,x} + 20.\, e^{-1.\,x} \sin[2.\,x] + 5.\, \sin[4.\,x]$$

- Solution from the method of Laplace transform:

```
ySoln[t]
```

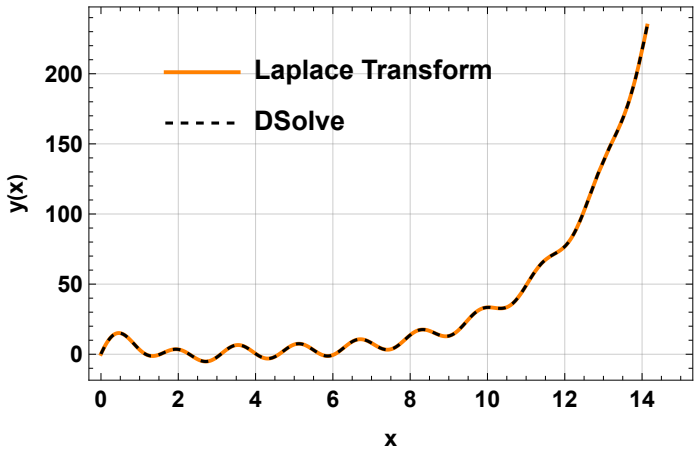
Out[ ]=

$$0.2 \cosh[0.5\,t] + 20.\, e^{-1.\,t} \sin[2.\,t] + 5.\, \sin[4.\,t] + 0.2 \sinh[0.5\,t]$$

- Let's compare the obtained results by plotting them on the same graph:

In[ ]:=
```
Plot[{ySoln[x] , yfromDsolve}, {x, 0, 15}, Frame → True,
 PlotStyle → {{Orange, Thick}, {Black, Dashed}}, Frame → True, FrameLabel → {"x", "y(x)"},
 BaseStyle → {FontWeight → "Bold", Black, FontSize → 12}, GridLines → Automatic,
 AxesStyle → Directive[RGBColor[0.`, 0.`, 0.`], AbsoluteThickness[1]],
 Method → {"DefaultBoundaryStyle" → Automatic,
   "DefaultMeshStyle" → AbsolutePointSize[6], "ScalingFunctions" → None},
 PlotLegends → Placed[{"Laplace Transform", "DSolve"}, {0.4, 0.75}], Background → White]
```

Out[ ]=



Modeling Mass-Spring System using the Unit Step & the Dirac's Delta Functions

Mass-Spring System Under a Square Wave

Determine the response of a damped mass-spring system under a square wave, modelled by

$$y''(t) + 3\,y'(t) + 2\,y(t) = r(t) = u(t-1) - u(t-2)$$

$$y(0) = 0,\ y'(0) = 0$$

This example is taken from the Textbook (Kreyszig, 2011, 10th Edition), Section 6.4, page 227.

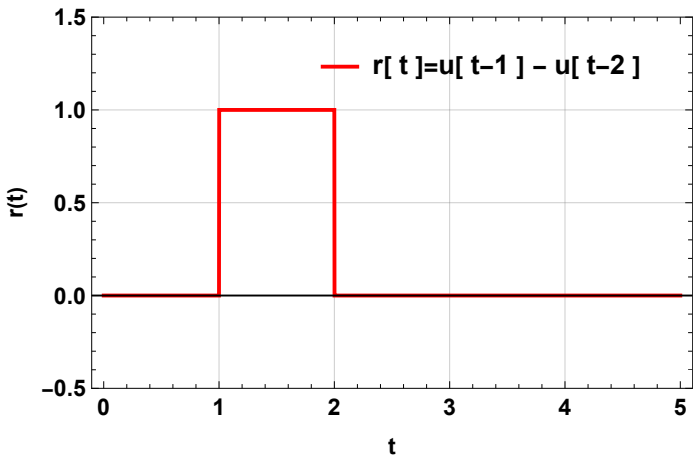
```
In[ ]:= ClearAll["Global`*"]
```

- Let's define the RHS as a function of *vm* and plot it:

```
In[ ]:= r[t_] := HeavisideTheta[t - 1] - HeavisideTheta[t - 2];
```

```
In[ ]:= Plot[r[t], {t, 0, 5}, PlotRange → {-0.5, 1.5}, PlotStyle → {{Red, Thick}},
  Frame → True, Exclusions → None, FrameLabel → {"t", "r(t)"},
  BaseStyle → {FontWeight → "Bold", Black, FontSize → 12}, GridLines → Automatic,
  AxesStyle → Directive[RGBColor[0.`, 0.`, 0.`], AbsoluteThickness[1]],
  PlotLegends → Placed[{"r[ t ]=u[ t-1 ] - u[ t-2 ]"}, {0.65, 0.87}],
  Method → {"DefaultBoundaryStyle" → Automatic,
    "DefaultMeshStyle" → AbsolutePointSize[6], "ScalingFunctions" → None}]
```

```
Out[ ]=
```



- **Step 0.** Write the ODE as an equation, and the initial conditions as a set of substitution rules.

```
In[ ]:= myODE = y''[t] + 3 y'[t] + 2 y[t] == r[t]
```

```
Out[ ]=
```

$$2\,y[t] + 3\,y'[t] + y''[t] == -\text{HeavisideTheta}[-2 + t] + \text{HeavisideTheta}[-1 + t]$$

```
In[ ]:= IC = {y[0] → 0, y'[0] → 0}
```

```
Out[ ]=
```

$$\{y[0] \to 0,\ y'[0] \to 0\}$$

- **Step 1.** Take Laplace transforms of both sides of the equation, and substitute the initial conditions into the equation.

```
In[ ]:= ltODE = LaplaceTransform[myODE, t, s] /. IC
```

```
Out[ ]=
```

$$2\,\text{LaplaceTransform}[y[t], t, s] + 3\,s\,\text{LaplaceTransform}[y[t], t, s] + s^2\,\text{LaplaceTransform}[y[t], t, s] == -\frac{e^{-2s}}{s} + \frac{e^{-s}}{s}$$

- This equation will be easier to read if we write **Y(s)** for **$\mathcal{L}${y(t)}(s)**, which we can do using a substitution rule.

In[ ]:=
```
eqnForY = ltODE /. LaplaceTransform[y[t], t, s] → Y[s]
```

Out[ ]=

$$2\,Y[s] + 3\,s\,Y[s] + s^2\,Y[s] == -\frac{e^{-2\,s}}{s} + \frac{e^{-s}}{s}$$

- **Step 2.** Solve the subsidiary equation by algebraic manipulations.

In[ ]:=
```
Solve[eqnForY, Y[s]]
```

Out[ ]=

$$\left\{\left\{Y[s] \to \frac{e^{-2\,s}\left(-1 + e^{s}\right)}{s\left(2 + 3\,s + s^2\right)}\right\}\right\}$$

In[ ]:=
```
YSoln[s_] := Y[s] /. Solve[eqnForY, Y[s]][[1]]; YSoln[s]
```

Out[ ]=

$$\frac{e^{-2\,s}\left(-1 + e^{s}\right)}{s\left(2 + 3\,s + s^2\right)}$$

- Now we have computed the Laplace transform of the solution. Take its inverse Laplace transform to get the solution.
- **Step 3.** The solution in Step 2, **Y(s),** is transformed back, resulting in the solution of the given problem.

In[ ]:=
```
InverseLaplaceTransform[YSoln[s], s, t]
```

Out[ ]=

$$-\frac{1}{2}\,e^{-2\,(-2+t)}\left(-1 + e^{-2+t}\right)^2\,\text{HeavisideTheta}[-2 + t] + \frac{1}{2}\,e^{-2\,(-1+t)}\left(-1 + e^{-1+t}\right)^2\,\text{HeavisideTheta}[-1 + t]$$

In[ ]:=
```
ySoln[t_] = InverseLaplaceTransform[YSoln[s], s, t]; ySoln[t]
```

Out[ ]=

$$-\frac{1}{2}\,e^{-2\,(-2+t)}\left(-1 + e^{-2+t}\right)^2\,\text{HeavisideTheta}[-2 + t] + \frac{1}{2}\,e^{-2\,(-1+t)}\left(-1 + e^{-1+t}\right)^2\,\text{HeavisideTheta}[-1 + t]$$

- **Step 4.** Verify the solution.

In[ ]:=
```
ODECheck = myODE /. y → ySoln
```

Out[ ]=

$$-2\,e^{-2-2\,(-2+t)+t}\left(-1+e^{-2+t}\right)\text{DiracDelta}[-2+t]+2\,e^{-2\,(-2+t)}\left(-1+e^{-2+t}\right)^2\text{DiracDelta}[-2+t]+$$
$$2\,e^{-1-2\,(-1+t)+t}\left(-1+e^{-1+t}\right)\text{DiracDelta}[-1+t]-2\,e^{-2\,(-1+t)}\left(-1+e^{-1+t}\right)^2\text{DiracDelta}[-1+t]-$$
$$e^{-4-2\,(-2+t)+2\,t}\,\text{HeavisideTheta}[-2+t]+3\,e^{-2-2\,(-2+t)+t}\left(-1+e^{-2+t}\right)\text{HeavisideTheta}[-2+t]-$$
$$2\,e^{-2\,(-2+t)}\left(-1+e^{-2+t}\right)^2\text{HeavisideTheta}[-2+t]+e^{-2-2\,(-1+t)+2\,t}\,\text{HeavisideTheta}[-1+t]-$$
$$3\,e^{-1-2\,(-1+t)+t}\left(-1+e^{-1+t}\right)\text{HeavisideTheta}[-1+t]+$$
$$2\,e^{-2\,(-1+t)}\left(-1+e^{-1+t}\right)^2\text{HeavisideTheta}[-1+t]+$$
$$3\left(-\frac{1}{2}\,e^{-2\,(-2+t)}\left(-1+e^{-2+t}\right)^2\text{DiracDelta}[-2+t]+\frac{1}{2}\,e^{-2\,(-1+t)}\left(-1+e^{-1+t}\right)^2\text{DiracDelta}[-1+t]-\right.$$
$$e^{-2-2\,(-2+t)+t}\left(-1+e^{-2+t}\right)\text{HeavisideTheta}[-2+t]+$$
$$e^{-2\,(-2+t)}\left(-1+e^{-2+t}\right)^2\text{HeavisideTheta}[-2+t]+e^{-1-2\,(-1+t)+t}\left(-1+e^{-1+t}\right)$$
$$\left.\text{HeavisideTheta}[-1+t]-e^{-2\,(-1+t)}\left(-1+e^{-1+t}\right)^2\text{HeavisideTheta}[-1+t]\right)+$$
$$2\left(-\frac{1}{2}\,e^{-2\,(-2+t)}\left(-1+e^{-2+t}\right)^2\text{HeavisideTheta}[-2+t]+\right.$$
$$\left.\frac{1}{2}\,e^{-2\,(-1+t)}\left(-1+e^{-1+t}\right)^2\text{HeavisideTheta}[-1+t]\right)-$$
$$\frac{1}{2}\,e^{-2\,(-2+t)}\left(-1+e^{-2+t}\right)^2\text{DiracDelta}'[-2+t]+\frac{1}{2}\,e^{-2\,(-1+t)}\left(-1+e^{-1+t}\right)^2\text{DiracDelta}'[-1+t]==$$
$$-\text{HeavisideTheta}[-2+t]+\text{HeavisideTheta}[-1+t]$$

In[ ]:=
```
FullSimplify[ODECheck]
```

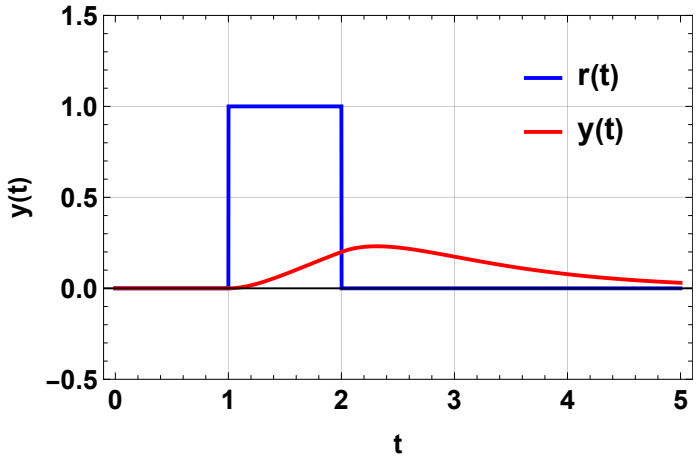
Out[ ]=
```
True
```

In[ ]:=
```
ICCheck = {ySoln[0] == y[0], ySoln'[0] == y'[0]} /. IC
```

Out[ ]=
```
{True, True}
```

- Now, let's take a look at the solution by plotting it:

```
Plot[{r[t], ySoln[t]}, {t, 0, 5}, PlotRange → {-0.5, 1.5},
 PlotStyle → {{Blue, Thick}, {Red, Thick}}, Frame → True,
 FrameLabel → {"t", "y(t)"}, Exclusions → None,
 BaseStyle → {FontWeight → "Bold", Black, FontSize → 12}, GridLines → Automatic,
 AxesStyle → Directive[RGBColor[0.`, 0.`, 0.`], AbsoluteThickness[1]],
 Method → {"DefaultBoundaryStyle" → Automatic,
   "DefaultMeshStyle" → AbsolutePointSize[6], "ScalingFunctions" → None},
 PlotLegends → Placed[{"r(t)", "y(t)"}, {0.8, 0.75}], Background → White]
```



Hammer-blow Response of a Mass-Spring System

Determine the response of a mass-spring system under a unit impulse at $t = 1$,i.e., $r(t) = \delta(t-1)$, modelled by

$$y''(t) + 3y'(t) + 2y(t) = r(t) \qquad y(0) = 0,\ y'(0) = 0$$

This example is taken from the Textbook (Kreyszig, 2011, 10th Edition), Section 6.4, page 227.

In[ ]:=
```
ClearAll["Global`*"]
```

- Let's define the RHS as a function of *r(t)*.

In[ ]:=
```
r[t_] := DiracDelta[t - 1]; r[t]
```

Out[ ]=
```
DiracDelta[-1 + t]
```

- **Step 0.** Write the ODE as an equation, and the initial conditions as a set of substitution rules.

In[ ]:=
```
myODE = y''[t] + 3 y'[t] + 2 y[t] == r[t]
```

Out[ ]=
```
2 y[t] + 3 y'[t] + y''[t] == DiracDelta[-1 + t]
```

```
In[•]:= IC = {y[0] → 0, y'[0] → 0}
Out[•]=
{y[0] → 0, y′[0] → 0}
```

- **Step 1.** Take Laplace transforms of both sides of the equation, and substitute the initial conditions into the equation.

```
In[•]:= ltODE = LaplaceTransform[myODE, t, s] /. IC
Out[•]=
2 LaplaceTransform[y[t], t, s] +
  3 s LaplaceTransform[y[t], t, s] + s^2 LaplaceTransform[y[t], t, s] == e^-s
```

- This equation will be easier to read if we write **Y(s)** for **ℒ{y(t)}(s)**, which we can do using a substitution rule.

```
In[•]:= eqnForY = ltODE /. LaplaceTransform[y[t], t, s] → Y[s]
Out[•]=
2 Y[s] + 3 s Y[s] + s^2 Y[s] == e^-s
```

- **Step 2.** Solve the subsidiary equation by algebraic manipulations.

```
In[•]:= Solve[eqnForY, Y[s]]
Out[•]=
{{Y[s] → e^-s / (2 + 3 s + s^2)}}
```

```
In[•]:= YSoln[s_] := Y[s] /. Solve[eqnForY, Y[s]][[1]]; YSoln[s]
Out[•]=
e^-s / (2 + 3 s + s^2)
```

- Now we have computed the Laplace transform of the solution. Take its inverse Laplace transform to get the solution.
- **Step 3.** The solution in Step 2, **Y(s)**, is transformed back, resulting in the solution of the given problem.

```
In[•]:= InverseLaplaceTransform[YSoln[s], s, t]
Out[•]=
e^(1-2 t) (-e + e^t) HeavisideTheta[-1 + t]
```

```
In[•]:= ySoln[t_] = InverseLaplaceTransform[YSoln[s], s, t]; ySoln[t]
Out[•]=
e^(1-2t) (-e + e^t) HeavisideTheta[-1 + t]
```

- **Step 4.** Verify the solution.

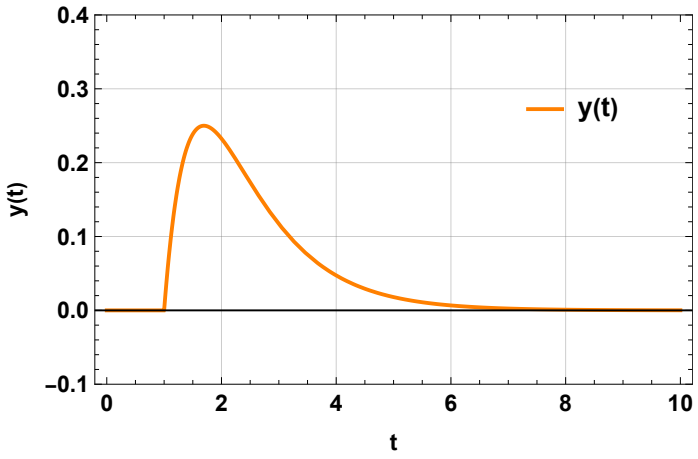
In[•]:=
```
ODECheck = myODE /. y → ySoln
```
Out[•]=
$$2\,e^{1-t}\,\text{DiracDelta}[-1+t] - 4\,e^{1-2t}\left(-e+e^{t}\right)\text{DiracDelta}[-1+t] - 3\,e^{1-t}\,\text{HeavisideTheta}[-1+t] + 6\,e^{1-2t}\left(-e+e^{t}\right)\text{HeavisideTheta}[-1+t] + 3\left(e^{1-2t}\left(-e+e^{t}\right)\text{DiracDelta}[-1+t] + e^{1-t}\,\text{HeavisideTheta}[-1+t] - 2\,e^{1-2t}\left(-e+e^{t}\right)\text{HeavisideTheta}[-1+t]\right) + e^{1-2t}\left(-e+e^{t}\right)\text{DiracDelta}'[-1+t] == \text{DiracDelta}[-1+t]$$

In[•]:=
```
FullSimplify[ODECheck]
```
Out[•]=
```
True
```

In[•]:=
```
ICCheck = {ySoln[0] == y[0], ySoln'[0] == y'[0]} /. IC
```
Out[•]=
```
{True, True}
```

◆ Now, let's take a look at the solution by plotting it:

In[•]:=
```
Plot[{ySoln[t]}, {t, 0, 10}, PlotRange → {-0.1, 0.4}, PlotStyle → {{Orange, Thick}},
 Frame → True, FrameLabel → {"t", "y(t)"}, Exclusions → None,
 BaseStyle → {FontWeight → "Bold", Black, FontSize → 12}, GridLines → Automatic,
 AxesStyle → Directive[RGBColor[0.`, 0.`, 0.`], AbsoluteThickness[1]],
 Method → {"DefaultBoundaryStyle" → Automatic,
   "DefaultMeshStyle" → AbsolutePointSize[6], "ScalingFunctions" → None},
 PlotLegends → Placed[{"y(t)"}, {0.8, 0.75}], Background → White]
```
Out[•]=



Mass-Spring System Under a Sinusoidal Force for Some Time Interval

Determine the response of a mass-spring system under a sinusoidal force for interval, modelled by

$$y''(t) + 3\,y'(t) + 2\,y(t) = r(t), \qquad y(0) = 1,\ y'(0) = -5$$

$$r(t) = \begin{cases} 10\sin(2\,t) & 0 < t < \pi \\ 0 & t > \pi \end{cases}$$

This example is taken from the Textbook (Kreyszig, 2011, 10th Edition), Section 6.4, page 229.

In[•]:=
```
ClearAll["Global`*"]
```

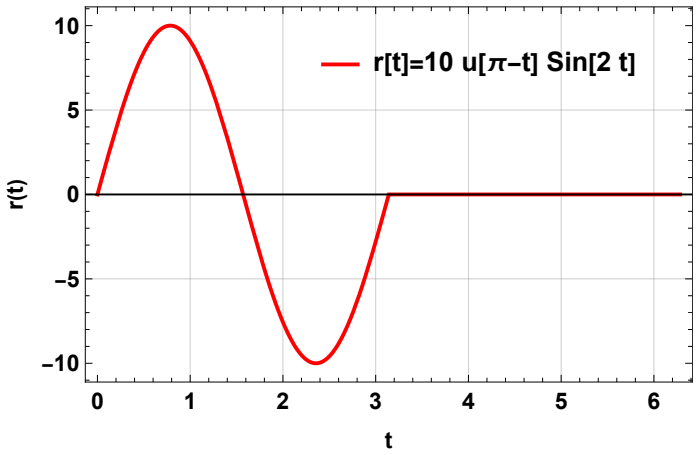
- Let's define the RHS as a function of *r(t)* and plot it.

In[•]:=
```
r[t_] := 10 * Sin[2 t] * HeavisideTheta[Pi - t] ; r[t]
```

Out[•]=
```
10 HeavisideTheta[π - t] Sin[2 t]
```

In[•]:=
```
Plot[r[t], {t, 0, 2 * Pi}, PlotStyle → {{Red, Thick}},
 Frame → True, Exclusions → None, FrameLabel → {"t", "r(t)"},
 BaseStyle → {FontWeight → "Bold", Black, FontSize → 12}, GridLines → Automatic,
 AxesStyle → Directive[RGBColor[0.`, 0.`, 0.`], AbsoluteThickness[1]],
 PlotLegends → Placed[{"r[t]=10 u[π-t] Sin[2 t]"}, {0.65, 0.85}],
 Method → {"DefaultBoundaryStyle" → Automatic,
   "DefaultMeshStyle" → AbsolutePointSize[6], "ScalingFunctions" → None}]
```

Out[•]=



- **Step 0.** Write the ODE as an equation, and the initial conditions as a set of substitution rules.

In[•]:=
```
myODE = y''[t] + 3 y'[t] + 2 y[t] == r[t]
```

Out[•]=
```
2 y[t] + 3 y′[t] + y′′[t] == 10 HeavisideTheta[π - t] Sin[2 t]
```

In[•]:=
```
IC = {y[0] → 1, y'[0] → -5}
```

Out[•]=
```
{y[0] → 1, y′[0] → -5}
```

- **Step 1.** Take Laplace transforms of both sides of the equation, and substitute the initial conditions into the equation.

In[•]:=
```
ltODE = LaplaceTransform[myODE, t, s] /. IC
```

Out[•]=

$$5 - s + 2\,\text{LaplaceTransform}[y[t], t, s] + s^2\,\text{LaplaceTransform}[y[t], t, s] + 3\,(-1 + s\,\text{LaplaceTransform}[y[t], t, s]) == \frac{10\,\left(2 - 2\,e^{-\pi s}\right)}{4 + s^2}$$

- This equation will be easier to read if we write **Y(s)** for **$\mathcal{L}${y(t)}(s)**, which we can do using a substitution rule.

In[•]:=
```
eqnForY = ltODE /. LaplaceTransform[y[t], t, s] → Y[s]
```

Out[•]=

$$5 - s + 2\,Y[s] + s^2\,Y[s] + 3\,(-1 + s\,Y[s]) == \frac{10\,\left(2 - 2\,e^{-\pi s}\right)}{4 + s^2}$$

- **Step 2.** Solve the subsidiary equation by algebraic manipulations.

In[•]:=
```
Solve[eqnForY, Y[s]]
```

Out[•]=

$$\left\{\left\{Y[s] \to \frac{-2 + s + \frac{10\,\left(2 - 2\,e^{-\pi s}\right)}{4 + s^2}}{2 + 3\,s + s^2}\right\}\right\}$$

In[•]:=
```
YSoln[s_] := Y[s] /. Solve[eqnForY, Y[s]][[1]]; YSoln[s]
```

Out[•]=

$$\frac{-2 + s + \frac{10\,\left(2 - 2\,e^{-\pi s}\right)}{4 + s^2}}{2 + 3\,s + s^2}$$

- Now we have computed the Laplace transform of the solution. Take its inverse Laplace transform to get the solution.
- **Step 3.** The solution in Step 2, **Y(s),** is transformed back, resulting in the solution of the given problem.

In[•]:=
```
InverseLaplaceTransform[YSoln[s], s, t]
```

Out[•]=

$$-e^{-2t}\left(-2 + e^{t}\right) - 2\,e^{-2t}\left(-1 + e^{t}\right) + 20\left(-\frac{1}{8}\,e^{-2t} + \frac{e^{-t}}{5} + \frac{1}{40}\,(-3\,\text{Cos}[2t] - \text{Sin}[2t])\right) - 20\,\text{HeavisideTheta}[-\pi + t]\left(\frac{e^{\pi - t}}{5} - \frac{1}{8}\,e^{-2(-\pi + t)} + \frac{1}{40}\,(-3\,\text{Cos}[2\,(-\pi + t)] - \text{Sin}[2\,(-\pi + t)])\right)$$

In[•]:=
```
ySoln[t_] = FullSimplify[InverseLaplaceTransform[YSoln[s], s, t]]; ySoln[t]
```

Out[•]=

$$\frac{1}{2}\,e^{-2t}\left(3 + 5\,e^{2\pi}\,\text{HeavisideTheta}[-\pi + t] + 2\,e^{t}\,(1 - 4\,e^{\pi}\,\text{HeavisideTheta}[-\pi + t]) + e^{2t}\,(-1 + \text{HeavisideTheta}[-\pi + t])\,(3\,\text{Cos}[2t] + \text{Sin}[2t])\right)$$

- **Step 4.** Verify the solution.

In[ ]:=
```
ODECheck = myODE /. y → ySoln
```

Out[ ]=

$$3\,e^{-2t}\left(3+5\,e^{2\pi}\,\text{HeavisideTheta}[-\pi+t]+2\,e^{t}\,(1-4\,e^{\pi}\,\text{HeavisideTheta}[-\pi+t])+e^{2t}\,(-1+\text{HeavisideTheta}[-\pi+t])\,(3\,\text{Cos}[2t]+\text{Sin}[2t])\right)-2\,e^{-2t}\left(5\,e^{2\pi}\,\text{DiracDelta}[-\pi+t]-8\,e^{\pi+t}\,\text{DiracDelta}[-\pi+t]+2\,e^{t}\,(1-4\,e^{\pi}\,\text{HeavisideTheta}[-\pi+t])+e^{2t}\,(-1+\text{HeavisideTheta}[-\pi+t])\,(2\,\text{Cos}[2t]-6\,\text{Sin}[2t])+e^{2t}\,\text{DiracDelta}[-\pi+t]\,(3\,\text{Cos}[2t]+\text{Sin}[2t])+2\,e^{2t}\,(-1+\text{HeavisideTheta}[-\pi+t])\,(3\,\text{Cos}[2t]+\text{Sin}[2t])\right)+3\left(-e^{-2t}\left(3+5\,e^{2\pi}\,\text{HeavisideTheta}[-\pi+t]+2\,e^{t}\,(1-4\,e^{\pi}\,\text{HeavisideTheta}[-\pi+t])+e^{2t}\,(-1+\text{HeavisideTheta}[-\pi+t])\,(3\,\text{Cos}[2t]+\text{Sin}[2t])\right)+\frac{1}{2}\,e^{-2t}\left(5\,e^{2\pi}\,\text{DiracDelta}[-\pi+t]-8\,e^{\pi+t}\,\text{DiracDelta}[-\pi+t]+2\,e^{t}\,(1-4\,e^{\pi}\,\text{HeavisideTheta}[-\pi+t])+e^{2t}\,(-1+\text{HeavisideTheta}[-\pi+t])\,(2\,\text{Cos}[2t]-6\,\text{Sin}[2t])+e^{2t}\,\text{DiracDelta}[-\pi+t]\,(3\,\text{Cos}[2t]+\text{Sin}[2t])+2\,e^{2t}\,(-1+\text{HeavisideTheta}[-\pi+t])\,(3\,\text{Cos}[2t]+\text{Sin}[2t])\right)+\frac{1}{2}\,e^{-2t}\left(-16\,e^{\pi+t}\,\text{DiracDelta}[-\pi+t]+2\,e^{t}\,(1-4\,e^{\pi}\,\text{HeavisideTheta}[-\pi+t])+2\,e^{2t}\,\text{DiracDelta}[-\pi+t]\,(2\,\text{Cos}[2t]-6\,\text{Sin}[2t])+4\,e^{2t}\,(-1+\text{HeavisideTheta}[-\pi+t])\,(2\,\text{Cos}[2t]-6\,\text{Sin}[2t])+e^{2t}\,(-1+\text{HeavisideTheta}[-\pi+t])\,(-12\,\text{Cos}[2t]-4\,\text{Sin}[2t])+4\,e^{2t}\,\text{DiracDelta}[-\pi+t]\,(3\,\text{Cos}[2t]+\text{Sin}[2t])+4\,e^{2t}\,(-1+\text{HeavisideTheta}[-\pi+t])\,(3\,\text{Cos}[2t]+\text{Sin}[2t])+5\,e^{2\pi}\,\text{DiracDelta}'[-\pi+t]-8\,e^{\pi+t}\,\text{DiracDelta}'[-\pi+t]+e^{2t}\,(3\,\text{Cos}[2t]+\text{Sin}[2t])\,\text{DiracDelta}'[-\pi+t]\right)==10\,\text{HeavisideTheta}[\pi-t]\,\text{Sin}[2t]$$

In[ ]:=
```
FullSimplify[ODECheck]
```

Out[ ]=

$$(-1+\text{HeavisideTheta}[\pi-t]+\text{HeavisideTheta}[-\pi+t])\,\text{Sin}[2t]==0$$

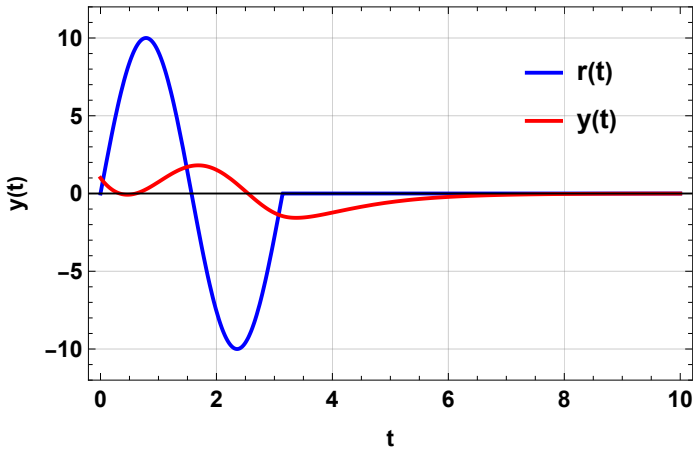
In[ ]:=
```
ICCheck = {ySoln[0] == y[0], ySoln'[0] == y'[0]} /. IC
```

Out[ ]=

```
{True, True}
```

◆ Now, let's take a look at the solution by plotting it:

```
In[•]:= Plot[{r[t], ySoln[t]}, {t, 0, 10},
 PlotRange → {-12, 12}, PlotStyle → {{Blue, Thick}, {Red, Thick}},
 Frame → True, FrameLabel → {"t", "y(t)"}, Exclusions → None,
 BaseStyle → {FontWeight → "Bold", Black, FontSize → 12}, GridLines → Automatic,
 AxesStyle → Directive[RGBColor[0.`, 0.`, 0.`], AbsoluteThickness[1]],
 Method → {"DefaultBoundaryStyle" → Automatic,
   "DefaultMeshStyle" → AbsolutePointSize[6], "ScalingFunctions" → None},
 PlotLegends → Placed[{"r(t)", "y(t)"}, {0.8, 0.75}], Background → White]
```



Convolution

◆ **According to the textbook**,

$\mathcal{L}(f)\ \mathcal{L}(g)$ is the transform of the convolution of f and g, denoted by the standard notation $f * g$ and defined by the integral:

$$h(t) = (f * g)(t) = \int_0^t f(\tau)\, g(t-\tau)\, d\tau$$

◆ **According to the Wolfram Mathematica**,

▪ The convolution $(f * g)(y)$ of two functions $f(x)$ and $g(x)$ is given by $\int_{-\infty}^{+\infty} f(x)\, g(y-x)\, dx$.

▪ The multidimensional convolution is given by

$$\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \cdots \mathrm{f}(x_1,\ x_2,\ \ldots)\, g(y_1 - x_1,\ y_2 - x_2,\ \ldots)\, dx_1\, dx_2 \cdots.$$

◆ Convolution uses the **Convolve** command, which is somewhat tricky and requires a bit of explanation.

◆ The syntax is: **Convolve**[first function , second function , dummy variable , final variable]

In[ ]:=
```
? Convolve
```

Out[ ]=

Symbol

Convolve[f, g, x, y] gives the convolution with respect to x of the expressions f and g.

Convolve[f, g, {x_1, x_2, ...}, {y_1, y_2, ...}] gives the multidimensional convolution.

- Convolution is defined in Wolfram Mathematica as an integral from -∞ to +∞, which is consistent with its use in signal processing.
- Many textbooks define convolution as an integral from 0 to t. The Heaviside fun will be required in order to input functions into the Convolve command.

In[ ]:=
```
ClearAll["Global`*"]
```

In[ ]:=
```
Convolve[Sin[τ] * UnitStep[τ], Cos[τ] * UnitStep[τ], τ, t]
```

Out[ ]=

$$\frac{1}{2} t \, \text{Sin}[t] \, \text{UnitStep}[t]$$

- Alternatively, Mathematica can be used to evaluate the integral directly:

In[ ]:=
```
Integrate[Sin[τ] Cos[t - τ], {τ, 0, t}]
```

Out[ ]=

$$\frac{1}{2} t \, \text{Sin}[t]$$

- The only **difference** between the two is the presence of the Heaviside function multiplied onto the result. **However**, that is fully consistent with the ***limits*** on the convolution integral.

In[ ]:=
```
Convolve[τ * UnitStep[τ], 1 * UnitStep[τ], τ, t]
```

Out[ ]=

$$\frac{1}{2} t^2 \, \text{UnitStep}[t]$$

In[ ]:=
```
Integrate[τ * 1, {τ, 0, t}]
```

Out[ ]=

$$\frac{t^2}{2}$$

- Another example:

In[ ]:=
```
convolve = Convolve[Sin[τ] * UnitStep[τ], Sin[τ] * UnitStep[τ], τ, t]
```

Out[ ]=

$$\frac{1}{2} \left(-t \, \text{Cos}[t] + \text{Sin}[t]\right) \text{UnitStep}[t]$$

```
In[ ]:= integral = Integrate[Sin[τ] Sin[t - τ], {τ, 0, t}]
```

```
Out[ ]=
```

$$\frac{1}{2}\,(-t\,\text{Cos}[t]+\text{Sin}[t])$$

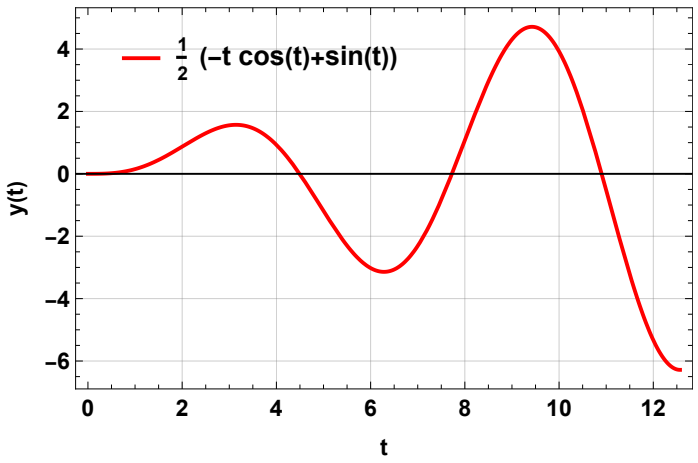
```
In[ ]:= convolve == integral * UnitStep[t]
```

```
Out[ ]= True
```

- Let's take a look at the graph:

```
In[ ]:= Plot[1/2 (-t Cos[t] + Sin[t]), {t, 0, 4 Pi}, PlotStyle → {{Red, Thick}},
  Frame → True, FrameLabel → {"t", "y(t)"}, Exclusions → None,
  BaseStyle → {FontWeight → "Bold", Black, FontSize → 12}, GridLines → Automatic,
  AxesStyle → Directive[RGBColor[0.`, 0.`, 0.`], AbsoluteThickness[1]],
  PlotLegends → Placed[{"1/2 (-t cos(t)+sin(t))"}, {0.3, 0.86}],
  Method → {"DefaultBoundaryStyle" → Automatic, "DefaultMeshStyle" → AbsolutePointSize[6],
    "ScalingFunctions" → None}, Background → White]
```

```
Out[ ]=
```



Summary

After completing this chapter, you should be able to

- develop SOPs to solve 1st-/2nd-order ODEs (IVPs) by the Laplace transform method
- perform Laplace and inverse Laplace transforms using Wolfram Mathematica
- use Mathematica to find the convolution of two functions.
- develop the habit of always checking your solutions for quality assurance.

Week 7: Series Solutions of ODEs

How to Use Series to Solve ODEs?

Table of Contents

Commands list

- Quit[]
- Series[f, {x, x_0, n}]
- Normal[*expr*]

- SeriesCoefficient[*series, n*]
- Sum[*expr, {n, n_{min}, n_{max}}*]
- SumConvergence[*f, n*]
- Factorial[*n*]
- Log[*z*]
- LogicalExpand[*expr*]
- Coefficient[*expr, form*]
- Table[*expr, n*]
- AsymptoticDSolveValue[*eqn, f, x → x_0*]

The Series Command in Wolfram Mathematica

To clear all definitions or to reclaim resources used by the kernel, you may want to quit the kernel by evaluating Quit. **Quit[].**

In[ ]:=
```
? Quit
```

Out[ ]=

Symbol

Quit[] terminates a Wolfram Language kernel session.

In[ ]:=
```
Quit[]
```

Use Series to make a power series out of a function. The first argument is the function. The second argument has the form *{var, pt, order}*, where *var* is the variable, *pt* is the point around which to expand, and *order* is the order:

In[ ]:=
```
? Series
```

Out[ ]=

Symbol

Series[*f*, {*x*, x_0, *n*}] generates a power series expansion

for *f* about the point $x = x_0$ to order $(x - x_0)^n$, where *n* is an explicit integer.

Series[*f*, $x \to x_0$] generates the leading term of a power series expansion for *f* about the point $x = x_0$.

Series[*f*, {*x*, x_0, n_x}, {*y*, y_0, n_y}, …] successively finds series expansions with respect to *x*, then *y*, etc.

- **Read more on** How to | Compute a Power Series
  https://en.wikipedia.org/wiki/Power_series
  https://reference.wolfram.com/language/ref/Series.html

- Power series for the **exponential** function around $x = 0$:

```
In[•]:= Series[Exp[x], {x, 0, 10}]
```

Out[•]=

$$1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \frac{x^6}{720} + \frac{x^7}{5040} + \frac{x^8}{40\,320} + \frac{x^9}{362\,880} + \frac{x^{10}}{3\,628\,800} + O[x]^{11}$$

- Power series for the function of $\frac{1}{e^x}$ around $x = 0$:

```
In[•]:= Series[1 / Exp[x], {x, 0, 10}]
```

Out[•]=

$$1 - x + \frac{x^2}{2} - \frac{x^3}{6} + \frac{x^4}{24} - \frac{x^5}{120} + \frac{x^6}{720} - \frac{x^7}{5040} + \frac{x^8}{40\,320} - \frac{x^9}{362\,880} + \frac{x^{10}}{3\,628\,800} + O[x]^{11}$$

- Power series for the function of $\frac{1}{x}$ around $x = 0$:

```
In[•]:= Series[1 / x, {x, 0, 10}]
```

Out[•]=

$$\frac{1}{x} + O[x]^{11}$$

- Power series for the function of **natural logarithm of x** around $x = 0$:

```
Series[Log[x], {x, 0, 10}] (**  Note that Log[x] gives the natural logarithm of x  **)
```

Out[•]=

$$\text{Log}[x] + O[x]^{11}$$

- Power series for the function of **cos(x)** around $x = 0$:

```
In[•]:= Series[Cos[x], {x, 0, 10}]
```

Out[•]=

$$1 - \frac{x^2}{2} + \frac{x^4}{24} - \frac{x^6}{720} + \frac{x^8}{40\,320} - \frac{x^{10}}{3\,628\,800} + O[x]^{11}$$

- Power series for the function of e^{ix} around $x = 0$:

```
In[•]:= Series[Exp[I * x], {x, 0, 10}]
```

Out[•]=

$$1 + i\,x - \frac{x^2}{2} - \frac{i\,x^3}{6} + \frac{x^4}{24} + \frac{i\,x^5}{120} - \frac{x^6}{720} - \frac{i\,x^7}{5040} + \frac{x^8}{40\,320} + \frac{i\,x^9}{362\,880} - \frac{x^{10}}{3\,628\,800} + O[x]^{11}$$

- We may find the derivatives of the power series using the **D[ ]** command.

In[ ]:=
```
? D
```

Out[ ]=

Symbol

D[*f*, *x*] gives the partial derivative $\partial f/\partial x$.

D[*f*, {*x*, *n*}] gives the multiple derivative $\partial^n f/\partial x^n$.

D[*f*, *x*, *y*, ...] gives the partial derivative $\cdots (\partial/\partial y)(\partial/\partial x) f$.

D[*f*, {*x*, *n*}, {*y*, *m*}, ...] gives the multiple partial derivative $\cdots (\partial^m/\partial y^m)(\partial^n/\partial x^n) f$.

D[*f*, {{x_1, x_2, ...}}] for a scalar *f* gives the vector derivative $(\partial f/\partial x_1, \partial f/\partial x_2, \ldots)$.

D[*f*, {*array*}] gives an array derivative.

In[ ]:=
```
D[D[Series[Exp[x], {x, 0, 10}], {x, 1}], {x, 1}]
```

Out[ ]=

$$1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \frac{x^6}{720} + \frac{x^7}{5040} + \frac{x^8}{40\,320} + O[x]^9$$

In[ ]:=
```
s1 = D[Series[Exp[x], {x, 0, 10}], {x, 1}]
```

Out[ ]=

$$1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \frac{x^6}{720} + \frac{x^7}{5040} + \frac{x^8}{40\,320} + \frac{x^9}{362\,880} + O[x]^{10}$$

- **Normal[ ]** turns the power series back into an ordinary polynomial expression.

In[ ]:=
```
Normal[s1]
```

Out[ ]=

$$1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \frac{x^6}{720} + \frac{x^7}{5040} + \frac{x^8}{40\,320} + \frac{x^9}{362\,880}$$

- We can find the coefficients of the terms in the particular power series by using the command **SeriesCoefficient[ ].**

In[ ]:=
```
Table[SeriesCoefficient[s1, n], {n, 0, 9}]
```

Out[ ]=

$$\left\{1, 1, \frac{1}{2}, \frac{1}{6}, \frac{1}{24}, \frac{1}{120}, \frac{1}{720}, \frac{1}{5040}, \frac{1}{40\,320}, \frac{1}{362\,880}\right\}$$

In[ ]:= `? SeriesCoefficient`

Out[ ]=

Symbol

SeriesCoefficient[*series*, *n*] finds the coefficient of

the n^{th}–order term in a power series in the form generated by Series.

SeriesCoefficient[*f*, {*x*, x_0, *n*}] finds the coefficient of $(x - x_0)^n$ in the expansion of *f* about the point $x = x_0$.

SeriesCoefficient[*f*, {*x*, x_0, n_x}, {*y*, y_0, n_y}, …] finds a coefficient in a multivariate series.

In[ ]:= `Series[Log[1 + x], {x, 0, 7}]`

Out[ ]=

$$x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} - \frac{x^6}{6} + \frac{x^7}{7} + 0[x]^8$$

- **Note:** when we do operations on a power series, the result is computed only to the appropriate order of x.

In[ ]:= `s1^2`

Out[ ]=

$$1 + 2x + 2x^2 + \frac{4x^3}{3} + \frac{2x^4}{3} + \frac{4x^5}{15} + \frac{4x^6}{45} + \frac{8x^7}{315} + \frac{2x^8}{315} + \frac{4x^9}{2835} + 0[x]^{10}$$

In[ ]:= `(Normal[s1])^2`

Out[ ]=

$$\left(1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \frac{x^6}{720} + \frac{x^7}{5040} + \frac{x^8}{40\,320} + \frac{x^9}{362\,880}\right)^2$$

Taylor and Maclaurin series

If f has derivatives of all orders at $x = a$, then the **Taylor series** for the function f at a is

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n + \cdots$$

The Taylor series for f at **0** is known as the **Maclaurin series** for f.

Read more on:
https://math.libretexts.org/Bookshelves/Calculus/Book%3_A_Calculus_(OpenStax)/10%3_A_Power_Series/10.3%3_A_Taylor_and_Maclaurin_Series

Example 7.1

Find the Taylor expansion of the given function around $x_0 = 1$ (up to 9 th order terms):

$$\sinh\left(3x^2-4\right)$$

```
In[•]:= Series[Sinh[3 x^2 - 4], {x, 1, 9}]
```

Out[•]=

$$-\text{Sinh}[1]+6\,\text{Cosh}[1]\,(x-1)+(3\,\text{Cosh}[1]-18\,\text{Sinh}[1])\,(x-1)^2+(36\,\text{Cosh}[1]-18\,\text{Sinh}[1])\,(x-1)^3+\left(54\,\text{Cosh}[1]-\frac{117\,\text{Sinh}[1]}{2}\right)(x-1)^4+\left(\frac{459\,\text{Cosh}[1]}{5}-108\,\text{Sinh}[1]\right)(x-1)^5+\left(\frac{333\,\text{Cosh}[1]}{2}-\frac{729\,\text{Sinh}[1]}{5}\right)(x-1)^6+\left(\frac{7614\,\text{Cosh}[1]}{35}-\frac{1107\,\text{Sinh}[1]}{5}\right)(x-1)^7+\left(\frac{1377\,\text{Cosh}[1]}{5}-\frac{80\,649\,\text{Sinh}[1]}{280}\right)(x-1)^8+\left(\frac{47\,547\,\text{Cosh}[1]}{140}-\frac{11\,502\,\text{Sinh}[1]}{35}\right)(x-1)^9+O[x-1]^{10}$$

Example 7.2

Find the Maclaurin expansion of the given function (up to 15 th order terms):

$$\log\left(2x^3+5\right)$$

```
Series[Log[2 x^3 + 5], {x, 0, 15}] (**   Maclaurin series means that x0=0   **)
```

$$\text{Log}[5]+\frac{2x^3}{5}-\frac{2x^6}{25}+\frac{8x^9}{375}-\frac{4x^{12}}{625}+\frac{32x^{15}}{15\,625}+O[x]^{16}$$

Basic Concepts

What do we mean by "Convergent vs. Divergent Series" ?

Convergent Series: A series is said to be convergent **if it approaches some limit** (D'Angelo and West 2000, p. 259).

Divergent Series: A series which is not convergent.

Read more on: https://en.wikipedia.org/wiki/Convergent_series

- **Sum**[expr, {n, nmin, nmax}] finds the sum of expr as n goes from $n_{\min}$ to $n_{\max}$.

```
In[•]:= Sum[x^n / n!, {n, 0, Infinity}]
```

Out[•]=

$$e^x$$

```
In[•]:= Sum[x^n / (n!)^2, {n, 0, Infinity}]
```

Out[•]=

$$\text{BesselI}\left[0,\,2\sqrt{x}\right]$$

In[ ]:=
```
? BesselI
```

Out[ ]=

Symbol

Bessel[n, z] gives the modified Bessel function of the first kind $I_n(z)$.

In[ ]:=
```
Sum[(n !) * x^n / (2 n) !, {n, 0, Infinity}]
```

Out[ ]=

$$\frac{1}{2}\left(2 + e^{x/4}\sqrt{\pi}\ \sqrt{x}\ \text{Erf}\left[\frac{\sqrt{x}}{2}\right]\right)$$

In[ ]:=
```
Sum[(n !) * x^n / (2 n) !, {n, 1, Infinity}]
```

Out[ ]=

$$\frac{1}{2}\, e^{x/4}\sqrt{\pi}\ \sqrt{x}\ \text{Erf}\left[\frac{\sqrt{x}}{2}\right]$$

In[ ]:=
```
Sum[1 / n, {n, 1, Infinity}]
```

Sum: Sum does not converge.

Out[ ]=

$$\sum_{n=1}^{\infty} \frac{1}{n}$$

- We can also use a built-in function **SumConvergence** to find out if the series is convergent or divergent.

In[ ]:=
```
? SumConvergence
```

Out[ ]=

Symbol

SumConvergence[f, n] gives conditions for the sum $\sum_n^{\infty} f$ to be convergent.

SumConvergence[f, {n_1, n_2, ...}] gives conditions for the multiple sum $\sum_{n_1}^{\infty}\sum_{n_2}^{\infty}\ldots f$ to be convergent.

Example 7.3

Test for convergence of the sum:

$$\sum_n^{\infty} \frac{1}{n}$$

In[ ]:=
```
SumConvergence[1 / n, n]
```

Out[ ]=
```
False
```

Example 7.4

Test for convergence of the sum:

$$\sum_n^{\infty} \frac{3^n\, n^2}{n!}$$

```
In[ ]:= SumConvergence[3^n * n^2 / n!, n]
Out[ ]=
True
```

Example 7.5

Test for convergence of the sum:

$$\sum_n^{\infty} \frac{1}{n!}$$

```
In[ ]:= SumConvergence[1 / Factorial[n], n]
True
```

What do we mean by "Analytic at point" ?

An **analytic function** is a function that is locally given by a convergent power series.

Any function is said to be analytic at a point a if it can be represented by a power series in x-a.

For example, functions such as e^x, $\sin x$, $\log x$ can be represented by Taylor series, thus these functions are analytic.

If function is analytic at point, it can be replaced either by Taylor Series or Maclaurin series, which are both power series.

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}\,(x-a)^n = f(a) + f'(a)\,(x-a) + \frac{f''(a)}{2!}\,(x-a)^2 + \cdots$$

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!}\,(x)^n = f(0) + f'(0)\,x + \frac{f''(0)}{2!}\,x^2 + \cdots$$

```
Series[1 / x, {x, 0, 5}]
Out[ ]=
1/x + O[x]^6
```

- Not analytic at $x = 0$.

```
Series[Log[x], {x, 0, 5}]
Out[ ]=
Log[x] + O[x]^6
```

- Not analytic at $x = 0$.

```
Series[Log[1 + x], {x, 0, 5}]
```

Out[•]=

$$x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} + O[x]^6$$

- Analytic at $x = 0$.

Solving ODEs by the Power Series Method

Standard Operating Procedures (SOPs)

Step 1. Define the solution as a Power Series (with coefficients to be determined; the center of the series is usually taken to be at $x_0 = 0$).

$$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots = \sum_{m=0}^{\infty} a_m x^m$$

Step 2. Insert the power series of y and the power series of y', y'' obtained by term-wise differentiation in to the ODE.

$$y' = a_1 + 2 a_2 x + 3 a_3 x^2 + \cdots = \sum_{m=1}^{\infty} m a_m x^{m-1}$$

Collect the powers of x finding.

$$(a_1 - a_0) + (2 a_2 - a_1) x + (3 a_3 - a_2) x^2 + \cdots = 0$$

Step 3. Equating the coefficient of each power of x to zero, we have a system of equations of the coefficients, a_m.

$$a_1 - a_0 = 0, \qquad 2 a_2 - a_1 = 0, \qquad 3 a_3 - a_2 = 0, \ \cdots .$$

Step 4. Solving these equations, we may express a_1, a_2, ... in terms of a_0 (for the first-order ODEs) or a_2, a_3, ... in terms of a_0 and a_1 (for the second-order ODEs).

$$a_1 = a_0, \qquad a_2 = \frac{a_1}{2} = \frac{a_0}{2!}, \qquad a_3 = \frac{a_2}{3} = \frac{a_0}{3!}, \ \cdots .$$

Step 5. With these values of the coefficients, the series solution becomes the familiar general solution.

$$y = a_0 + a_0 x + \frac{a_0}{2!} x^2 + \frac{a_0}{3!} x^3 + \cdots = a_0 \left(1 + x + \frac{x^2}{2!} + \frac{x^3}{3!}\right) = a_0 e^x$$

Example 7.6

Find the general solution to the given ODE:

$y' - y = 0$

In[ ]:=
```
ClearAll["Global`*"]
```

- **Step 1.** Define the solution as a Power Series. Here, we omit the terms of $p + 1$. The value of p (max value) can be varied.

In[ ]:=
```
p = 8; y = Sum[c[i] x^i, {i, 0, p}] + O[x]^(p + 1)
```

Out[ ]=

$$c[0] + c[1]\, x + c[2]\, x^2 + c[3]\, x^3 + c[4]\, x^4 + c[5]\, x^5 + c[6]\, x^6 + c[7]\, x^7 + c[8]\, x^8 + O[x]^9$$

- **Step 2.** Insert the power series solution (with undetermined coefficients) to the given ODE.

In[ ]:=
```
de = D[y, x] - y == 0
```

Out[ ]=

$$(-c[0] + c[1]) + (-c[1] + 2\,c[2])\, x + (-c[2] + 3\,c[3])\, x^2 + (-c[3] + 4\,c[4])\, x^3 + (-c[4] + 5\,c[5])\, x^4 + (-c[5] + 6\,c[6])\, x^5 + (-c[6] + 7\,c[7])\, x^6 + (-c[7] + 8\,c[8])\, x^7 + O[x]^8 == 0$$

- **Step 3.** Use **LogicalExpand[]** to generate a sequence of equations for each power of x.

In[ ]:=
```
coeffeqns = LogicalExpand[de]
```

Out[ ]=

$$-c[0] + c[1] == 0 \&\& -c[1] + 2\,c[2] == 0 \&\& -c[2] + 3\,c[3] == 0 \&\& -c[3] + 4\,c[4] == 0 \&\& -c[4] + 5\,c[5] == 0 \&\& -c[5] + 6\,c[6] == 0 \&\& -c[6] + 7\,c[7] == 0 \&\& -c[7] + 8\,c[8] == 0$$

In[ ]:=
```
? LogicalExpand
```

Out[ ]=

Symbol

LogicalExpand[*expr*] expands out logical combinations of equations, inequalities, and other functions.

- **Step 4.** Solve the equations for the coefficients a[i]. We can also feed equations involving power series directly to **Solve[]**:

In[ ]:=
```
solvedcoeffs = Solve[coeffeqns, Table[c[i], {i, 1, 8}]]
```

Out[ ]=

$$\left\{\left\{c[1] \to c[0],\ c[2] \to \frac{c[0]}{2},\ c[3] \to \frac{c[0]}{6},\ c[4] \to \frac{c[0]}{24},\ c[5] \to \frac{c[0]}{120},\ c[6] \to \frac{c[0]}{720},\ c[7] \to \frac{c[0]}{5040},\ c[8] \to \frac{c[0]}{40\,320}\right\}\right\}$$

- **Step 4.** Substitute the obtained coefficients to get our solution.

In[ ]:=
```
y = y /. solvedcoeffs
```

Out[ ]=

$$\left\{c[0] + c[0]\,x + \frac{1}{2}c[0]\,x^2 + \frac{1}{6}c[0]\,x^3 + \frac{1}{24}c[0]\,x^4 + \frac{1}{120}c[0]\,x^5 + \frac{1}{720}c[0]\,x^6 + \frac{c[0]\,x^7}{5040} + \frac{c[0]\,x^8}{40\,320} + O[x]^9\right\}$$

In[ ]:=
```
Coefficient[y, c[0]]
```

Out[ ]=

$$\left\{1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \frac{x^6}{720} + \frac{x^7}{5040} + \frac{x^8}{40\,320}\right\}$$

- Summation of Series: The Wolfram System recognizes this as the power series expansion of **exp(*x*)**.

In[ ]:=
```
Sum[x^n / n!, {n, 0, Infinity}]
```

Out[ ]=

$$e^x$$

In[ ]:=
```
Series[Exp[x], {x, 0, 8}]
```

Out[ ]=

$$1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \frac{x^6}{720} + \frac{x^7}{5040} + \frac{x^8}{40\,320} + O[x]^9$$

- Thus we have obtained the familiar solution of $y = c_0\, e^x$.
- **Step 5.** Verify the solution.

In[ ]:=
```
D[c0 * Exp[x], x] - c0 * Exp[x] == 0
```

Out[ ]=
```
True
```

Example 7.7

Find the general solution to the given ODE:

$$y'' + y = 0$$

In[ ]:=
```
ClearAll["Global`*"]
```

- **Step 1.** Define the solution as a Power Series. Here, we omit the terms of $p + 1$. The value of p (max value) can be varied.

In[ ]:=
```
p = 9; y = Sum[c[i] x^i, {i, 0, p}] + O[x]^(p + 1)
```

Out[ ]=

$$c[0] + c[1]\,x + c[2]\,x^2 + c[3]\,x^3 + c[4]\,x^4 + c[5]\,x^5 + c[6]\,x^6 + c[7]\,x^7 + c[8]\,x^8 + c[9]\,x^9 + O[x]^{10}$$

- **Step 2.** Insert the power series solution (with undetermined coefficients) to the given ODE.

In[•]:=
```
de = D[y, {x, 2}] + y == 0
```

Out[•]=

$$(c[0] + 2\,c[2]) + (c[1] + 6\,c[3])\,x + (c[2] + 12\,c[4])\,x^2 + (c[3] + 20\,c[5])\,x^3 + (c[4] + 30\,c[6])\,x^4 + (c[5] + 42\,c[7])\,x^5 + (c[6] + 56\,c[8])\,x^6 + (c[7] + 72\,c[9])\,x^7 + O[x]^8 == 0$$

- **Step 3.** Use **LogicalExpand[]** to generate a sequence of equations for each power of ***x***.

In[•]:=
```
coeffeqns = LogicalExpand[de]
```

Out[•]=

$$c[0] + 2\,c[2] == 0 \&\& c[1] + 6\,c[3] == 0 \&\& c[2] + 12\,c[4] == 0 \&\& c[3] + 20\,c[5] == 0 \&\& c[4] + 30\,c[6] == 0 \&\& c[5] + 42\,c[7] == 0 \&\& c[6] + 56\,c[8] == 0 \&\& c[7] + 72\,c[9] == 0$$

- **Step 4.** Solve the equations for the coefficients ***a*[i]**. We can also feed equations involving power series directly to **Solve[]**:

In[•]:=
```
solvedcoeffs = Solve[coeffeqns, Table[c[i], {i, 1, 10}]]
```

Solve: Equations may not give solutions for all "solve" variables.

Out[•]=

$$\left\{\left\{c[2] \to -\frac{c[0]}{2}, c[3] \to -\frac{c[1]}{6}, c[4] \to \frac{c[0]}{24}, c[5] \to \frac{c[1]}{120}, c[6] \to -\frac{c[0]}{720}, c[7] \to -\frac{c[1]}{5040}, c[8] \to \frac{c[0]}{40\,320}, c[9] \to \frac{c[1]}{362\,880}\right\}\right\}$$

- **Step 5.** Substitute the obtained coefficients to get our solution.

In[•]:=
```
y = y /. solvedcoeffs
```

Out[•]=

$$\left\{\left\{c[0] + c[1]\,x - \frac{1}{2}c[0]\,x^2 - \frac{1}{6}c[1]\,x^3 + \frac{1}{24}c[0]\,x^4 + \frac{1}{120}c[1]\,x^5 - \frac{1}{720}c[0]\,x^6 - \frac{c[1]\,x^7}{5040} + \frac{c[0]\,x^8}{40\,320} + \frac{c[1]\,x^9}{362\,880} + O[x]^{10}\right\}\right\}$$

In[•]:=
```
Coefficient[y, c[0]]
```

Out[•]=

$$\left\{\left\{1 - \frac{x^2}{2} + \frac{x^4}{24} - \frac{x^6}{720} + \frac{x^8}{40\,320}\right\}\right\}$$

In[•]:=
```
Series[Cos[x], {x, 0, 10}]
```

Out[•]=

$$1 - \frac{x^2}{2} + \frac{x^4}{24} - \frac{x^6}{720} + \frac{x^8}{40\,320} - \frac{x^{10}}{3\,628\,800} + O[x]^{11}$$

- Expressing the coefficients in terms of the arbitrary **c[0]**, we get the solution of $y = c_0 \cos(x)$.

In[ ]:=
```
Coefficient[y, c[1]]
```
Out[ ]=
$$\left\{\left\{x-\frac{x^3}{6}+\frac{x^5}{120}-\frac{x^7}{5040}+\frac{x^9}{362\,880}\right\}\right\}$$

In[ ]:=
```
Series[Sin[x], {x, 0, 10}]
```
Out[ ]=
$$x-\frac{x^3}{6}+\frac{x^5}{120}-\frac{x^7}{5040}+\frac{x^9}{362\,880}+O[x]^{11}$$

- Expressing the coefficients in terms of the arbitrary **c[1]**, we get the solution $y = c_1 \sin(x)$.

In[ ]:=
```
ysoln = Coefficient[y, c[0]] + Coefficient[y, c[1]]
```
Out[ ]=
$$\left\{\left\{1+x-\frac{x^2}{2}-\frac{x^3}{6}+\frac{x^4}{24}+\frac{x^5}{120}-\frac{x^6}{720}-\frac{x^7}{5040}+\frac{x^8}{40\,320}+\frac{x^9}{362\,880}\right\}\right\}$$

- Thus the general solution is $y = c_0 \cos(x) + c_1 \sin(x)$.

In[ ]:=
```
Series[Cos[x] + Sin[x], {x, 0, 9}]
```
Out[ ]=
$$1+x-\frac{x^2}{2}-\frac{x^3}{6}+\frac{x^4}{24}+\frac{x^5}{120}-\frac{x^6}{720}-\frac{x^7}{5040}+\frac{x^8}{40\,320}+\frac{x^9}{362\,880}+O[x]^{10}$$

- **Step 5.** Verify the solution.

In[ ]:=
```
D[c0 * Cos[x] + c1 * Sin[x], {x, 2}] + c0 * Cos[x] + c1 * Sin[x] == 0
```
Out[ ]=
```
True
```

Example 7.8

Find the general solution to the given ODE:

$$(y')^2 - y = x$$

In[ ]:=
```
ClearAll["Global`*"]
```

- **Step 1.** Define the solution as a Power Series.

In[ ]:=
```
y = Sum[c[i] x^i, {i, 0, 8}] + O[x]^9
```
Out[ ]=
$$c[0]+c[1]\,x+c[2]\,x^2+c[3]\,x^3+c[4]\,x^4+c[5]\,x^5+c[6]\,x^6+c[7]\,x^7+c[8]\,x^8+O[x]^9$$

- **Step 2.** Insert the power series solution (with undetermined coefficients) to the given ODE.

In[•]:=
```
de = D[y, x]^2 - y == x
```

Out[•]=

$$\left(-c[0]+c[1]^2\right)+\left(-c[1]+4\,c[1]\times c[2]\right)x+\left(-c[2]+4\,c[2]^2+6\,c[1]\times c[3]\right)x^2+$$
$$\left(-c[3]+12\,c[2]\times c[3]+8\,c[1]\times c[4]\right)x^3+\left(9\,c[3]^2-c[4]+16\,c[2]\times c[4]+10\,c[1]\times c[5]\right)x^4+$$
$$\left(24\,c[3]\times c[4]-c[5]+20\,c[2]\times c[5]+12\,c[1]\times c[6]\right)x^5+$$
$$\left(16\,c[4]^2+30\,c[3]\times c[5]-c[6]+24\,c[2]\times c[6]+14\,c[1]\times c[7]\right)x^6+$$
$$\left(40\,c[4]\times c[5]+36\,c[3]\times c[6]-c[7]+28\,c[2]\times c[7]+16\,c[1]\times c[8]\right)x^7+O[x]^8==x$$

- **Step 3.** Use **LogicalExpand[]** to generate a sequence of equations for each power of ***x***.

In[•]:=
```
coeffeqns = LogicalExpand[de]
```

Out[•]=

$$-c[0]+c[1]^2==0\;\&\&\;-1-c[1]+4\,c[1]\times c[2]==0\;\&\&\;-c[2]+4\,c[2]^2+6\,c[1]\times c[3]==0\;\&\&$$
$$-c[3]+12\,c[2]\times c[3]+8\,c[1]\times c[4]==0\;\&\&\;9\,c[3]^2-c[4]+16\,c[2]\times c[4]+10\,c[1]\times c[5]==0\;\&\&$$
$$24\,c[3]\times c[4]-c[5]+20\,c[2]\times c[5]+12\,c[1]\times c[6]==0\;\&\&$$
$$16\,c[4]^2+30\,c[3]\times c[5]-c[6]+24\,c[2]\times c[6]+14\,c[1]\times c[7]==0\;\&\&$$
$$40\,c[4]\times c[5]+36\,c[3]\times c[6]-c[7]+28\,c[2]\times c[7]+16\,c[1]\times c[8]==0$$

- **Step 4.** Solve the equations for the coefficients ***a*[i]**. We can also feed equations involving power series directly to **Solve[]**:

In[•]:=
```
c[0] = 1; solvedcoeffs = Solve[coeffeqns, Table[c[i], {i, 1, 8}]]
```

Out[•]=

$$\left\{\{c[1]\to -1,\,c[2]\to 0,\,c[3]\to 0,\,c[4]\to 0,\,c[5]\to 0,\,c[6]\to 0,\,c[7]\to 0,\,c[8]\to 0\},\right.$$
$$\left\{c[1]\to 1,\,c[2]\to \frac{1}{2},\,c[3]\to -\frac{1}{12},\,c[4]\to \frac{5}{96},\right.$$
$$\left.\left.c[5]\to -\frac{41}{960},\,c[6]\to \frac{469}{11\,520},\,c[7]\to -\frac{6889}{161\,280},\,c[8]\to \frac{24\,721}{516\,096}\right\}\right\}$$

- **Step 5.** Substitute the obtained coefficients to get our solution.

In[•]:=
```
y = y /. solvedcoeffs
```

Out[•]=

$$\left\{1-x+O[x]^9,\;1+x+\frac{x^2}{2}-\frac{x^3}{12}+\frac{5\,x^4}{96}-\frac{41\,x^5}{960}+\frac{469\,x^6}{11\,520}-\frac{6889\,x^7}{161\,280}+\frac{24\,721\,x^8}{516\,096}+O[x]^9\right\}$$

In[•]:=
```
(D[1 - x, x])^2 - (1 - x) == x
```

Out[•]=
```
True
```

In[•]:=
```
(D[1 + x + x^2/2 - x^3/12 + 5 x^4/96 - 41 x^5/960 + O[x]^6, x])^2 - (1 + x + x^2/2 - x^3/12 + 5 x^4/96 - 41 x^5/960 + O[x]^6) == x
```

Out[•]=

$$x+O[x]^5==x$$

Different Approach Using a Function Embedded in Wolfram Mathematica

In[ ]:=
```
? AsymptoticDSolveValue
```

Out[ ]=

Symbol

AsymptoticDSolveValue[eqn, f, $x \to x_0$] computes an asymptotic approximation to the differential equation eqn for $f[x]$ centered at x_0.

AsymptoticDSolveValue[{eqn_1, eqn_2, …}, {f_1, f_2, …}, $x \to x_0$] computes an asymptotic approximation to a system of differential equations.

AsymptoticDSolveValue[eqn, f, x, $\epsilon \to \epsilon_0$] computes an asymptotic approximation of $f[x, \epsilon]$ for the parameter ϵ centered at ϵ_0.

AsymptoticDSolveValue[eqn, f, …, {ξ, ξ_0, n}] computes the asymptotic approximation to order n.

The same ODE as in the Example 7.2 with the corresponding initial conditions:

$$y'' + y = 0 \qquad y(0) = 1,\ y'(0) = 0$$

In[ ]:=
```
ClearAll["Global`*"]
```

In[ ]:=
```
sol1 = AsymptoticDSolveValue[{y''[x] + y[x] == 0, y[0] == 1, y'[0] == 0}, y[x], {x, 0, 8}]
```

Out[ ]=

$$1 - \frac{x^2}{2} + \frac{x^4}{24} - \frac{x^6}{720} + \frac{x^8}{40\,320}$$

In[ ]:=
```
sol2 = AsymptoticDSolveValue[{y''[x] + y[x] == 0, y[0] == 1, y'[0] == 0}, y[x], {x, 0, 16}]
```

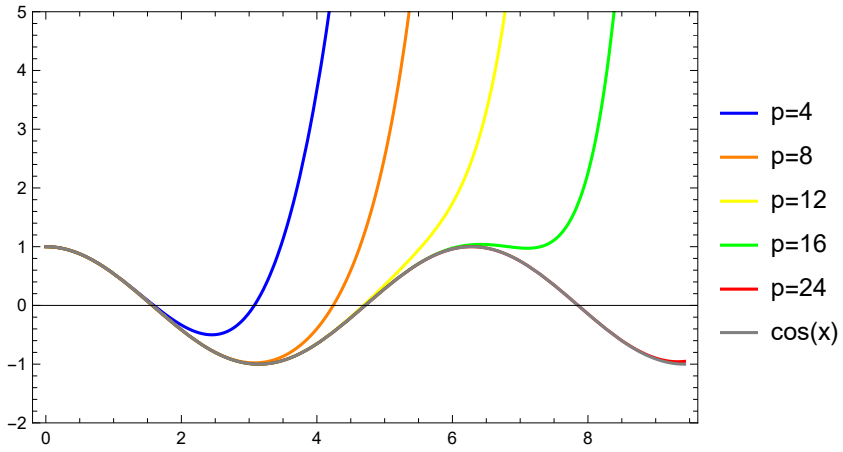
Out[ ]=

$$1 - \frac{x^2}{2} + \frac{x^4}{24} - \frac{x^6}{720} + \frac{x^8}{40\,320} - \frac{x^{10}}{3\,628\,800} + \frac{x^{12}}{479\,001\,600} - \frac{x^{14}}{87\,178\,291\,200} + \frac{x^{16}}{20\,922\,789\,888\,000}$$

- Asymptotic approximation by varying the order n.

In[ ]:=
```
sol[n_] := AsymptoticDSolveValue[{y''[x] + y[x] == 0, y[0] == 1, y'[0] == 0}, y[x], {x, 0, n}]
```

```
In[•]:= Plot[{sol[4], sol[8], sol[12], sol[16], sol[24], Cos[x]} // Evaluate,
 {x, 0, 3 Pi}, PlotRange → {-2, 5}, Frame → True,
 PlotLegends → {"p=4", "p=8", "p=12", "p=16", "p=24", "cos(x)"},
 PlotStyle → {Blue, Orange, Yellow, Green, Red, Gray}]
```



Extended Power Series Method: Frobenius Method

Let $b(x)$ and $c(x)$ be any functions that are analytic at $x = 0$. Then the ODE

$$y'' + \frac{b(x)}{x}\, y' + \frac{c(x)}{x^2}\, y = 0$$

has at least one solution that can be represented in the form

$$y(x) = x^r \sum_{m=0}^{\infty} a_m\, x^m = x^r \left(a_0 + a_1\, x + a_2\, x^2 + \cdots\right)$$

where the exponent r may be any (real or complex) number (and r is chosen so that $a_0 \neq 0$).

Frobenius Method. Standard Operating Procedures (SOPs)

Step 1. Rewrite the ODE in the form of $x^2\, y'' + x\, b(x)\, y' + c(x)\, y = 0$. Find $b(x)$ and $c(x)$

$$x^2\, y'' + x b(x)\, y' + c(x)\, y = 0$$

Step 2. Expand $b(x)$ and $c(x)$ in power series. To apply the Frobenius Method, $b(x)$ and $c(x)$ must be analytic at $x = 0$. If $b(x)$ and $c(x)$ are polynomials we do nothing in this step. The purpose of this step is to obtain $b_0 = b(x = 0)$ and $c_0 = c(x = 0)$.

$$b(x) = b_0 + b_1\, x + b_2\, x^2 + \cdots, \qquad c(x) = c_0 + c_1\, x + c_2\, x^2 + \cdots$$

Step 3. Obtain the **indicial equation:** $r(r - 1) + b_0\, r + c_0 = 0$

$$r(r-1)+b_0\,r+c_0=0$$

Step 4. Solve the **indicial equation**, and obtain its roots r_1 and r_2. Depending on the values of r_1 and r_2, we have the following three cases:

(i) Distinct roots not differing by an integer;

(ii) Double root $r_1=r_2$;

(iii) Roots differing by an integer.

Case 1. ***Distinct Roots Not Differing by an Integer.*** A basis is

$$y_1(x)=x^{r_1}\left(a_0+a_1\,x+a_2\,x^2+\cdots\right)$$

and

$$y_2(x)=x^{r_2}\left(A_0+A_1\,x+A_2\,x^2+\cdots\right)$$

Case 2. ***Double Root*** $r_1=r_2=r$. A basis is

$$y_1(x)=x^{r}\left(a_0+a_1\,x+a_2\,x^2+\cdots\right)\qquad\left[r=\frac{1}{2}\,(1-b_0)\right]$$

(of the same general form as before) and

$$y_2(x)=y_1(x)\ln x+x^{r}\left(A_1\,x+A_2\,x^2+\cdots\right)\qquad(x>0)$$

Case 3. ***Roots Differing by an Integer.*** A basis is

$$y_1(x)=x^{r_1}\left(a_0+a_1\,x+a_2\,x^2+\cdots\right)$$

(of the same general form as before) and

$$y_2(x)=k\,y_1(x)\ln x+x^{r_2}\left(A_0+A_1\,x+A_2\,x^2+\cdots\right)$$

whre the roots are so denoted that $r_1-r_2>0$ and k may turn out to be zero.

Example 7.9

$$x(x-1)\,y''+(3\,x-1)\,y'+y=0$$

- **Step 1.** Rewrite the ODE in the form of $x^2\,y''+x\,b(x)\,y'+c(x)\,y=0$. Find $b(x)$ and $c(x)$.

$$y'' + \frac{(3x-1)}{x(x-1)}y' + \frac{1}{x(x-1)}y = 0 \implies x^2 y'' + x\frac{(3x-1)}{(x-1)}y' + \frac{x^2}{x(x-1)}y = 0 \implies b(x) = \frac{(3x-1)}{(x-1)}$$
$$c(x) = \frac{x^2}{x(x-1)}$$

- **Step 2.** Expand $b(x)$ and $c(x)$ in power series. To apply the Frobenius Method, $b(x)$ and $c(x)$ must be analytic at $x = 0$. If $b(x)$ and $c(x)$ are polynomials we do nothing in this step. The purpose of this step is to obtain $b_0 = b(x = 0)$ and $c_0 = c(x = 0)$.

```
In[•]:= Series[(3 x - 1)/(x - 1), {x, 0, 5}]
Out[•]=
1 - 2 x - 2 x^2 - 2 x^3 - 2 x^4 - 2 x^5 + O[x]^6
```

```
In[•]:= Series[x^2/(x (x - 1)), {x, 0, 5}]
Out[•]=
-x - x^2 - x^3 - x^4 - x^5 + O[x]^6
```

- **Step 3.** Obtain the **indicial equation.** With $b(0) = 1$ and $c(0) = 0$, we have an indicial equation $r(r-1) + r = 0$.

```
In[•]:= Solve[r (r - 1) + r == 0, r]
Out[•]=
{{r → 0}, {r → 0}}
```

- **Step 4.** Solving the **indicial equation**, and we obtained its roots $r_1 = 0$ and $r_2 = 0$. This corresponds to the Case (ii) with a double root.
- The indicial root:

```
In[•]:= ClearAll[a, r];
```

```
In[•]:= r = 0;
```

- The first $k+1$ terms of a proposed Frobenius series solution:

```
In[•]:= k = 6;
y = x^r (Sum[a[n] x^n, {n, 0, k}] + O[x]^(k + 1))
Out[•]=
a[0] + a[1] x + a[2] x^2 + a[3] x^3 + a[4] x^4 + a[5] x^5 + a[6] x^6 + O[x]^7
```

- Substitute this series into the given ODE: $x(x-1)y'' + (3x-1)y' + y = 0$.

```
In[•]:= deq =  x * (x - 1) * D[y, {x, 2}] + (3 x - 1) * D[y, {x, 1}] + y == 0
Out[•]=
(a[0] - a[1]) + (4 a[1] - 4 a[2]) x + (9 a[2] - 9 a[3]) x^2 +
  (16 a[3] - 16 a[4]) x^3 + (25 a[4] - 25 a[5]) x^4 + (36 a[5] - 36 a[6]) x^5 + O[x]^6 == 0
```

- Write the equations that the coefficients must satisfy.

```
In[•]:= coeffEqns = LogicalExpand[ deq ]
```

```
Out[•]=
a[0] - a[1] == 0 && 4 a[1] - 4 a[2] == 0 && 9 a[2] - 9 a[3] == 0 &&
 16 a[3] - 16 a[4] == 0 && 25 a[4] - 25 a[5] == 0 && 36 a[5] - 36 a[6] == 0
```

- Table listing the successive coefficients.

```
In[•]:= succCoeffs = Table[ a[n], {n, 1, 6} ]
```

```
Out[•]=
{a[1], a[2], a[3], a[4], a[5], a[6]}
```

- Solve for these coefficients in terms of **a[0]**.

```
In[•]:= ourCoeffs = Solve[coeffEqns, succCoeffs]
```

```
Out[•]=
{{a[1] → a[0], a[2] → a[0], a[3] → a[0], a[4] → a[0], a[5] → a[0], a[6] → a[0]}}
```

- Substitute these coefficients in original series to obtain desired particular solution.

```
In[•]:= y = y /. ourCoeffs
```

```
Out[•]=
{a[0] + a[0] x + a[0] x^2 + a[0] x^3 + a[0] x^4 + a[0] x^5 + a[0] x^6 + O[x]^7}
```

- Take the common factor **a[0]** out:

```
In[•]:= Coefficient[y, a[0]]
```

```
Out[•]=
{1 + x + x^2 + x^3 + x^4 + x^5 + x^6}
```

- Calculate the infinite sum of our series solution.

```
In[•]:= Sum[x^n, {n, 0, Infinity}]
```

```
Out[•]=
1/(1 - x)
```

- Verify the infinite sum of the series solution.

```
In[•]:= Series[1/(1 - x), {x, 0, k}]
```

```
Out[•]=
1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + O[x]^7
```

- Now we have obtained one solution to the given ODE, which is given by

$$y = a[0] \sum_{m=0}^{\infty} x^m = \frac{a[0]}{1-x}$$

- By choosing $a[0] = 1$, we have $y_1 = \frac{1}{1-x}$.
- We may get a second independent solution $y_2(x)$ by using two methods:

(1) following the Case (ii) rule of the Frobenius method;

(2) the method of reduction of order.

◆ Let's find the second independent solution $y_2(x)$.

◆ **Method 1.** Following the **Case (ii)** rule of the **Frobenius method.**

```
In[•]:= r = 0; k = 6;
        y = Log[x]/(1 - x) + x^r (Sum[ A[n] x^n, {n, 0, k} ] + O[x]^(k + 1))
```

Out[•]=

$$(A[0] + \text{Log}[x]) + (A[1] + \text{Log}[x])\, x + (A[2] + \text{Log}[x])\, x^2 + (A[3] + \text{Log}[x])\, x^3 + (A[4] + \text{Log}[x])\, x^4 + (A[5] + \text{Log}[x])\, x^5 + (A[6] + \text{Log}[x])\, x^6 + O[x]^7$$

```
In[•]:= deq =  x * (x - 1) D[y, {x, 2}] + (3 x - 1) D[y, {x, 1}] + y == 0
```

Out[•]=

$$(A[0] - A[1]) + (-3 + A[1] - 4\,A[2] - 3\,\text{Log}[x] + 3\,(1 + A[1] + \text{Log}[x]))\, x + (-3 + 3\,A[2] - 9\,A[3] - 6\,\text{Log}[x] + 3\,(1 + 2\,A[2] + 2\,\text{Log}[x]))\, x^2 + (-3 + 7\,A[3] - 16\,A[4] - 9\,\text{Log}[x] + 3\,(1 + 3\,A[3] + 3\,\text{Log}[x]))\, x^3 + (-3 + 13\,A[4] - 25\,A[5] - 12\,\text{Log}[x] + 3\,(1 + 4\,A[4] + 4\,\text{Log}[x]))\, x^4 + (-3 + 21\,A[5] - 36\,A[6] - 15\,\text{Log}[x] + 3\,(1 + 5\,A[5] + 5\,\text{Log}[x]))\, x^5 + O[x]^6 == 0$$

```
In[•]:= coeffEqns = LogicalExpand[ deq ]
```

Out[•]=

$$A[0] - A[1] == 0 \;\&\&\; -3 + A[1] - 4\,A[2] - 3\,\text{Log}[x] + 3\,(1 + A[1] + \text{Log}[x]) == 0 \;\&\&\; -3 + 3\,A[2] - 9\,A[3] - 6\,\text{Log}[x] + 3\,(1 + 2\,A[2] + 2\,\text{Log}[x]) == 0 \;\&\&\; -3 + 7\,A[3] - 16\,A[4] - 9\,\text{Log}[x] + 3\,(1 + 3\,A[3] + 3\,\text{Log}[x]) == 0 \;\&\&\; -3 + 13\,A[4] - 25\,A[5] - 12\,\text{Log}[x] + 3\,(1 + 4\,A[4] + 4\,\text{Log}[x]) == 0 \;\&\&\; -3 + 21\,A[5] - 36\,A[6] - 15\,\text{Log}[x] + 3\,(1 + 5\,A[5] + 5\,\text{Log}[x]) == 0$$

```
In[•]:= succCoeffs = Table[ A[n], {n, 1, 6} ]
```

Out[•]=

$$\{A[1], A[2], A[3], A[4], A[5], A[6]\}$$

```
In[•]:= ourCoeffs = Solve[coeffEqns, succCoeffs]
```

Out[•]=

$$\{\{A[1] \to A[0], A[2] \to A[0], A[3] \to A[0], A[4] \to A[0], A[5] \to A[0], A[6] \to A[0]\}\}$$

```
In[•]:= y = y /. ourCoeffs
```

Out[•]=

$$\{(A[0] + \text{Log}[x]) + (A[0] + \text{Log}[x])\, x + (A[0] + \text{Log}[x])\, x^2 + (A[0] + \text{Log}[x])\, x^3 + (A[0] + \text{Log}[x])\, x^4 + (A[0] + \text{Log}[x])\, x^5 + (A[0] + \text{Log}[x])\, x^6 + O[x]^7\}$$

```
y = y /. A[0] → 0 (* Note that Log[x] represents the natural logarithm of x*)
```

Out[•]=

$$\{\text{Log}[x] + \text{Log}[x]\, x + \text{Log}[x]\, x^2 + \text{Log}[x]\, x^3 + \text{Log}[x]\, x^4 + \text{Log}[x]\, x^5 + \text{Log}[x]\, x^6 + O[x]^7\}$$

◆ We can easily see that the second independent solution $y_2(x) = \dfrac{ln(x)}{1-x}$.

◆ Verify two solutions to the given ODE: $x(x-1)y'' + (3x-1)y' + y = 0$.

```
In[•]:= ClearAll[y];
        myODE = x (x - 1) * y''[x] + (3 x - 1) y'[x] + y[x] == 0
Out[•]=
        y[x] + (-1 + 3 x) y'[x] + (-1 + x) x y''[x] == 0

In[•]:= y1Soln[x_] = 1/(1 - x)
Out[•]=
        1/(1 - x)

In[•]:= ODECheck = myODE /. y → y1Soln
Out[•]=
        1/(1 - x) + 2 (-1 + x) x/(1 - x)^3 + (-1 + 3 x)/(1 - x)^2 == 0

In[•]:= FullSimplify[ODECheck]
Out[•]=
        True

In[•]:= y2Soln[x_] = Log[x]/(1 - x)
Out[•]=
        Log[x]/(1 - x)

In[•]:= ODECheck = myODE /. y → y2Soln
Out[•]=
        Log[x]/(1 - x) + (-1 + x) x (-1/((1 - x) x^2) + 2/((1 - x)^2 x) + 2 Log[x]/(1 - x)^3) + (-1 + 3 x) (1/((1 - x) x) + Log[x]/(1 - x)^2) == 0

In[•]:= FullSimplify[ODECheck]
Out[•]=
        True
```

◆ **Method 2. The method of Reduction of Order.**

```
In[•]:= ClearAll[y];
        myODE = x (x - 1) * y''[x] + (3 x - 1) y'[x] + y[x] == 0
Out[•]=
        y[x] + (-1 + 3 x) y'[x] + (-1 + x) x y''[x] == 0
```

◆ Substitute $y_2(x) = \frac{u(x)}{1-x}$, where the form of $u(x)$ is yet to be determined.

```
In[ ]:= yuSoln[x_] = u[x] / (1 - x)
```

```
Out[ ]=
u[x] / (1 - x)
```

```
In[ ]:= uODE = myODE /. y → yuSoln
```

```
Out[ ]=
u[x] / (1 - x) + (-1 + 3 x) (u[x] / (1 - x)^2 + u'[x] / (1 - x)) + (-1 + x) x (2 u[x] / (1 - x)^3 + 2 u'[x] / (1 - x)^2 + u''[x] / (1 - x)) == 0
```

```
In[ ]:= u2 = FullSimplify[uODE]
```

```
Out[ ]=
u'[x] + x u''[x] == 0
```

- Let's introduce a new variable t so that $t = u'$ and $t' = u''$.

```
In[ ]:= u2 /. {u' → t, u'' → t'}
```

```
Out[ ]=
t[x] + x t'[x] == 0
```

```
In[ ]:= DSolve[t[x] + x t'[x] == 0, t[x], x]
```

```
Out[ ]=
{{t[x] → C[1] / x}}
```

```
In[ ]:= DSolve[u'[x] == 1 / x, u[x], x]
```

```
Out[ ]=
{{u[x] → C[1] + Log[x]}}
```

- Thus we have $y_2(x) = \frac{u[x]}{1-x} = \frac{ln(x)}{1-x}$.
- $y_1(x) = \frac{1}{1-x}$ and $y_2(x) = \frac{ln(x)}{1-x}$ are linearly independent and thus form a basis of solutions of the given ODE.

```
In[ ]:= yGen = c1 * 1 / (1 - x) + c2 * Log[x] / (1 - x);
FullSimplify[x * (x - 1) * D[yGen, {x, 2}] + (3 x - 1) * D[yGen, x] + (yGen)] == 0
True
```

Example 7.10

$$(x^2 - x)y'' - xy' + y = 0$$

```
In[ ]:= ClearAll["Global`*"]
```

- **Step 1.** Rewrite the ODE in the form of $x^2 y'' + x\,b(x)\,y' + c(x)\,y = 0$. Find $b(x)$ and $c(x)$.

$$y'' + \frac{-x}{x(x-1)}\,y' + \frac{1}{x(x-1)}\,y = 0 \implies x^2\,y'' + x\,\frac{-x}{(x-1)}\,y' + \frac{x^2}{x(x-1)}\,y = 0 \implies b(x) = \frac{-x}{(x-1)}$$

$$c(x) = \frac{x^2}{x(x-1)}$$

$$b(0) = 0;\ c(0) = 0$$

- **Step 2.** Expand $b(x)$ and $c(x)$ in power series. To apply the Frobenius Method, $b(x)$ and $c(x)$ must be analytic at $x = 0$. $b(x)$ and $c(x)$ are already polynomials so we do nothing in this step.
- **Step 3.** Obtain the **indicial equation.** With $b(0) = 1$ and $c(0) = 0$, we have an indicial equation $r(r-1) = 0$.

```
In[ ]:= ClearAll[r]; Solve[r (r - 1) == 0, r]
Out[ ]=
{{r → 0}, {r → 1}}
```

- **Step 4.** Solving the **indicial equation**, and we obtained its roots $r_1 = 0$ and $r_2 = 1$. This corresponds to the Case (i) with two roots differing by an integer. **Notice** that in this case we need to set $r_1 > r_2$ in order to follow the recipe of the Frobenius method.
- For the first solution, we have $r = r_1 = 1$. Based on the recipe of the Frobenius method, we have:

```
In[ ]:= r = 1;
        k = 6;
        y = x^r (Sum[ a[n] x^n, {n, 0, k} ] + O[x]^(k + 1))
Out[ ]=
a[0] x + a[1] x^2 + a[2] x^3 + a[3] x^4 + a[4] x^5 + a[5] x^6 + a[6] x^7 + O[x]^8
```

- Substitute this series into the given ODE: $(x^2 - x)\,y'' - xy' + y = 0$.

```
In[ ]:= deq =  x * (x - 1) D[y, {x, 2}] - x * D[y, {x, 1}] + y == 0
Out[ ]=
-2 a[1] x + (a[1] - 6 a[2]) x^2 + (4 a[2] - 12 a[3]) x^3 +
  (9 a[3] - 20 a[4]) x^4 + (16 a[4] - 30 a[5]) x^5 + (25 a[5] - 42 a[6]) x^6 + O[x]^7 == 0
```

- Substitute this series into the given ODE: $(x^2 - x)\,y'' - xy' + y = 0$.
- Write the equations that the coefficients must satisfy:

```
In[ ]:= coeffEqns = LogicalExpand[ deq ]
Out[ ]=
-2 a[1] == 0 && a[1] - 6 a[2] == 0 && 4 a[2] - 12 a[3] == 0 &&
 9 a[3] - 20 a[4] == 0 && 16 a[4] - 30 a[5] == 0 && 25 a[5] - 42 a[6] == 0
```

- Table listing the successive coefficients:

```
In[•]:= succCoeffs = Table[a[n], {n, 1, 6}]
Out[•]=
{a[1], a[2], a[3], a[4], a[5], a[6]}
```

- Solve for these coefficients in terms of $a[0]$:

```
In[•]:= ourCoeffs = Solve[coeffEqns, succCoeffs]
Out[•]=
{{a[1] → 0, a[2] → 0, a[3] → 0, a[4] → 0, a[5] → 0, a[6] → 0}}
```

- Substitute these coefficients in original series to obtain desired particular solution:

```
In[•]:= y = y /. ourCoeffs
Out[•]=
{a[0] x + O[x]^8}
```

- By choosing $a[0] = 1$, we have $y_1(x) = x$.
- Let's check the result.

```
In[•]:= ClearAll[y];
myODE = x (x - 1) * y''[x] - x * y'[x] + y[x] == 0;
y1Soln[x_] = x;
ODECheck = myODE /. y → y1Soln;
FullSimplify[ODECheck]
Out[•]=
True
```

- Let's find the second independent solution $y_2(x)$ by the method of **Reduction of Order.**

```
In[•]:= yuSoln[x_] = u[x] * x;
uODE = FullSimplify[myODE /. y → yuSoln]
Out[•]=
x ((-2 + x) u'[x] + (-1 + x) x u''[x]) == 0
```

- Let's introduce a new variable t so that $t = u'$ and $t' = u''$.

```
In[•]:= tODE = FullSimplify[uODE /. {u' → t, u'' → t'}]
Out[•]=
x ((-2 + x) t[x] + (-1 + x) x t'[x]) == 0

In[•]:= DSolve[tODE, t[x], x]
Out[•]=
{{t[x] → (1 - x) C[1] / x^2}}
```

- Let's take $c_1 = -1$ a. So we have $t(x) = \frac{-(1-x)}{x^2} = u'(x)$, from which we can find $u[x]$.

In[ ]:=
```
DSolve[u'[x] == -(1 - x) / x^2, u[x], x]
```

Out[ ]=

$$\left\{\left\{u[x] \to \frac{1}{x} + c_1 + \text{Log}[x]\right\}\right\}$$

- Take $c_1 = 0$ a. So we have $u(x) = \frac{1}{x} + ln(x)$, and

$$y_2(x) = u(x)\, y_1(x) = \left(\frac{1}{x} + ln(x)\right) x = 1 + x\, ln(x)$$

- Let's check the second solution.

In[ ]:=
```
ClearAll[y];
myODE = x (x - 1) * y''[x] - x * y'[x] + y[x] == 0;
y2Soln[x_] = x * Log[x] + 1;
ODECheck = myODE /. y → y2Soln;
FullSimplify[ODECheck]
```

Out[ ]=
```
True
```

- $y_1(x) = x$ and $y_2(x) = 1 + x\, ln(x)$ are linearly independent and $y_2(x)$ has a logarithmic term, thus they constitute a basis of solutions for positive x.

In[ ]:=
```
yGen = c1 * x + c2 * (1 + x * Log[x]);
FullSimplify[(x^2 - x) * D[yGen, {x, 2}] - x * D[yGen, x] + yGen] == 0
```

```
True
```

Summary

After completing this chapter, you should be able to

- use Mathematica to find the power series representation of a function.
- recognise and work with some higher transcendental functions of mathematics.
- develop SOPs to solve ODEs by the power series method.
- develop SOPs to solve ODEs by the Frobenius method.
- develop the habit of always checking your solutions for quality assurance.

Week 8: Systems of Linear Equations

How to solve systems of linear equations?

Table of Contents

Commands list

- Column
- Solve
- MatrixForm
- FullSimplify
- LinearSolve
- ArrayFlatten
- Normal
- CoefficientArrays
- RowReduce

Prerequisite: How to Get Parts of a Matrix

The Wolfram Language has many matrix operations that support operations such as building, computing, and visualizing matrices. It also has a rich language for picking out and extracting parts of matrices.

How to | Get Parts of a Matrix:
https://reference.wolfram.com/language/howto/GetPartsOfAMatrix.html

Example 8.1: The Solve Command

Definition 8.1: System of Linear Equations

A **system of linear equations** is a collection of equations of the form

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots + a_{3n}x_n &= b_3 \\ \vdots \quad\quad & \quad\quad \vdots \quad\quad \vdots \quad\quad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \cdots + a_{mn}x_n &= b_m \end{aligned}$$

Definition 8.2: Consistent and Inconsistent Linear System

If a linear system has at least one solution, then we say that it is **consistent**. If not, **inconsistent**.

Find all solutions to the consistent system of linear equations using the **Solve** command:

$$\begin{aligned} x_1 + 2x_2 - x_3 + 3x_5 &= 7 \\ x_2 - 4x_3 + x_5 &= -2 \\ x_4 - 2x_5 &= 1 \end{aligned}$$

- **Step 1.** First, set up the system of equations:

In[ ]:=
```
ClearAll["Global`*"]
sys = {x1 + 2 x2 - x3 + 3 x5 == 7, x2 - 4 x3 + x5 == -2, x4 - 2 x5 == 1};
Column[sys]
```

Out[ ]=
```
x1 + 2 x2 - x3 + 3 x5 == 7
x2 - 4 x3 + x5 == -2
x4 - 2 x5 == 1
```

- **Step 2.** Set leading and free variable. In this system, x_1, x_2, x_4 are leading variables and x_3, x_5 are free variables. Therefore,

In[ ]:=
```
x3 = s1;
x5 = s2;
```

- **Step 3.** Looking at the given system of equations, it is apparent that the easiest way to start is at the bottom. Therefore, apply **back substitution method**.

In[ ]:=
```
? Solve
```

Out[ ]=

Symbol

Solve[*expr*, *vars*] attempts to solve the system *expr* of equations or inequalities for the variables *vars*.

Solve[*expr*, *vars*, *dom*] solves over the domain *dom*. Common choices of *dom* are Reals, Integers, and Complexes.

- Substituting x_5 into the bottom equation and solving it for x_4 gives:

In[ ]:=
```
Solve[x4 - 2 x5 == 1, x4]
```

Out[ ]=
```
{{x4 → 1 + 2 s2}}
```

- So, $x_4 = 1 + 2\,s_2$. Solving the next equation up for x_2 by substituting values for x_3 and x_5 gives:

In[ ]:=
```
x4 = 1 + 2 s2;
Solve[ x2 - 4 x3 + x5 == -2, x2]
```

Out[ ]=
```
{{x2 → -2 + 4 s1 - s2}}
```

- So, $x_2 = 2 + 4\,s_1 - s_2$. Finally, substituting x_2, x_3 and x_5 into the top equation gives:

In[ ]:=
```
x2 = -2 + 4 s1 - s2;
Solve[x1 + 2 x2 - x3 + 3 x5 == 7, x1]
```

Out[ ]=
```
{{x1 → 11 - 7 s1 - s2}}
```

- So, $x_1 = 11 - 7\,s_1 - s_2$.
- **Step 4.** Verify the solution.

In[ ]:=
```
Clear[x1, x2, x4]
soln = Solve[sys, {x1, x2, x4}]
```

Out[ ]=
```
{{x1 → 11 - 7 s1 - s2, x2 → -2 + 4 s1 - s2, x4 → 1 + 2 s2}}
```

In[ ]:=
```
FullSimplify[sys /. soln[[1]]]
```

Out[ ]=
```
{True, True, True}
```

- Hence, the general solution for given system of linear equations is:

$$x_1 = 11 - 7\,s_1 - s_2$$
$$x_2 = 2 + 4\,s_1 - s_2$$
$$x_3 = s_1$$
$$x_4 = 1 + 2\,s_2$$
$$x_5 = s_2$$

where, s_1 and s_2 are ***free parameters*** and can be *any real numbers*. Remember that each distinct choice for free parameters give a *new* ***particular solution***, so the given system has ***infinitely many solutions***.

Example 8.2: The LinearSolve Command

Solve the following system of linear equations using command **LinearSolve**:

$$x - 2y + z = 0$$
$$2y - 8z = 8$$
$$-4x + 5y + 9z = -9$$

- In the system of linear equations the coefficients change, but variables do not. Therefore, coefficients can be simply transferred to ***matrix,*** which can be thought as a rectangular table of numbers.
- **Step 1.** Construct the **coefficient matrix** of the given system.

```
ClearAll["Global`*"]
A = {{1, -2, 1}, {0, 2, -8}, {-4, 5, 9}}; MatrixForm[A]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & -2 & 1 \\ 0 & 2 & -8 \\ -4 & 5 & 9 \end{pmatrix}$$

- **Step 2.** Construct a **column matrix** that contains all the constant terms on the right-hand-side of each equation.

In[ ]:=
```
b = {0, 8, -9}; MatrixForm[b]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 \\ 8 \\ -9 \end{pmatrix}$$

- **Step 3.** Solve the system using **LinearSolve** command.

In[ ]:=
```
? LinearSolve
```

Out[ ]=

Symbol

LinearSolve[m, b] finds an x that solves the matrix equation $m.x == b$.

LinearSolve[m] generates a LinearSolveFunction[…] that can be applied repeatedly to different b.

```
In[ ]:= LinearSolve[A, b]
Out[ ]=
{29, 16, 3}
```

- **Step 4.** Verify the result.

```
In[ ]:= sys = {x - 2 y + z == 0, 2 y - 8 z == 8, -4 x + 5 y + 9 z == -9}; Column[sys]
Out[ ]=
x - 2 y + z == 0
2 y - 8 z == 8
-4 x + 5 y + 9 z == -9

In[ ]:= FullSimplify[sys /. {x → 29, y → 16, z → 3}]
Out[ ]=
{True, True, True}
```

- Hence, the unique solution for given consistent system of linear equations is:

$$x = 29$$
$$y = 16$$
$$z = 3$$

Example 8.3: Gaussian Elimination

Solve the following system of linear equations using **Gaussian elimination:**

$$2x_1 - 2x_2 - 6x_3 + x_4 = 3$$
$$-x_1 + x_2 + 3x_3 - x_4 = -3$$
$$x_1 - 2x_2 - x_3 + x_4 = 2$$

Definition 8.3: Augmented Matrix

When a matrix contains all the coefficients of a linear system, including the constant terms on the right side of each equation, it is called an **augmented matrix.** Augmented matrices include a vertical lin separating the left and right sides of the equation.

- **Step 1.** Construct the **augmented matrix** of the system, **AugMat= [A | b]**.
- Construct the **coefficient matrix.**

```
In[ ]:= ClearAll["Global`*"]
        A = {{2, -2, -6, 1}, {-1, 1, 3, -1}, {1, -2, -1, 1}}; A // MatrixForm
Out[ ]//MatrixForm=
```

$$\begin{pmatrix} 2 & -2 & -6 & 1 \\ -1 & 1 & 3 & -1 \\ 1 & -2 & -1 & 1 \end{pmatrix}$$

◆ Construct the matrix of **constant terms** on the RHS (right-hand-side).

```
In[ ]:= b = {3, -3, 2}; b // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 3 \\ -3 \\ 2 \end{pmatrix}$$

◆ Augmented matrix:

```
In[ ]:= AugMat = {{2, -2, -6, 1, 3}, {-1, 1, 3, -1, -3}, {1, -2, -1, 1, 2}}; AugMat // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 & -2 & -6 & 1 & 3 \\ -1 & 1 & 3 & -1 & -3 \\ 1 & -2 & -1 & 1 & 2 \end{pmatrix}$$

Definition 8.4: Elementary Row Operations

1. Interchange two rows

2. Multiply a row by nonzero constant

3. Add a multiple of one row to another row

Definition 8.5: Gaussian Elimination, Echelon Form, Leading Term

The procedure of reducing the matrix to the echelon form (or row echelon form) using the elementary row operations is known as **Gaussian elimination.**

A matrix is in **echelon form** if

(a) Every leading term is in a column to the left of the leading term of the row below it.

(b) Any zero rows are at the bottom of the matrix.

where the **leading term** of a row is defined as the leftmost nonzero term in that row.

Definition 8.6: Pivot Positions, Pivot Columns, Pivot

For a matrix in echelon form, the **pivot positions** are those that contain a leading term. The **pivot columns** are the columns that contain pivot positions, and a **pivot** is a nonzero number in a pivot position.

◆ **Step 2.** Apply **elementary row operations** to transform the augmented matrix to **row echelon form.**

◆ Identify pivot position for Row 1. $R_1 \leftrightarrow R_3$

```
In[ ]:= AugMat[[{1, 3}]] = AugMat[[{3, 1}]]; AugMat // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & -2 & -1 & 1 & 2 \\ -1 & 1 & 3 & -1 & -3 \\ 2 & -2 & -6 & 1 & 3 \end{pmatrix}$$

- Eliminate the coefficients down the first column below the pivot position, a_{21} and a_{31} by transforming them to zero. $R_1 + R_2 \to R_2$ and $-2R_1 + R_3 \to R_3$.

```
In[ ]:= AugMat[[2]] = AugMat[[1]] + AugMat[[2]];
        AugMat[[3]] = -2 * AugMat[[1]] + AugMat[[3]]; AugMat // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & -2 & -1 & 1 & 2 \\ 0 & -1 & 2 & 0 & -1 \\ 0 & 2 & -4 & -1 & -1 \end{pmatrix}$$

- Eliminate the coefficient of a_{32} below the pivot position in the second column: $2R_2 + R_3 \to R_3$.

```
In[ ]:= AugMat[[3]] = 2 * AugMat[[2]] + AugMat[[3]]; AugMat // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & -2 & -1 & 1 & 2 \\ 0 & -1 & 2 & 0 & -1 \\ 0 & 0 & 0 & -1 & -3 \end{pmatrix}$$

- The matrix is now in **row echelon form.**
- **Step 3.** Interpret the result of **Step 2** and find all solutions.
- The corresponding echelon system is:

```
In[ ]:= sys = {x1 - 2 x2 - x3 + x4 == 2, -x2 + 2 x3 == -1, -x4 == -3}; Column[sys]
```

Out[ ]=

$$\begin{aligned} &x1 - 2\,x2 - x3 + x4 == 2 \\ &-x2 + 2\,x3 == -1 \\ &-x4 == -3 \end{aligned}$$

- Apply **back substitution** procedure from Example 8.1. Here, x_3 is **free variable**.

```
In[ ]:= Solve[sys /. x3 → s1, {x1, x2, x4}]
```

Out[ ]=

$$\{\{x1 \to 1 + 5\,s_1,\ x2 \to 1 + 2\,s_1,\ x4 \to 3\}\}$$

- **Step 4.** Verify the solution.

```
In[ ]:= sys1 = {2 x1 - 2 x2 - 6 x3 + x4 == 3, -x1 + x2 + 3 x3 - x4 == -3, x1 - 2 x2 - x3 + x4 == 2};

In[ ]:= FullSimplify[sys1 /. {x1 → 1 + 5 s1, x2 → 1 + 2 s1, x3 → s1, x4 → 3}]
```

Out[ ]=

```
{True, True, True}
```

In[ ]:=
```
LinearSolve[A, b]
```
Out[ ]=
```
{1, 1, 0, 3}
```

In[ ]:=
```
FullSimplify[sys1 /. {x1 → 1, x2 → 1, x3 → 0, x4 → 3}]
```
Out[ ]=
```
{True, True, True}
```

- Hence, the given system of linear equations is consistent as has a **general solution** of:

$$x_1 = 1 + 5\,s_1$$
$$x_2 = 1 + 2\,s_1$$
$$x_3 = s_1$$
$$x_4 = 3$$

where, s_1 can be *any real number*.

Example 8.4: Gauss-Jordan Elimination

Solve the following system of linear equations using **Gauss-Jordan elimination:**

$$5x + 2y + 11z = 4$$
$$7x + 3y + 4z = 1$$
$$12x + 5y + 15z = 6$$

- **Step 1.** Construct the **augmented matrix** of the system, **AugMat= [A|b].**

In[ ]:=
```
ClearAll["Global`*"]
sys = {5 x + 2 y + 11 z == 4, 7 x + 3 y + 4 z == 1, 12 x + 5 y + 15 z == 6}; Column[sys]
```
Out[ ]=
```
5 x + 2 y + 11 z == 4
7 x + 3 y + 4 z == 1
12 x + 5 y + 15 z == 6
```

- Coefficient matrix A:

In[ ]:=
```
A = Normal[CoefficientArrays[sys, {x, y, z}]][[2]]; MatrixForm[A]
```
Out[ ]//MatrixForm=
$$\begin{pmatrix} 5 & 2 & 11 \\ 7 & 3 & 4 \\ 12 & 5 & 15 \end{pmatrix}$$

- Column matrix b with right-hand-side constants:

```
In[ ]:= b = {{4}, {1}, {6}}; MatrixForm[b]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 4 \\ 1 \\ 6 \end{pmatrix}$$

- Augmented matrix:

```
In[ ]:= AugMat = ArrayFlatten[{{A, b}}]; MatrixForm[AugMat]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 5 & 2 & 11 & 4 \\ 7 & 3 & 4 & 1 \\ 12 & 5 & 15 & 6 \end{pmatrix}$$

Definition 8.7: Gauss-Jordan Elimination, Reduced Echelon Form

The procedure of reducing the matrix to the reduced echelon form (or reduced row echelon form) using the elementary row operations is known as **Gauss-Jordan elimination.**

A matrix is in **reduced echelon form** if

(a) It is in echelon form.

(b) All pivot positions contain a 1.

(c) The only nonzero term in a pivot column is in the pivot position.

- **Step 2.** Apply **elementary row operations** to transform the augmented matrix to **reduced row echelon form.**

$$\frac{1}{5} R_1 \to R_1$$
$$-7 R_1 + R_2 \to R_2$$
$$-12 R_1 + R_3 \to R_3$$
$$5 R_2 \to R_2$$
$$R_1 - \frac{2}{5} R_2 \to R_1$$
$$-\frac{1}{5} R_2 + R_3 \to R_3$$
$$R_1 - 10 R_3 \to R_1$$
$$R_2 + 23 R_3 \to R_2$$

```
In[ ]:= AugMat[[1]] = 1/5 * AugMat[[1]];
        AugMat[[2]] = -7 * AugMat[[1]] + AugMat[[2]];
        AugMat[[3]] = -12 * AugMat[[1]] + AugMat[[3]];
        AugMat[[2]] = 5 * AugMat[[2]];
        AugMat[[1]] = AugMat[[1]] - 2/5 * AugMat[[2]];
        AugMat[[3]] = -1/5 * AugMat[[2]] + AugMat[[3]];
        AugMat[[1]] = AugMat[[1]] - 10 * AugMat[[3]];
        AugMat[[2]] = AugMat[[2]] + 23 * AugMat[[3]];
        AugMat // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 25 & 0 \\ 0 & 1 & -57 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- The matrix is now in **reduced row echelon form**. But, the third row shows ***inconsistency***. Since **0 = 1** is not a true condition, given system of equations has no solution.
- **Step 4.** Verify the solution.

```
In[ ]:= RREF = RowReduce[ArrayFlatten[{{A, b}}]]; RREF // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 25 & 0 \\ 0 & 1 & -57 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

```
In[ ]:= AugMat == RREF
Out[ ]= True

In[ ]:= LinearSolve[A, b]
```

LinearSolve: Linear equation encountered that has no solution.

```
Out[ ]= LinearSolve[{{5, 2, 11}, {7, 3, 4}, {12, 5, 15}}, {{4}, {1}, {6}}]
```

Considering the last row, the system is inconsistent, i.e., has **no solution**.

Summary

After completing this chapter, you should be able to

- develop SOPs to solve systems of linear equations using wolfram Mathematica.
- be familiar with the list form and matrix form representations of data in Mathematica.

- practice Gaussian elimination and Gauss-Jordan elimination in Mathematica.
- develop the habit of always checking your solutions for quality assurance.

Week 9: Matrix Operations and Inverse

Properties of Matrix Operations and Inverse

Table of Contents

Commands list

- Table
- Dimensions
- ConstantArray
- RandomInteger
- RandomReal
- Do
- For
- Sum
- SeedRandom
- UpperTriangularize
- LowerTriangularize
- Dot
- Transpose
- Inverse

Prerequisite: How to Create a Matrix

Matrices are represented in the Wolfram Language with lists. They can be entered directly with the { } notation, constructed from a formula, or imported from a data file. The Wolfram

Language also has commands for creating diagonal matrices, constant matrices, and other special matrix types.

How to | Create a Matrix:
https://reference.wolfram.com/language/howto/CreateAMatrix.html

Properties of Matrix Algebra

Example 9.1: Matrix Addition | Scalar Multiplication

Definition 9.1: Addition, Scalar Multiplication of Matrices

Let c be a scalar, and let

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1\,m} \\ a_{21} & a_{22} & \cdots & a_{2\,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1\,\mathrm{m}} \\ b_{21} & b_{22} & \cdots & b_{2\,\mathrm{m}} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{\mathrm{n2}} & \cdots & b_{\mathrm{nm}} \end{pmatrix}$$

be $n \times m$ matrices. Then addition and scalar multiplication of matrices are defined as follows:

Addition:
$$A + B = \begin{pmatrix} (a_{11} + b_{11}) & (a_{12} + b_{12}) & \cdots & (a_{1\,m} + b_{1\,m}) \\ (a_{21} + b_{21}) & (a_{22} + b_{22}) & \cdots & (a_{2\,m} + b_{2\,m}) \\ \vdots & \vdots & \ddots & \vdots \\ (a_{n1} + b_{n1}) & (a_{n2} + b_{n2}) & \cdots & (a_{nm} + b_{nm}) \end{pmatrix}$$

Scalar Multiplication:
$$cA = \begin{pmatrix} ca_{11} & ca_{12} & \cdots & ca_{1\,m} \\ ca_{21} & ca_{22} & \cdots & ca_{2\,m} \\ \vdots & \vdots & \ddots & \vdots \\ ca_{n1} & ca_{n2} & \cdots & ca_{nm} \end{pmatrix}$$

Construct two 3×3 matrices and prove one of the properties below by performing corresponding matrix operations.

Theorem 9.1: Properties of Addition and Scalar Multiplication

Let s and t be scalars, A, B, and C be matrices of dimensions $n \times m$, and 0_{nm} be the $n \times m$ matrix with all zero entries. Then

(a) $A + B = B + A$

(b) $s(A+B) = sA + sB$

(c) $(s+t)A = sA + tA$

(d) $(A+B)+C = A+(B+C)$

(e) $(st)A = s(tA)$

(f) $A + 0_{nm} = A$

⚠ Note that two matrices can be **equal** if they have the same dimensions and if their corresponding entries are equal.

- **2nd property** is chosen for this example.
- **Step 1.** Construct two 3×3 matrices.

```
In[ ]:= ClearAll["Global`*"]
        A = Table[a_{i,j}, {i, 3}, {j, 3}]; MatrixForm[A]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix}$$

```
In[ ]:= B = Table[b_{i,j}, {i, 3}, {j, 3}]; MatrixForm[B]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,1} & b_{3,2} & b_{3,3} \end{pmatrix}$$

- **Step 2.** Check that they have the same size.

```
In[ ]:= Dimensions[A] == Dimensions[B]
```

Out[ ]=

```
True
```

Method 1. Use a **Do loop** to perform matrix operations.

- **Step 3.** Compute their **sum** $A + B$.
- Get the dimension of matrix A(or B).

```
In[ ]:= dim = Dimensions[A]
```

Out[ ]=

```
{3, 3}
```

- Initially, new matrix **sumAB** will be a zero matrix and its elemts will be replaced later.

In[ ]:=
```
sumAB = ConstantArray[0, dim]; MatrixForm[sumAB]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

- Run a **do loop** to perform matrix addition.

In[ ]:=
```
Do[sumAB[[i, j]] = A[[i, j]] + B[[i, j]], {i, 1, dim[[1]]}, {j, 1, dim[[2]]}];
MatrixForm[sumAB]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} a_{1,1}+b_{1,1} & a_{1,2}+b_{1,2} & a_{1,3}+b_{1,3} \\ a_{2,1}+b_{2,1} & a_{2,2}+b_{2,2} & a_{2,3}+b_{2,3} \\ a_{3,1}+b_{3,1} & a_{3,2}+b_{3,2} & a_{3,3}+b_{3,3} \end{pmatrix}$$

- **Step 4.** Multiply the sum ***A* + *B*** by the scalar ***s***.
- Assing a value of zero to resulting matrix **RHS** (right-hand-side).

In[ ]:=
```
RHS = ConstantArray[0, dim]; RHS // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

- Run a **do loop** to perform scalar multiplication.

In[ ]:=
```
Do[RHS[[i, j]] = s * sumAB[[i, j]], {i, 1, dim[[1]]}, {j, 1, dim[[2]]}];
MatrixForm[RHS]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} s\,(a_{1,1}+b_{1,1}) & s\,(a_{1,2}+b_{1,2}) & s\,(a_{1,3}+b_{1,3}) \\ s\,(a_{2,1}+b_{2,1}) & s\,(a_{2,2}+b_{2,2}) & s\,(a_{2,3}+b_{2,3}) \\ s\,(a_{3,1}+b_{3,1}) & s\,(a_{3,2}+b_{3,2}) & s\,(a_{3,3}+b_{3,3}) \end{pmatrix}$$

Method 2. Use a **For loop** to perform matrix operations.

- **Step 5.** Find the **scalar multiples** of ***A*** and ***B***, ***sA*** and ***sB*** using a **for loop**.
- Define new zero matrices elements of which will be replaced later.

```
sA = ConstantArray[0, dim];
sB = ConstantArray[0, dim];
```

- Run a **for loop** to perform scalar multiplication.

```
In[ ]:= For[i = 1, i ≤ 3, i++,
          For[j = 1, j ≤ 3, j++,
           {sA[[i, j]] = s * A[[i, j]], sB[[i, j]] = s * B[[i, j]]}
          ]];
        MatrixForm[sA]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} s\,a_{1,1} & s\,a_{1,2} & s\,a_{1,3} \\ s\,a_{2,1} & s\,a_{2,2} & s\,a_{2,3} \\ s\,a_{3,1} & s\,a_{3,2} & s\,a_{3,3} \end{pmatrix}$$

```
In[ ]:= MatrixForm[sB]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} s\,b_{1,1} & s\,b_{1,2} & s\,b_{1,3} \\ s\,b_{2,1} & s\,b_{2,2} & s\,b_{2,3} \\ s\,b_{3,1} & s\,b_{3,2} & s\,b_{3,3} \end{pmatrix}$$

- **Step 6.** Find the sum ***sA +sB*** using a **for loop**.

```
In[ ]:= LHS = ConstantArray[0, dim]; MatrixForm[LHS]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

```
In[ ]:= For[i = 1, i ≤ 3, i++,
          For[j = 1, j ≤ 3, j++,
           LHS[[i, j]] = sA[[i, j]] + sB[[i, j]]
          ]];
        MatrixForm[LHS]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} s\,a_{1,1} + s\,b_{1,1} & s\,a_{1,2} + s\,b_{1,2} & s\,a_{1,3} + s\,b_{1,3} \\ s\,a_{2,1} + s\,b_{2,1} & s\,a_{2,2} + s\,b_{2,2} & s\,a_{2,3} + s\,b_{2,3} \\ s\,a_{3,1} + s\,b_{3,1} & s\,a_{3,2} + s\,b_{3,2} & s\,a_{3,3} + s\,b_{3,3} \end{pmatrix}$$

- **Step 7.** Check and verify the **2nd property** of matrix addition and scalar multiplication.

```
In[ ]:= FullSimplify[RHS == LHS]
```

Out[ ]=

```
True
```

- **Step 8.** Verify the results.

```
In[ ]:= RHS == s * (A + B)
```

Out[ ]=

```
True
```

```
In[ ]:= LHS == s * A + s * B
```

Out[ ]=

```
True
```

Example 9.2: Matrix Multiplication

Definition 9.2: Matrix Multiplication

Let A be an $n \times k$ matrix and $B = [\ \mathbf{b}_1 \quad \mathbf{b}_2 \quad \cdots \quad \mathbf{b}_m\ ]$ a $k \times m$ matrix. We define the product

$$AB = [\ A\mathbf{b}_1 \quad A\mathbf{b}_2 \quad \cdots \quad A\mathbf{b}_m\ ]$$

which is an $n \times m$ matrix.

$$c_{ij} = a_{i1}\, b_{1j} + a_{i2}\, b_{2j} + \cdots + a_{in}\, b_{nj} = \sum_{k=1}^{n} a_{ik}\, b_{kj}$$

⚠ Note that for AB to exist, the number of columns of A must equal the number of rows of B.

Construct two matrices of dimensions 4×3 and 3×4, respectively, with random entries and prove one of the properties below by performing corresponding matrix operations.

Theorem 9.2: Properties of Matrix Multiplication

Let s be a scalar, and let A, B, and C be matrices. Then each of the following holds in the cases where the indicated operations are defined:

(a) $A(BC) = (AB)C$

(b) $A(B + C) = AB + AC$

(c) $(A + B)C = AC + BC$

(d) $s(AB) = (sA)B = A(sB)$

(e) $AI = A$

(f) $AB \neq BA$ (even if the matrices have compatible dimensions)

Here I denotes an identity matrix of appropriate dimension.

- **6th property** is chosen for this example.
- **Step 1.** Construct two matrices with appropriate dimensions and elements.

In[ ]:=
```
ClearAll["Global`*"]
A = RandomInteger[{-10, 10}, {4, 3}]; MatrixForm[A]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 3 & -2 & -1 \\ -10 & 6 & -6 \\ -8 & -7 & 2 \\ -5 & -2 & -7 \end{pmatrix}$$

```
In[ ]:= B = RandomInteger[{-10, 10}, {3, 4}]; MatrixForm[B]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -10 & 7 & 2 & 1 \\ 8 & -1 & -5 & -1 \\ 1 & -3 & 7 & -1 \end{pmatrix}$$

- Since the dimension of A is 4×3 and dimension of B is 3×4, their product will give a new matrices $\boldsymbol{AB}$ of dimension 4×4 and $\boldsymbol{BA}$ of dimension 3×3.
- **Step 2.** Calculate the product $\boldsymbol{AB}$ using a **do loop** and based on the given formula below:

$$c_{ij} = a_{i1}\, b_{1j} + a_{i2}\, b_{2j} + \cdots + a_{in}\, b_{nj} = \sum_{k=1}^{n} a_{ik}\, b_{kj}$$

- Get the dimension of matrix A and B.

```
In[ ]:= dimA = Dimensions[A];
        dimB = Dimensions[B];
```

- Initially, new matrix $\boldsymbol{AB}$ will be a zero matrix and its elemts will be replaced later.

```
In[ ]:= AB = ConstantArray[0, {4, 4}];
```

- Run a **do loop** to perform matrix multiplication.

```
In[ ]:= Do[AB[[i, j]] = Sum[(A[[i, k]]) * (B[[k, j]]), {k, 1, 3}], {i, 1, dimA[[1]]}, {j, 1, dimB[[2]]}];
        MatrixForm[AB]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -47 & 26 & 9 & 6 \\ 142 & -58 & -92 & -10 \\ 26 & -55 & 33 & -3 \\ 27 & -12 & -49 & 4 \end{pmatrix}$$

- **Step 3.** Calculate the product $\boldsymbol{BA}$ using a **for loop**.

$$c_{ij} = a_{i1}\, b_{1j} + a_{i2}\, b_{2j} + \cdots + a_{in}\, b_{nj} = \sum_{k=1}^{n} a_{ik}\, b_{kj}$$

- Define a zero matrix $\boldsymbol{BA}$.

```
In[ ]:= BA = ConstantArray[0, {3, 3}];
```

- Run a **for loop** based on the formula above.

```
In[ ]:= For[i = 1, i ≤ 3, i++,
          For[j = 1, j ≤ 3, j++,
           BA[[i, j]] = Sum[(B[[i, k]]) * (A[[k, j]]), {k, 1, 4}]]];
        MatrixForm[BA]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -121 & 46 & -35 \\ 79 & 15 & -5 \\ -18 & -67 & 38 \end{pmatrix}$$

- **Step 4.** Check and verify the **6th property.**

In[ ]:=
```
AB ≠ BA
```
Out[ ]=
```
True
```

◆ **Step 5.** Verify the results.

In[ ]:=
```
AB == Dot[A, B]
```
Out[ ]=
```
True
```

In[ ]:=
```
BA == Dot[B, A]
```
Out[ ]=
```
True
```

In[ ]:=
```
A.B ≠ B.A
```
Out[ ]=
```
True
```

Example 9.3: Transpose of a Matrix

Definition 9.3: Transpose

The **transpose** of a matrix A is denoted by A^T and results from interchanging the rows and columns of A. Focusing on individual entries, the entry in row i and column j of A becomes the entry in row j and column i of A^T.

Construct upper/lower triangular matrix or matrices with random entries and prove one of the properties below by performing corresponding matrix operations.

Theorem 9.3: Properties of Matrix Transposes

Let s be a scalar, A and B be $n \times m$ matrices, and C an $m \times k$ matrix. Then

(a) $\left(A^T\right)^T = A$

(b) $(A+B)^T = A^T + B^T$

(c) $(sA)^T = sA^T$

(d) $(AC)^T = C^T A^T$

◆ **1st and2nd properties** are chosen for this example.

Definition 9.4: Upper and Lower Triangular Matrices

An $n \times n$ matrix A is **upper triangular** if the entries below the diagonal are all zero.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & a_{22} & a_{23} & \cdots & a_{2n} \\ 0 & 0 & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn} \end{pmatrix}$$

Similarly, an $n \times n$ matrix A is **lower triangular** if the terms above the diagonal are all zero.

$$A = \begin{pmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ a_{21} & a_{22} & 0 & \cdots & 0 \\ a_{31} & a_{32} & a_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix}$$

◆ **Step 1.** Construct two matrices with appropriate dimensions and elements.

In[ ]:=
```
ClearAll["Global`*"]
SeedRandom[1];
A = UpperTriangularize[RandomInteger[{-10, 10}, {4, 4}]]; A // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -5 & -10 & -3 & -10 \\ 0 & -7 & -10 & -10 \\ 0 & 0 & -7 & -2 \\ 0 & 0 & 0 & 6 \end{pmatrix}$$

In[ ]:=
```
SeedRandom[2]; B = LowerTriangularize[RandomInteger[{-10, 10}, {4, 4}]]; B // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -7 & 0 & 0 & 0 \\ 7 & -2 & 0 & 0 \\ -10 & 9 & -1 & 0 \\ -6 & -7 & -6 & 2 \end{pmatrix}$$

◆ **Step 2.** Verify that $\left(A^T\right)^T = A$.

In[ ]:=
```
At = Transpose [A]; At // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -5 & 0 & 0 & 0 \\ -10 & -7 & 0 & 0 \\ -3 & -10 & -7 & 0 \\ -10 & -10 & -2 & 6 \end{pmatrix}$$

In[ ]:=
```
Att = Transpose[At]; Att // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -5 & -10 & -3 & -10 \\ 0 & -7 & -10 & -10 \\ 0 & 0 & -7 & -2 \\ 0 & 0 & 0 & 6 \end{pmatrix}$$

In[ ]:=
```
Att == A
```
Out[ ]=
```
True
```

In[ ]:=
```
Transpose [Transpose[A]] == A
```
Out[ ]=
```
True
```

◆ **Step 3.** Verify that $(A+B)^T = A^T + B^T$.

In[ ]:=
```
RHS = Transpose[A + B]; RHS // MatrixForm
```
Out[ ]//MatrixForm=
$$\begin{pmatrix} -12 & 7 & -10 & -6 \\ -10 & -9 & 9 & -7 \\ -3 & -10 & -8 & -6 \\ -10 & -10 & -2 & 8 \end{pmatrix}$$

In[ ]:=
```
LHS = Transpose[A] + Transpose[B]; LHS // MatrixForm
```
Out[ ]//MatrixForm=
$$\begin{pmatrix} -12 & 7 & -10 & -6 \\ -10 & -9 & 9 & -7 \\ -3 & -10 & -8 & -6 \\ -10 & -10 & -2 & 8 \end{pmatrix}$$

In[ ]:=
```
RHS == LHS
```
Out[ ]=
```
True
```

Example 9.4: Inverse of a Matrix

Definition 9.4: Invertible Matrix

An $n \times n$ matrix A is **invertible** if there exists an $n \times n$ matrix matrix B such that $\boldsymbol{AB = I_n}$.

Definition 9.6: Identity Matrix

Identity matrix is an $n \times n$ matrix that contain only 1's on its main diagonal and 0's elsewhere else.

$$I_n = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1_n \end{pmatrix}$$

Construct invertible matrix or matrices with random entries and prove one of the properties

below.

Theorem 9.4: Properties of Matrix Inverse

Let A and B be invertible $n \times n$ matrices. Then

(a) If matrix A is invertible, then $\left(A^{-1}\right)^{-1} = A$.

(b) If A and B are both invertible n×n matrices, then $(AB)^{-1} = B^{-1} A^{-1}$.

(c) If A is invertible, then $\left(A^{T}\right)^{-1} = \left(A^{-1}\right)^{T}$.

- **1st property** is chosen for this example.
- **Step 1.** Construct a matrix with appropriate dimension and elements.

```
In[ ]:= ClearAll["Global`*"]
        SeedRandom[1];
        A = RandomReal[{0, 10}, {3, 3}]; A // MatrixForm
```

```
Out[ ]//MatrixForm=
        8.17389   1.1142   7.89526
        1.87803   2.41361  0.657388
        5.42247   2.31155  3.96006
```

- **Step 2.** Check whether this matrix is invertible or not. The matrix is invertible since its determinant is nonzero.

```
In[ ]:= Det[A] ≠ 0
```

```
Out[ ]=
        True
```

Theorem 9.5: Comdition for Invertible Matrix

An $n \times n$ matrix A is **invertible** if and only if **det(A) ≠ 0**.

- **Step 3.** Find A^{-1}.
- Construct the **augmented matrix**, $[A \mid I_2]$.

```
In[ ]:= AugMat = ArrayFlatten[{{A, IdentityMatrix[3]}}]; MatrixForm[AugMat]
```

```
Out[ ]//MatrixForm=
        8.17389   1.1142   7.89526   1  0  0
        1.87803   2.41361  0.657388  0  1  0
        5.42247   2.31155  3.96006   0  0  1
```

- Transform the augmented matrix to **reduced row echelon form.**

```
In[ ]:= RREF = RowReduce[AugMat]; RREF // MatrixForm
```

```
Out[ ]//MatrixForm=
        1  0.  0.  -1.04865  -1.80522   2.39039
        0  1   0.  0.505178   1.36229  -1.23333
        0  0   1   1.14102    1.67668  -2.3007
```

- On the left-hand-side is the identity matrix and on the right-hand-side is the inverse matrix. Thus, we find that A^{-1} is:

```
In[ ]:= InvA = RREF[[All, 4 ;; 6]]; InvA // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -1.04865 & -1.80522 & 2.39039 \\ 0.505178 & 1.36229 & -1.23333 \\ 1.14102 & 1.67668 & -2.3007 \end{pmatrix}$$

- **Step 4.** Find $\left(A^{-1}\right)^{-1}$.
- Augmented matrix, $\left[A^{-1} \mid I_2\right]$.

```
In[ ]:= AugMat2 = ArrayFlatten[{{InvA, IdentityMatrix[3]}}]; MatrixForm[AugMat2]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -1.04865 & -1.80522 & 2.39039 & 1 & 0 & 0 \\ 0.505178 & 1.36229 & -1.23333 & 0 & 1 & 0 \\ 1.14102 & 1.67668 & -2.3007 & 0 & 0 & 1 \end{pmatrix}$$

- Reduced row echelon form:

```
In[ ]:= RREF2 = RowReduce[AugMat2]; RREF2 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 0. & 0. & 8.17389 & 1.1142 & 7.89526 \\ 0 & 1 & 0. & 1.87803 & 2.41361 & 0.657388 \\ 0 & 0 & 1 & 5.42247 & 2.31155 & 3.96006 \end{pmatrix}$$

- The right-hand-side is the inverse matrix. Thus, $\left(A^{-1}\right)^{-1}$ is:

```
In[ ]:= RHS = RREF2[[All, 4 ;; 6]]; RHS // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 8.17389 & 1.1142 & 7.89526 \\ 1.87803 & 2.41361 & 0.657388 \\ 5.42247 & 2.31155 & 3.96006 \end{pmatrix}$$

- **Step 5.** Check and verify the **1st property**.

```
In[ ]:= RHS == A
```

Out[ ]=

```
True
```

- **Step 6.** Verify the solution.

```
In[ ]:= RHS == Inverse[Inverse[A]]
```

Out[ ]=

```
True
```

```
In[ ]:= Inverse[Inverse[A]] == A
```

Out[ ]=

```
True
```

Summary

After completing this chapter, you should be able to

- perform various standard matrix operations using Mathematica.
- write simple programs that involve looping and conditional expressions in Mathematica
- generate random numbers in Mathematica.

Week 10: LU Factorization and Determinants

Applications of Matrix Operations and Determinants

Table of Contents

Commands list

- LUDecomposition
- UpperTriangularize
- LowerTriangularize
- Det
- Times
- Diagonal
- Reverse
- Transpose
- Inverse
- Minors
- Cofactor

Prerequisite: Basic Matrix Operations

The Wolfram Language's matrix operations handle both numeric and symbolic matrices, automatically accessing large numbers of highly efficient algorithms. The Wolfram Language uses state-of-the-art algorithms to work with both dense and sparse matrices, and incorporates a number of powerful original algorithms, especially for high-precision and symbolic matrices.

Basic Matrix Operations:
https://reference.wolfram.com/language/guide/MatrixOperations.html
https://reference.wolfram.com/language/tutorial/LinearAlgebra.html#25697

Example 10.1: The LU Factorization

The LU factorization can be used to solve linear systems of equations in a convenient manner. Solve the following linear system of equations using LU factorization algothithm.

$$x_1 - x_2 - 2\,x_3 = 2$$
$$-3\,x_1 + -2\,x_2 + 1\,x_3 = 5$$
$$6\,x_1 + 11\,x_2 - 2\,x_3 = 1$$

Definition 10.1: LU Factorization

A nonsingular square matrix A is said to have an **LU factorization** if it can be written in the form:

$$A = LU$$

where L is a lower triangular matrix with 1's on the diagonaland U is an upper triangular matrix in echelon form.

- **Step 1.** Define coefficient matrix A and matrix b from the system expressed as $Ax = b$.

```
In[ ]:= ClearAll["Global`*"]
        A = {{1, -1, -2}, {-3, 2, 1}, {6, 11, -2}}; MatrixForm[A]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & -1 & -2 \\ -3 & 2 & 1 \\ 6 & 11 & -2 \end{pmatrix}$$

```
In[ ]:= b = {2, 5, 1}; MatrixForm[b]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 \\ 5 \\ 1 \end{pmatrix}$$

- **Step 2.** Compute the **LU factorization** of A.

Method 1. LU factorization using **Row Operations.**

- Obtain **U** by the process of row reduction of A, and build up **L** one column at a time as we transform A to echelon form. For this, first construct a lower triangular matrix **L1**, where the • symbol represents a matrix entry that has not yet been determined.

```
In[ ]:= L1 = ConstantArray[•, {3, 3}]; L1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{pmatrix}$$

- Take the first column of A, divide each entry by the pivot (**1**), and use the resulting values to form the first column of **L1**.

```
In[ ]:= L1[[All, 1]] = A[[All, 1]] / 1; L1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & \bullet & \bullet \\ -3 & \bullet & \bullet \\ 6 & \bullet & \bullet \end{pmatrix}$$

- Our goal is to construct upper-triangular matrix **U1** by transforming A to echeclon form. For this, set **U1** equal to A and perform a row operations on A to introduce zeros down the first column of A.

```
In[ ]:= U1 = A;
        U1[[2]] = U1[[2]] + 3 U1[[1]];
        U1[[3]] = U1[[3]] - 6 U1[[1]]; U1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & -1 & -2 \\ 0 & -1 & -5 \\ 0 & 17 & 10 \end{pmatrix}$$

- Take the second column of A, starting from the pivot entry (**-1**) down, and divide each entry by the pivot. Use the resulting values to form the lower portion of the second column of **L1**.

```
In[ ]:= L1[[2 ;; 3, 2]] = U1[[2 ;; 3, 2]] / -1; L1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & \bullet & \bullet \\ -3 & 1 & \bullet \\ 6 & -17 & \bullet \end{pmatrix}$$

- Perform a row operations on A to introduce zeros down the second column of A.

In[ ]:=
```
U1[[3]] = U1[[3]] + 17 U1[[2]]; U1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & -1 & -2 \\ 0 & -1 & -5 \\ 0 & 0 & -75 \end{pmatrix}$$

- Now we have finished with **U1**. The original matrix is in echelon form and upper triangular.
- Finish filling in **L1**. Since **L1** must be unit lower triangular, we put a **1** in the lower right corner and fill in the remaining entries with **0**'s.

In[ ]:=
```
L1[[3, 3]] = 1;
L1[[1, 2]] = 0;
L1[[1 ;; 2, 3]] = 0; L1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ 6 & -17 & 1 \end{pmatrix}$$

- Verify the results of **Method 1** using standard matrix multiplication.

```
Dot[L1, U1] == A
```

Out[ ]=
```
True
```

Method 2. LU factorization using **LUDecomposition** command

Definition 10.2: LUDecomposition Command

LUDecomposition returns a list of three elements. The first element is a combination of upper and lower-triangular matrices, the second element is a permutation vector specifying rows used for pivoting, and the third element is an estimate of the condition number.

- Find the LU factorization of A using **LUDecomposition** command.

In[ ]:=
```
? LUDecomposition
```

Out[ ]=

Symbol

LUDecomposition[m] generates a representation of the LU decomposition of a square matrix m.

In[ ]:=
```
{lu, p, c} = LUDecomposition[A];
```

- Compute the factors **L2** and **U2**.

```
In[ ]:= L2 = LowerTriangularize[lu, -1] + IdentityMatrix[3]; L2 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ 6 & -17 & 1 \end{pmatrix}$$

```
In[ ]:= U2 = UpperTriangularize[lu]; U2 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & -1 & -2 \\ 0 & -1 & -5 \\ 0 & 0 & -75 \end{pmatrix}$$

- Define the permutation vector. The **permutation vector** indicates that the rows were interchanged while factoring the matrix.

```
In[ ]:= p
```

Out[ ]=

```
{1, 2, 3}
```

- Verify the results of **Method 2** by permutting the rows of A.

```
In[ ]:= L2.U2 == A[[p]]
```

Out[ ]=

```
True
```

- **Step 3.** Solve the given system of linear equations using the result of one of the methods above.
- Since we have verified that $\mathbf{A} = \mathbf{LU}$, the system can be written as $\mathbf{LU}\boldsymbol{x} = \boldsymbol{b}$. The first step is to denote $\boldsymbol{y} = \mathbf{U}\boldsymbol{x}$, so that our system can be expressed as $\mathbf{L}\boldsymbol{y} = \boldsymbol{b}$.
- Solve the equation $\mathbf{L}\boldsymbol{y} = \boldsymbol{b}$:

```
In[ ]:= y = LinearSolve[L2, b[[p]]]
```

Out[ ]=

```
{2, 11, 176}
```

- Solve the equation $\boldsymbol{y} = \mathbf{U}\boldsymbol{x}$:

```
In[ ]:= x = LinearSolve[U2, y]
```

Out[ ]=

$$\left\{-\frac{49}{25}, \frac{11}{15}, -\frac{176}{75}\right\}$$

- **Step 4.** Verify the solution.

```
In[ ]:= x == LinearSolve[A, b]
```

Out[ ]=

```
True
```

Example 10.2: Determinant and Its Properties | Part 1

Definition 10.3: Determinant of a 3×3 Matrix

Let A be the 3×3 matrix.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Then the determinant of A is given by

$$\det(A) = a_{11}\,a_{22}\,a_{33} + a_{12}\,a_{23}\,a_{31} + a_{13}\,a_{21}\,a_{32} - a_{11}\,a_{23}\,a_{32} - a_{12}\,a_{21}\,a_{33} - a_{13}\,a_{22}\,a_{31}$$

Construct a matrix(or matrices) of dimension 3×3. Compute its(or their) determinant(s) and prove the properties of your choice listed below.

Theorem 10.1: Properties of the Determinant

Let A and B be a $n\times n$ square matrices.

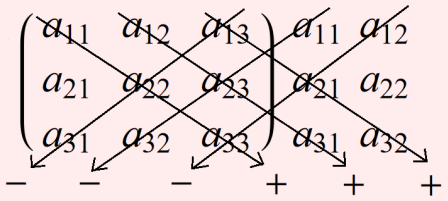
1. A is invertible if and only if $\det(A) \neq 0$.
2. For $n \geq 1$, we have $\det(I_n) = 1$.
3. If A is a triangular $n\times n$ matrix, then $\det(A)$ is the product of the terms along the diagonal.
4. $\det\left(A^T\right) = \det(A)$
5. If A has a row(or column) of zeros or if A has two identical rows(or columns), then $\det(A) = 0$.
6. $\det(AB) = \det(A)\det(B)$
7. Let A be invertible matrix, then $\det\left(A^{-1}\right) = \frac{1}{\det(A)}$
8. $\det(A+B) \neq \det(A) + \det(B)$

◆ **1st** and **4th properties** are chosen for this example.

Method 1. Use the **Shortcut Method** to compute the determinant.

Definition 10.4: The Shortcut Method for 3×3 Matrices

This method is also known as **Rule of Sarrus.**



⚠ Note that the Shortcut Method will not work for $n \times n$ matrices with dimensions larger than 3×3.

- **Step 1.** Construct a non-invertible matrix for proving the **1st property**.

```
In[ ]:= ClearAll["Global`*"]
        A1 = {{-3, 1, 2}, {5, 5, -8}, {4, 2, -5}}; MatrixForm[A1]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -3 & 1 & 2 \\ 5 & 5 & -8 \\ 4 & 2 & -5 \end{pmatrix}$$

- **Step 2.** Duplicate the first two columns of the matrix to the right of the third column of the original matrix.

```
In[ ]:= extraColumns = {A1[[All, 1]], A1[[All, 2]]};
        newA1 = Transpose[Join[Transpose[A1], extraColumns]]; newA1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -3 & 1 & 2 & -3 & 1 \\ 5 & 5 & -8 & 5 & 5 \\ 4 & 2 & -5 & 4 & 2 \end{pmatrix}$$

- **Step 3.1.** Now, for each diagonal arrow multiply terms and then add or subtract based on the labels.

```
In[ ]:= detA1 =
          A1[[1, 1]] * A1[[2, 2]] * A1[[3, 3]] +
          A1[[1, 2]] * A1[[2, 3]] * A1[[3, 1]] +
          A1[[1, 3]] * A1[[2, 1]] * A1[[3, 2]] -
          A1[[1, 3]] * A1[[2, 2]] * A1[[3, 1]] -
          A1[[1, 1]] * A1[[2, 3]] * A1[[3, 2]] -
          A1[[1, 2]] * A1[[2, 1]] * A1[[3, 3]]
```

Out[ ]=

```
0
```

```
In[ ]:= detA12 = (-3 * 5 * -5) + (1 * -8 * 4) + (2 * 5 * 2) - (2 * 5 * 4) - (-3 * -8 * 2) - (1 * 5 * -5)
```

Out[ ]=

```
0
```

- **Step 3.2.** Use **for loop** to eliminate repeating units.

- Here **Diagonal** command is used to get the elements along the desired diagonal, **Reverse** command is for getting the elements of back diagonal and **Times** command is used to get the product of that diagonal elements in a list.

```
In[ ]:= detA13 = 0;
        For[i = 0, i ≤ 2, i++,
          detA13 = detA13 + Times @@ Diagonal[newA1, i] - Times @@ Diagonal[Reverse[newA1], i]];
        Print[detA13]
```

```
0
```

- **Step 4.** Verify the result and check the **1st property.**

```
In[ ]:= detA1 == detA12 == detA13 == Det[A1]
Out[ ]=
        True

In[ ]:= Inverse[A1]
```

```
··· Inverse: Matrix {{-3, 1, 2}, {5, 5, -8}, {4, 2, -5}} is singular.
```

```
Out[ ]=
        Inverse[{{-3, 1, 2}, {5, 5, -8}, {4, 2, -5}}]
```

Method 2. Use the **Cofactor Expansion** to compute the determinant.

First of all, let's focus on the concepts like **minor** and **cofactor** that are basis of the **Cofactor Expansion Method.**

Definition 10.5: Minor, Cofactor

det(M_{ij}), the i,j **minor** of the matrix, is defined as the determinant of the M_{ij} matrix, $(n-1)\times(n-1)$ matrix that results from deleting the i^{th} row and j^{th} column of the original $n \times n$ matrix.

C_{ij}, the i,j **cofactor** of the matrix, is defined by the formula:

$$C_{ij} = (-1)^{i+j} \det(M_{ij})$$

where $\det(M_{ij})$ is the i,j **minor** of the matrix.

Theorem 10.2: Cofactor Expansions

Let A be the $n \times n$ matrix. Then

(a) $\det(A) = a_{i1} C_{i1} + a_{i2} C_{i2} + \cdots + a_{in} C_{in}$ (Expand across row i)

(b) $\det(A) = a_{2j} C_{2j} + a_{2j} C_{2j} + \cdots + a_{nj} C_{nj}$ (Expand across column j)

where C_{ij} is denoted as the **cofactor** of a_{ij}.

- **Step 1.** Construct a 3×3 matrix for proving the **4th property**.

```
In[ ]:= A2 = Array[Subscript[a, ##] &, {3, 3}]; A2 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix}$$

- **Step 2.1.** Compute the determinant of a matrix **A2** using **column expansion**.
- Calculate the **minors** of matrix **A2** across the 1st column.

Definition 10.6: Determinant of a 2×2 Matrix

The determinant of a 2×2 matrix is equal to the difference between the product of the elements along the main diagonal and the product of the elements on the secondary diagonal.

```
In[ ]:= m11 = A2[[2, 2]] * A2[[3, 3]] - A2[[2, 3]] * A2[[3, 2]]
```

Out[ ]=

$$-a_{2,3}\,a_{3,2} + a_{2,2}\,a_{3,3}$$

```
In[ ]:= m21 = A2[[1, 2]] * A2[[3, 3]] - A2[[1, 3]] * A2[[3, 2]]
```

Out[ ]=

$$-a_{1,3}\,a_{3,2} + a_{1,2}\,a_{3,3}$$

```
In[ ]:= m31 = A2[[1, 2]] * A2[[2, 3]] - A2[[1, 3]] * A2[[2, 2]]
```

Out[ ]=

$$-a_{1,3}\,a_{2,2} + a_{1,2}\,a_{2,3}$$

- Use a for loop and the formula of column expansion given above to calculate the **det(A2.1).**

```
minorA2 = {{m11}, {m21}, {m31}};
detA21 = 0;
For[i = 1, i ≤ 3, i++,
 For[j = 1, j < 2, j++,
  detA21 = detA21 + A2[[i, j]] * (-1)^(i+j) * minorA2[[i, j]]]]
detA21
```

Out[ ]=

$$(-a_{1,3}\,a_{2,2} + a_{1,2}\,a_{2,3})\,a_{3,1} - a_{2,1}\,(-a_{1,3}\,a_{3,2} + a_{1,2}\,a_{3,3}) + a_{1,1}\,(-a_{2,3}\,a_{3,2} + a_{2,2}\,a_{3,3})$$

- **Step 2.2.** Compute the determinant of a matrix **A2** using **row expansion**.
- Calculate the **minors** of matrix **A2** across the 1st row using built-in command **Minors**.

```
In[ ]:= (mA2 = Minors[A2]) // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -a_{1,2}\,a_{2,1}+a_{1,1}\,a_{2,2} & -a_{1,3}\,a_{2,1}+a_{1,1}\,a_{2,3} & -a_{1,3}\,a_{2,2}+a_{1,2}\,a_{2,3} \\ -a_{1,2}\,a_{3,1}+a_{1,1}\,a_{3,2} & -a_{1,3}\,a_{3,1}+a_{1,1}\,a_{3,3} & -a_{1,3}\,a_{3,2}+a_{1,2}\,a_{3,3} \\ -a_{2,2}\,a_{3,1}+a_{2,1}\,a_{3,2} & -a_{2,3}\,a_{3,1}+a_{2,1}\,a_{3,3} & -a_{2,3}\,a_{3,2}+a_{2,2}\,a_{3,3} \end{pmatrix}$$

- Calculate the **cofactors** using the corresponding formula.

```
In[ ]:= C11 = (-1)^(1+1) mA2[[3, 3]]
```

Out[ ]=

$$-a_{2,3}\,a_{3,2}+a_{2,2}\,a_{3,3}$$

```
In[ ]:= C12 = (-1)^(1+2) mA2[[3, 2]]
```

Out[ ]=

$$a_{2,3}\,a_{3,1}-a_{2,1}\,a_{3,3}$$

```
In[ ]:= C13 = (-1)^(1+3) mA2[[3, 1]]
```

Out[ ]=

$$-a_{2,2}\,a_{3,1}+a_{2,1}\,a_{3,2}$$

- Calculate the **det(A2.2)** using the formula for row expansion given above.

```
cofactor = Transpose[{{C11}, {C12}, {C13}}];
detA22 = 0;
For[i = 1, i < 2, i++,
 For[j = 1, j ≤ 3, j++,
  detA22 = detA22 + A2[[i, j]] * cofactor[[i, j]]]]
detA22
```

Out[ ]=

$$a_{1,3}\,(-a_{2,2}\,a_{3,1}+a_{2,1}\,a_{3,2})+a_{1,2}\,(a_{2,3}\,a_{3,1}-a_{2,1}\,a_{3,3})+a_{1,1}\,(-a_{2,3}\,a_{3,2}+a_{2,2}\,a_{3,3})$$

- **Step 3.** Compute the determinant of A^T using the built-in command **Cofactor.**
- Find the transpose of the **A2.**

```
In[ ]:= A2t = Transpose[A2]; A2t // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} a_{1,1} & a_{2,1} & a_{3,1} \\ a_{1,2} & a_{2,2} & a_{3,2} \\ a_{1,3} & a_{2,3} & a_{3,3} \end{pmatrix}$$

- To use **Cofactor**, first load the Combinatorica Package.

```
In[ ]:= Needs["Combinatorica`"]
```

- Calculate the **det(A^T)** using the formula for row expansion.

```
In[ ]:= detA2t = 0;
        For[i = 1, i < 2, i++,
         For[j = 1, j ≤ 3, j++,
          detA2t = detA2t + A2t[[i, j]] * Cofactor[A2t, {i, j}]]]
        detA2t
```

Out[ ]=

$$(-a_{1,3}\,a_{2,2}+a_{1,2}\,a_{2,3})\,a_{3,1}+a_{2,1}\,(a_{1,3}\,a_{3,2}-a_{1,2}\,a_{3,3})+a_{1,1}\,(-a_{2,3}\,a_{3,2}+a_{2,2}\,a_{3,3})$$

◆ **Step 4.** Verify the results and check the **4th property.**

```
        FullSimplify[Det[A2] == detA2 1 == detA22]
```

Out[ ]=

```
        True
```

```
In[ ]:= FullSimplify[Det[A2t] == detA2t]
```

Out[ ]=

```
        True
```

```
In[ ]:= FullSimplify[detA21 == detA2t]
```

Out[ ]=

```
        True
```

```
In[ ]:= FullSimplify[Det[A2t] == Det[A2]]
```

Out[ ]=

```
        True
```

Example 10.3: Determinant and Its Properties | Part 2

Computing the determinant using row operations may be more efficient than cofactor expansion. Construct a 4×4 matrix and find its determinant using properties below.

Theorem 10.3: Influence of Row Operations on Determinants

Let A be a square matrix.

1. Let B be produced by interchanging two rows of A. Then $\mathbf{det}\,(A) = -\mathbf{det}\,(B)$.

2. Let B be produced by multiplying a row of A by a scalar c. Then $\mathbf{det}\,(A) = \frac{1}{c}\,\mathbf{det}\,(B)$.

3. Let B be produced by adding a multiple of one row of A to another. Then $\mathbf{det}\,(A) = \mathbf{det}\,(B)$.

This is also true if rows are replaced with columns.

Method 3. Use **Row Operations** to compute the determinant.

◆ **Step 1.** Construct a matrix with appropriate dimension and elements.

In[ ]:=
```
ClearAll["Global`*"]
SeedRandom[123];
A = RandomInteger[{-10, 10}, {4, 4}]; A // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 4 & 8 & -10 & -6 \\ -3 & -4 & -3 & 0 \\ 1 & -4 & 0 & 2 \\ -1 & 7 & -10 & -1 \end{pmatrix}$$

- **Step 2.** Convert the matrix A to echelon form using elementary row operations and reduce it to triangular form.
- Further steps are separated to keep track of the effect of the row operations on the determi nant.

$\frac{1}{4}\ R_1 \to R_1$	$\det(A) = 4\ \det(A_1)$

In[ ]:=
```
A1 = A;
A1[[1]] = 1/4 A1[[1]]; A1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 2 & -\frac{5}{2} & -\frac{3}{2} \\ -3 & -4 & -3 & 0 \\ 1 & -4 & 0 & 2 \\ -1 & 7 & -10 & -1 \end{pmatrix}$$

$R_2 + 3\ R_1 \to R_2$	$\det(A_1) = \det(A_2)$

In[ ]:=
```
A2 = A1;
A2[[2]] = A2[[2]] + 3 A2[[1]]; A2 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 2 & -\frac{5}{2} & -\frac{3}{2} \\ 0 & 2 & -\frac{21}{2} & -\frac{9}{2} \\ 1 & -4 & 0 & 2 \\ -1 & 7 & -10 & -1 \end{pmatrix}$$

$R_3 - R_1 \to R_3$	$\det(A_2) = \det(A_3)$

In[ ]:=
```
A3 = A2;
A3[[3]] = A3[[3]] - A3[[1]]; A3 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 2 & -\frac{5}{2} & -\frac{3}{2} \\ 0 & 2 & -\frac{21}{2} & -\frac{9}{2} \\ 0 & -6 & \frac{5}{2} & \frac{7}{2} \\ -1 & 7 & -10 & -1 \end{pmatrix}$$

$R_4 + R_1 \to R_4$ | $\det(A_3) = \det(A_4)$

In[ ]:=
```
A4 = A3;
A4[[4]] = A4[[4]] + A4[[1]]; A4 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 2 & -\frac{5}{2} & -\frac{3}{2} \\ 0 & 2 & -\frac{21}{2} & -\frac{9}{2} \\ 0 & -6 & \frac{5}{2} & \frac{7}{2} \\ 0 & 9 & -\frac{25}{2} & -\frac{5}{2} \end{pmatrix}$$

$\frac{1}{2} R_2 \to R_2$ | $\det(A_4) = 2 \det(A_5)$

In[ ]:=
```
A5 = A4;
A5[[2]] = 1/2 A5[[2]]; A5 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 2 & -\frac{5}{2} & -\frac{3}{2} \\ 0 & 1 & -\frac{21}{4} & -\frac{9}{4} \\ 0 & -6 & \frac{5}{2} & \frac{7}{2} \\ 0 & 9 & -\frac{25}{2} & -\frac{5}{2} \end{pmatrix}$$

$R_3 + 6 R_2 \to R_3$ | $\det(A_5) = \det(A_6)$

In[ ]:=
```
A6 = A5;
A6[[3]] = A6[[3]] + 6 A6[[2]]; A6 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 2 & -\frac{5}{2} & -\frac{3}{2} \\ 0 & 1 & -\frac{21}{4} & -\frac{9}{4} \\ 0 & 0 & -29 & -10 \\ 0 & 9 & -\frac{25}{2} & -\frac{5}{2} \end{pmatrix}$$

$R_4 - 9 R_2 \to R_4$ | $\det(A_7) = \det(A_8)$

In[ ]:=
```
A7 = A6;
A7[[4]] = A7[[4]] - 9 A7[[2]]; A7 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 2 & -\frac{5}{2} & -\frac{3}{2} \\ 0 & 1 & -\frac{21}{4} & -\frac{9}{4} \\ 0 & 0 & -29 & -10 \\ 0 & 0 & \frac{139}{4} & \frac{71}{4} \end{pmatrix}$$

$$-\frac{1}{29} R_3 \to R_3 \quad | \quad \det(A_7) = -29 \det(A_8)$$

In[ ]:=
```
A8 = A7;
A8[[3]] = -1/29 A8[[3]]; A8 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 2 & -\frac{5}{2} & -\frac{3}{2} \\ 0 & 1 & -\frac{21}{4} & -\frac{9}{4} \\ 0 & 0 & 1 & \frac{10}{29} \\ 0 & 0 & \frac{139}{4} & \frac{71}{4} \end{pmatrix}$$

$$R_4 - \frac{139}{4} R_3 \to R_4 \quad | \quad \det(A_8) = \det(A_9)$$

In[ ]:=
```
A9 = A8;
A9[[4]] = A9[[4]] - 139/4 A9[[3]]; A9 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 2 & -\frac{5}{2} & -\frac{3}{2} \\ 0 & 1 & -\frac{21}{4} & -\frac{9}{4} \\ 0 & 0 & 1 & \frac{10}{29} \\ 0 & 0 & 0 & \frac{669}{116} \end{pmatrix}$$

- **Step 3.** Since A_9 is a triangular form of A, find its determinant using the property of triangular matrices. Their determinant is equal to the product of its diagonal elements. (7th property from previous example)

In[ ]:=
```
detA9 = 1;
For[i = 1, i ≤ 4, i++,
  detA9 = detA9 * A9[[i, i]]];
detA9
```

Out[ ]=

$$\frac{669}{116}$$

◆ Verify the result.

```
In[ ]:= detA9 == Det[A9]
Out[ ]=
True
```

◆ **Step 4.** Substitute the value of **det(A_9)** and find **det(A)** using a **back substitution**.

```
In[ ]:= detA = 4 detA1;
detA1 = detA2;
detA2 = detA3;
detA3 = detA4;
detA4 = 2 detA5;
detA5 = detA6;
detA6 = detA7;
detA7 = -29 detA8;
detA8 = detA9;
Print[detA]
```

$$-1338$$

◆ **Step 5.** Verify the final result using built-in command.

```
In[ ]:= Det[A] == detA
Out[ ]=
True
```

Example 10.4: Applications of the Determinant

Cramer's Rule

Solve the following system of linear equations using **Cramer's rule**.

$$\begin{aligned} x_1 - 2x_2 + x_3 &= 0 \\ 2x_2 - 8x_3 &= 8 \\ -4x_1 + 5x_2 + 9x_3 &= -9 \end{aligned}$$

Theorem 10.4: Cramer's Rule

Let A be an invertible $n \times n$ matrix. Then the components of the unique solution to $Ax = b$ are given by

$$x_i = \frac{\det(A_i)}{\det(A)} \qquad \text{for } i = 1, 2, \ldots, n$$

◆ **Step 1.** The system is equivalent to $\boldsymbol{Ax = b}$, where A and b are defined as follows:

In[ ]:=
```
ClearAll["Global`*"]
A = {{1, -2, 1}, {0, 2, -8}, {-4, 5, 9}}; A // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & -2 & 1 \\ 0 & 2 & -8 \\ -4 & 5 & 9 \end{pmatrix}$$

In[ ]:=
```
b = {0, 8, -9}; b // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 \\ 8 \\ -9 \end{pmatrix}$$

- **Step 2.** Compute the determinant of the coefficient matrix **A.**

In[ ]:=
```
detA = Det[A]
```

Out[ ]=

2

- **Step 3.** Replace the 1st, 2nd and 3rd column values with the values of the answer column ***b*** and construct new three matrices, respectively.

In[ ]:=
```
A1 = A;
A1[[All, 1]] = b; A1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 & -2 & 1 \\ 8 & 2 & -8 \\ -9 & 5 & 9 \end{pmatrix}$$

In[ ]:=
```
A2 = A;
A2[[All, 2]] = b; A2 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 8 & -8 \\ -4 & -9 & 9 \end{pmatrix}$$

In[ ]:=
```
A3 = A;
A3[[All, 3]] = b; A3 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & -2 & 0 \\ 0 & 2 & 8 \\ -4 & 5 & -9 \end{pmatrix}$$

- **Step 4.** Compute their determinants.

In[ ]:=
```
detA1 = Det[A1]
```

Out[ ]=

58

In[ ]:= detA2 = Det[A2]

Out[ ]=

32

In[ ]:= detA3 = Det[A3]

Out[ ]=

6

◆ **Step 5.** Find the solution using the formula for **Cramer's rule**.

$$x1 = \frac{detA1}{detA}$$

Out[ ]=

29

$$x2 = \frac{detA2}{detA}$$

Out[ ]=

16

$$x3 = \frac{detA3}{detA}$$

Out[ ]=

3

In[ ]:= x = {x1, x2, x3}

Out[ ]=

{29, 16, 3}

◆ **Step 6.** Verify the obtained solution.

In[ ]:= A.x == b

Out[ ]=

True

In[ ]:= LinearSolve[A, b]

Out[ ]=

{29, 16, 3}

In[ ]:= % == x

Out[ ]=

True

Inverses from Determinants

Find the inverse of the following matrix using its determinant.

$$\begin{pmatrix} 3 & 1 & 0 \\ -1 & 2 & 1 \\ 0 & -1 & 2 \end{pmatrix}$$

Theorem 10.5: Inverse from Determinant

If A is an invertible matrix, then

$$A^{-1} = \frac{1}{\det(A)} \text{adj}(A)$$

Definition 10.7: Adjoint Matrix

The formal **adjoint** of A is the transpose of the matrix formed by the cofactors, C, of A and is denoted by **adj(A)**.

$$\text{adj}(A) = C^T = \begin{pmatrix} C_{11} & C_{21} & \cdots & C_{n1} \\ C_{12} & C_{22} & \cdots & C_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ C_{1n} & C_{2n} & \cdots & C_{nm} \end{pmatrix}$$

- **Step 1.** Construct the matrix A.

In[ ]:=
```
ClearAll["Global`*"]
A = {{3, 1, 0}, {-1, 2, 1}, {0, -1, 2}}; A // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 3 & 1 & 0 \\ -1 & 2 & 1 \\ 0 & -1 & 2 \end{pmatrix}$$

- **Step 2.** Find the determinant of A. Since **det(A)≠0**, the matrix A is invertible.

In[ ]:=
```
detA = Det[A]
```

Out[ ]=

```
17
```

- **Step 3.** Compute the nine cofactors for A as a matrix.

```
In[ ]:= Needs["Combinatorica`"]
        cofactorA = ConstantArray[0, {3, 3}];
        For[i = 1, i ≤ 3, i++,
         For[j = 1, j ≤ 3, j++,
          cofactorA[[i, j]] = Cofactor[A, {i, j}]]]
        cofactorA // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 5 & 2 & 1 \\ -2 & 6 & 3 \\ 1 & -3 & 7 \end{pmatrix}$$

- **Step 4.** Define the **adjoint matrix** corresponding to A.

```
In[ ]:= adjA = Transpose[cofactorA]; adjA // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 5 & -2 & 1 \\ 2 & 6 & -3 \\ 1 & 3 & 7 \end{pmatrix}$$

- **Step 5.** Finally, compute the inverse of A.

```
In[ ]:= (invA = 1/detA * adjA) // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} \frac{5}{17} & -\frac{2}{17} & \frac{1}{17} \\ \frac{2}{17} & \frac{6}{17} & -\frac{3}{17} \\ \frac{1}{17} & \frac{3}{17} & \frac{7}{17} \end{pmatrix}$$

- **Step 6.** Verify the solution.

```
In[ ]:= invA.A == A.invA == IdentityMatrix[3]
Out[ ]=
        True

In[ ]:= Inverse[A] == invA
Out[ ]=
        True
```

Summary

After completing this chapter, you should be able to

- perform LU factorization of a matrix in Mathematica
- find the determinant of a matrix in Mathematica
- practice applications of determinants in Mathematica.

- develop the habit of always checking your solutions for quality assurance.

Week 11: Eigenvalues and Eigenvectors

Determination and applications of eigenvalues and eigenvectors

Table of Contents

Commands list

- NullSpace
- CharacteristicPolynomial
- Eigenvalues
- Eigenvectors
- Eigensystem
- Factor
- DiagonalizableMatrixQ
- Diagonal
- Power
- MatrixPower

Prerequisite: Eigenvalues and Eigenvectors

The Wolfram Language includes funtions for finding the eigenvalues, eigenvectors, and eigensystems of matrices. There are also functions related to characteristic polynomials, diagonalization and many other matrix-related topics.

Eigenvalues and Eigenvectors:
https://reference.wolfram.com/language/tutorial/LinearAlgebra.html#9501

Example 11.1: Characteristic Polynomial and Equation

Determine the eigenvalues for the given matrix and eigenvectors associated with each eigenvalue using the **characteristic polynomial**:

$$A = \begin{pmatrix} 1 & -3 & 3 \\ 2 & -2 & 2 \\ 2 & 0 & 0 \end{pmatrix}$$

Definition 11.1: Eigenvector, Eigenvalue

Let A be an $n \times n$ matrix. Then a nonzero vector u is an **eigenvector** of A if there exists a scalar λ such that

$$\mathbf{Au} = \lambda \mathbf{u}$$

where the scalar λ is called an **eigenvalue** of A.

⚠ Note that an eigenvalue λ can be zero, but an eigenvactor **u** must be a nonzero vector.

- **Step 1.** Define the matrix A.

```
In[ ]:= ClearAll["Global`*"]
        A = {{1, -3, 3}, {2, -2, 2}, {2, 0, 0}}; MatrixForm[A]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & -3 & 3 \\ 2 & -2 & 2 \\ 2 & 0 & 0 \end{pmatrix}$$

- **Step 2.** Determine the eigenvalues of A by calculating the characteristic polynomial.

Theorem 11.1: How to use determinants to find eigenvalues

Let A be an $n \times n$ matrix. Then λ is an eigenvalue of A if and only if $\mathbf{det}(A - \lambda I_n) = 0$.

where the polynomial from $\mathbf{det}(A - \lambda I_n)$ is called the **characteristic polynomial** of A, and the equation $\mathbf{det}(A - \lambda I_n) = 0$ is called the **characteristic equation**.

- Form a new matrix by subtracting λ from diagonal elements of A such that $\mathbf{P} = A - \lambda I_3$.

```
In[ ]:= Poly = A - λ * IdentityMatrix[3]; Poly // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1-\lambda & -3 & 3 \\ 2 & -2-\lambda & 2 \\ 2 & 0 & -\lambda \end{pmatrix}$$

- The eigenvalues for a matrix A are given by the roots of the characteristic equation.

In[ ]:=
```
eqn = Det[Poly] == 0
```

Out[ ]=

$$2\lambda - \lambda^2 - \lambda^3 == 0$$

In[ ]:=
```
Solve[eqn, λ]
```

Out[ ]=

$$\{\{\lambda \to -2\}, \{\lambda \to 0\}, \{\lambda \to 1\}\}$$

Theorem 11.2: Eigenvectors

The corresponding eigenvectors are found by solving the homogeneous system:

$$(\mathbf{A} - \lambda \mathbf{I_n})\,\mathbf{u} = \mathbf{0}$$

- **Step 3.1.** Find the eigenvectors associated with $\lambda_1 = -2$ by solving the corresponding homogeneous system: $(\mathbf{A} + 2\,\mathbf{I_3})\,\mathbf{u_1} = \mathbf{0}$.
- **Step 3.1.1.** Construct the augmented matrix of the system, **AugMat= [A1|b].**
- Construct the coefficient matrix $\mathbf{A1} = \mathbf{A} + 2\,\mathbf{I_3}$.

In[ ]:=
```
A1 = A + 2 * IdentityMatrix[3]; A1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 3 & -3 & 3 \\ 2 & 0 & 2 \\ 2 & 0 & 2 \end{pmatrix}$$

- Column matrix *b* with right-hand-side constants:

In[ ]:=
```
b = {{0}, {0}, {0}}; MatrixForm[b]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

- Augmented matrix:

In[ ]:=
```
AugMat1 = ArrayFlatten[{{A1, b}}]; MatrixForm[AugMat1]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 3 & -3 & 3 & 0 \\ 2 & 0 & 2 & 0 \\ 2 & 0 & 2 & 0 \end{pmatrix}$$

- **Step 3.1.2.** Transform the augmented matrix to echelon form.
- $\mathbf{R_1} \longleftrightarrow \mathbf{R_2}$

```
In[ ]:= AugMat1[[{1, 2}]] = AugMat1[[{2, 1}]]; AugMat1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 & 0 & 2 & 0 \\ 3 & -3 & 3 & 0 \\ 2 & 0 & 2 & 0 \end{pmatrix}$$

- $-\frac{3}{2}\,R_1 + R_2 \to R_2$

```
In[ ]:= AugMat1[[2]] = -3/2 AugMat1[[1]] + AugMat1[[2]]; AugMat1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 & 0 & 2 & 0 \\ 0 & -3 & 0 & 0 \\ 2 & 0 & 2 & 0 \end{pmatrix}$$

- $-R_1 + R_3 \to R_3$

```
In[ ]:= AugMat1[[3]] = -AugMat1[[1]] + AugMat1[[3]]; AugMat1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 & 0 & 2 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

- **Step 3.1.3.** Perform the back substitution to find the eigenvector. Take $x_3 = s$:

```
In[ ]:= x3 = s;
        eq0 = 2 x1 + 2 x3 == 0;
        eq1 = -3 x2 == 0;
        soln = Solve[{eq0, eq1}, {x1, x2}]
```

Out[ ]=

```
{{x1 → -s, x2 → 0}}
```

- The general solution to $\mathbf{u}_1$:

```
In[ ]:= u1 = {x1, x2, x3} /. soln[[1]]; u1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -s \\ 0 \\ s \end{pmatrix}$$

Definition 11.2: Eigenspace

Each distinct eigenvalue λ has its own associated eigenspace. The subspace of all eigenvectors associated with that eigenvalue λ, together with the zero vector, is called the **eigenspace** of λ.

- Basis for eigenspace of $\lambda_1 = -2$:

In[ ]:=
```
u1 = u1 /. s → 1; u1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}$$

- **Step 3.2.** Find the eigenvectors associated with $\lambda_2 = 1$ by solving the corresponding homogeneous system: $(A - I_3)\, u_2 = 0$.
- **Step 3.2.1.** Construct the augmented matrix of the system, **AugMat= [A2|b].**
- Construct the coefficient matrix $A2 = A - I_3$.

In[ ]:=
```
A2 = A - IdentityMatrix[3]; A2 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 & -3 & 3 \\ 2 & -3 & 2 \\ 2 & 0 & -1 \end{pmatrix}$$

- Augmented matrix:

In[ ]:=
```
AugMat2 = ArrayFlatten[{{A2, b}}]; MatrixForm[AugMat2]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 & -3 & 3 & 0 \\ 2 & -3 & 2 & 0 \\ 2 & 0 & -1 & 0 \end{pmatrix}$$

- **Step 3.2.2.** Transform the augmented matrix to echelon form.
- $R_1 \longleftrightarrow R_3$

In[ ]:=
```
AugMat2[[{1, 3}]] = AugMat2[[{3, 1}]]; AugMat2 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 & 0 & -1 & 0 \\ 2 & -3 & 2 & 0 \\ 0 & -3 & 3 & 0 \end{pmatrix}$$

- $-R_1 + R_2 \to R_2$

In[ ]:=
```
AugMat2[[2]] = -AugMat2[[1]] + AugMat2[[2]]; AugMat2 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 & 0 & -1 & 0 \\ 0 & -3 & 3 & 0 \\ 0 & -3 & 3 & 0 \end{pmatrix}$$

- $-R_2 + R_3 \to R_3$

In[ ]:=
```
AugMat2[[3]] = -AugMat2[[2]] + AugMat2[[3]]; AugMat2 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 & 0 & -1 & 0 \\ 0 & -3 & 3 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

◆ **Step 3.2.3.** Perform the back substitution to find the eigenvector. Take $x_3 = 2\,s$:

```
In[ ]:= Clear[x1, x2, x3]
        x3 = 2 s;
        eq0 = 2 x1 - x3 == 0;
        eq1 = -3 x2 + 3 x3 == 0;
        soln = Solve[{eq0, eq1}, {x1, x2}]
Out[ ]=
        {{x1 → s, x2 → 2 s}}
```

◆ The general solution to $\mathbf{u}_2$:

```
In[ ]:= u2 = {x1, x2, x3} /. soln[[1]]; u2 // MatrixForm
Out[ ]//MatrixForm=
        ( s   )
        ( 2 s )
        ( 2 s )
```

◆ Basis for eigenspace of $\lambda_2 = 1$:

```
In[ ]:= u2 = u2 /. s → 1; u2 // MatrixForm
Out[ ]//MatrixForm=
        ( 1 )
        ( 2 )
        ( 2 )
```

◆ **Step 3.2.** Find the eigenvectors associated with $\lambda_3 = 0$ by solving the homogeneous system: $(\mathbf{A} - 0\,\mathbf{I}_3)\,\mathbf{u}_3 = \mathbf{A}\mathbf{u} = \mathbf{0}$.

◆ **Step 3.2.1.** Construct the matrix $\mathbf{A3} = \mathbf{A} - 0\,\mathbf{I}_3 = \mathbf{A}$.

```
In[ ]:= A3 = A - 0 * IdentityMatrix[3]; A3 // MatrixForm
Out[ ]//MatrixForm=
        ( 1  -3  3 )
        ( 2  -2  2 )
        ( 2   0  0 )
```

◆ **Step 3.3.2.** Use the **NullSpace** function to find the eigenvector.

◆ The general solution to $\mathbf{u}_3$:

```
In[ ]:= u3 = MatrixForm[Transpose[NullSpace[A3] * s]] == MatrixForm[Transpose[NullSpace[A3]]] * s
Out[ ]=
        ( 0 )     ( 0 )
        ( s )  == s ( 1 )
        ( s )     ( 1 )
```

◆ Basis for eigenspace of $\lambda_3 = 0$:

```
In[ ]:= u3 = NullSpace[A3]; Transpose[u3] // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

◆ **Step 4.** Verify the results using the built-in functions.

```
In[ ]:= checkPoly = CharacteristicPolynomial[A, λ]
```

Out[ ]=

$$2\lambda - \lambda^2 - \lambda^3$$

```
In[ ]:= Solve[checkPoly == 0, λ]
```

Out[ ]=

$$\{\{\lambda \to -2\}, \{\lambda \to 0\}, \{\lambda \to 1\}\}$$

```
In[ ]:= Eigenvalues[A]
```

Out[ ]=

$$\{-2, 1, 0\}$$

```
In[ ]:= v = Eigenvectors[A];
        {MatrixForm[v[[1]]], MatrixForm[v[[2]]], MatrixForm[v[[3]]]}
```

Out[ ]=

$$\left\{\begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}\right\}$$

```
In[ ]:= sys = Eigensystem[A];
        {sys[[1]], MatrixForm[sys[[2, 1]]], MatrixForm[sys[[2, 2]]], MatrixForm[sys[[2, 3]]]}
```

Out[ ]=

$$\left\{\{-2, 1, 0\}, \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 2 \\ 2 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}\right\}$$

Example 11.2: Multiplicity of an Eigenvalue

Find the eigenvalues and a basis for each eigenspace of the given matrix A:

$$A = \begin{pmatrix} 4 & 4 & -2 \\ 1 & 4 & -1 \\ 3 & 6 & -1 \end{pmatrix}$$

◆ **Step 1.** Define the matrix A.

```
In[ ]:= ClearAll["Global`*"]
        A = {{4, 4, -2}, {1, 4, -1}, {3, 6, -1}}; MatrixForm[A]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 4 & 4 & -2 \\ 1 & 4 & -1 \\ 3 & 6 & -1 \end{pmatrix}$$

- **Step 2.** Determine the eigenvalues of A by calculating the characteristic polynomial.
- The characteristic polynomial of A is $\mathbf{P = det(A - \lambda I_3)}$.

```
In[ ]:= poly = A - λ * IdentityMatrix[3]; poly // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 4-\lambda & 4 & -2 \\ 1 & 4-\lambda & -1 \\ 3 & 6 & -1-\lambda \end{pmatrix}$$

```
In[ ]:= Poly = Det[poly]
```

Out[ ]=

$$12 - 16\,\lambda + 7\,\lambda^2 - \lambda^3$$

- **Step 3.** Define the characteristic equation of A as $\mathbf{det(A - \lambda I_3) = 0}$.

```
In[ ]:= eqn = Poly == 0
```

Out[ ]=

$$12 - 16\,\lambda + 7\,\lambda^2 - \lambda^3 == 0$$

- **Step 4.** The eigenvalues for a matrix A are given by the roots of the characteristic equation.
- Factorise the characteristic equation.

```
In[ ]:= Factor[eqn]
```

Out[ ]=

$$-\left((-3+\lambda)\ (-2+\lambda)^2\right) == 0$$

- From the factored form we see that the matrix A has **two distinct eigenvalues**, $\lambda_1 = 2$ (**multiplicity 2**) and $\lambda_2 = 3$ (**multiplicity 1**). Confirm the results by solving the characteristic equation for λ.

```
In[ ]:= Solve[eqn, λ]
```

Out[ ]=

$$\{\{\lambda \to 2\}, \{\lambda \to 2\}, \{\lambda \to 3\}\}$$

Definition 11.3: Multiplicity

The *multiplicity* of an eigenvalue is equal to its factor's exponent. In general, for a polynomial $P(x)$, a root α of $P(x) = 0$ has **muliplicity** r if $P(x) = (x - \alpha)^r\, Q(x)$ with $Q(\alpha) \neq 0$.

- **Step 5.1.** Find the eigenvectors associated with $\lambda_1 = 2$ by solving the corresponding homogeneous system: $(A - 2\,I_3)\,u_1 = 0$.
- **Step 5.1.1.** Construct the augmented matrix of the system, **AugMat= [A1|b].**
- Construct the coefficient matrix $A1 = A + 2\,I_3$.

```
In[ ]:= A1 = A - 2 * IdentityMatrix[3]; A1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 & 4 & -2 \\ 1 & 2 & -1 \\ 3 & 6 & -3 \end{pmatrix}$$

- Column matrix *b* with right-hand-side constants:

```
In[ ]:= b = {{0}, {0}, {0}}; MatrixForm[b]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

- Augmented matrix:

```
In[ ]:= AugMat1 = ArrayFlatten[{{A1, b}}]; MatrixForm[AugMat1]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 & 4 & -2 & 0 \\ 1 & 2 & -1 & 0 \\ 3 & 6 & -3 & 0 \end{pmatrix}$$

- **Step 5.1.2.** Transform the augmented matrix to echelon form.
- $R_2 \longleftrightarrow R_1$

```
In[ ]:= AugMat1[[{1, 2}]] = AugMat1[[{2, 1}]]; AugMat1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 2 & -1 & 0 \\ 2 & 4 & -2 & 0 \\ 3 & 6 & -3 & 0 \end{pmatrix}$$

- $-2\,R_1 + R_2 \to R_2$

```
In[ ]:= AugMat1[[2]] = -2 * AugMat1[[1]] + AugMat1[[2]]; AugMat1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 3 & 6 & -3 & 0 \end{pmatrix}$$

- $-3\,R_1 + R_3 \to R_3$

```
In[ ]:= AugMat1[[3]] = -3 * AugMat1[[1]] + AugMat1[[3]]; AugMat1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

◆ **Step 5.1.3.** Perform the back substitution to find the eigenvector. Let $x_2 = s_1$ and $x_3 = s_2$:

```
In[ ]:= x3 = s2;
        x2 = s1;
        eqn = x1 + 2 x2 - x3 == 0;
        soln = Solve[eqn, x1]
Out[ ]=
        {{x1 → -2 s1 + s2}}
```

◆ The general solution to $\mathbf{u}_1$:

```
In[ ]:= u1 = {x1, x2, x3} /. soln[[1]];
        u11 = u1 /. {s1 → 1, s2 → 0};
        u12 = u1 /. {s1 → 0, s2 → 1};
        MatrixForm[u1] == s1 * MatrixForm[u11] + s2 * MatrixForm[u12]
Out[ ]=
```

$$\begin{pmatrix} -2\,s_1 + s_2 \\ s_1 \\ s_2 \end{pmatrix} == \begin{pmatrix} -2 \\ 1 \\ 0 \end{pmatrix} s_1 + \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} s_2$$

◆ Basis for eigenspace of $\lambda_1 = 2$:

```
In[ ]:= u1 = { MatrixForm[u11] , MatrixForm[u12] }
Out[ ]=
```

$$\left\{\begin{pmatrix} -2 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}\right\}$$

⚠ Note that for the eigenvalue $\lambda_1 = 2$, which has **multiplicity of 2**, its associated eigenspace has **dimension 2** (i.e., a plane).

Theorem 11.3: Multiplicity and Eigenspace Dimension

> Let A be a square matrix with eigenvalue λ. Then the dimension of the associated eigenspace is less that or equal to the multiplicity of λ.

◆ **Step 5.2.** Find the eigenvectors associated with $\lambda_2 = 3$ by solving the corresponding homogeneous system: $(\mathbf{A} - 3\,\mathbf{I}_3)\,\mathbf{u}_3 = \mathbf{0}$.

◆ **Step 5.2.1.** Construct the augmented matrix of the system, **AugMat= [A2|b].**

◆ Construct the coefficient matrix $\mathbf{A2} = \mathbf{A} - 3\,\mathbf{I}_3$.

```
In[ ]:= A2 = A - 3 IdentityMatrix[3]; A2 // MatrixForm
Out[ ]//MatrixForm=
```

$$\begin{pmatrix} 1 & 4 & -2 \\ 1 & 1 & -1 \\ 3 & 6 & -4 \end{pmatrix}$$

◆ Augmented matrix:

```
In[ ]:= AugMat2 = ArrayFlatten[{{A2, b}}]; MatrixForm[AugMat2]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 4 & -2 & 0 \\ 1 & 1 & -1 & 0 \\ 3 & 6 & -4 & 0 \end{pmatrix}$$

◆ **Step 5.2.2.** Transform the augmented matrix to echelon form.

◆ $-1\,R_1 + R_2 \to R_2$

```
In[ ]:= AugMat2[[2]] = -1 * AugMat2[[1]] + AugMat2[[2]]; AugMat2 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 4 & -2 & 0 \\ 0 & -3 & 1 & 0 \\ 3 & 6 & -4 & 0 \end{pmatrix}$$

◆ $-3\,R_1 + R_3 \to R_3$

```
In[ ]:= AugMat2[[3]] = -3 * AugMat2[[1]] + AugMat2[[3]]; AugMat2 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 4 & -2 & 0 \\ 0 & -3 & 1 & 0 \\ 0 & -6 & 2 & 0 \end{pmatrix}$$

◆ $-2\,R_2 + R_3 \to R_3$

```
In[ ]:= AugMat2[[3]] = -2 * AugMat2[[2]] + AugMat2[[3]]; AugMat2 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 4 & -2 & 0 \\ 0 & -3 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

◆ **Step 5.2.3.** Perform the back substitution to find the eigenvector. Take $x_3 = 3\,s$:

```
In[ ]:= Clear[x1, x2, x3]
        x3 = 3 s;
        eq0 = x1 + 4 x2 - 2 x3 == 0;
        eq1 = -3 x2 + x3 == 0;
        soln = Solve[{eq0, eq1}, {x1, x2}]
```

Out[ ]=

```
{{x1 → 2 s, x2 → s}}
```

◆ The general solution to $\mathbf{u}_2$:

```
In[ ]:= u2 = {x1, x2, x3} /. soln[[1]]; u2 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2\,s \\ s \\ 3\,s \end{pmatrix}$$

◆ Basis for eigenspace of $\lambda_2 = 3$:

```
In[ ]:= u2 = u2 /. s → 1; u2 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix}$$

◆ **Step 6.** Verify the results using the built-in functions.

```
In[ ]:= checkPoly = CharacteristicPolynomial[A, λ]
```

Out[ ]=

$$12 - 16\,\lambda + 7\,\lambda^2 - \lambda^3$$

```
In[ ]:= Solve[checkPoly == 0, λ]
```

Out[ ]=

```
{{λ → 2}, {λ → 2}, {λ → 3}}
```

```
In[ ]:= Eigenvalues[A]
```

Out[ ]=

```
{3, 2, 2}
```

```
In[ ]:= v = Eigenvectors[A];
        {MatrixForm[v[[1]]], MatrixForm[v[[2]]], MatrixForm[v[[3]]]}
```

Out[ ]=

$$\left\{\begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} -2 \\ 1 \\ 0 \end{pmatrix}\right\}$$

```
In[ ]:= sys = Eigensystem[A];
        {sys[[1]], MatrixForm[sys[[2, 1]]], MatrixForm[sys[[2, 2]]], MatrixForm[sys[[2, 3]]]}
```

Out[ ]=

$$\left\{\{3, 2, 2\}, \begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} -2 \\ 1 \\ 0 \end{pmatrix}\right\}$$

Example 11.3: Diagonalization

Definition 11.4: Diagonalizable Matrix

An $n \times n$ matrix A is **diagonalizable** if there exist $n \times n$ matrices D and P, with D diagonal and P invertible, such that

$$\mathbf{A = PDP^{-1}}$$

where the columns of P are the eigenvectors of the matrix A, such that $P = [u_1 \ldots u_n]$ and the diagonal entries of D given by the corresponding eigenvalues $\lambda_1, \ldots, \lambda_n$.

⚠ Note that the order of the eigenvalues in D does not matter, as long as it matches the order of the

corresponding eigenvectors in P.

Theorem 11.4: Conditions for a Matrix to be Diagonalizable

1. An $n \times n$ matrix A is diagonalizable if and only if A has eigenvectors that form a basis for $\mathbf{R}^n$.
2. If A has n linearly independent eigenvectors $u_1, \ldots, u_n$, then A is diagonalizable.
3. Suppose that an $n \times n$ matrix A has only real eigenvalues. Then A is diagonalizable if and only if the dimension of each eigenspace is equal to the multiplicity of the corresponding eigenvalue.
4. If A is an $n \times n$ matrix with n distinct real eigenvalues, then A is always diagonalizable.

Theorem 11.5: Eigenvalues and Linear Independence

If $\{\lambda_1, \ldots, \lambda_k\}$ are distinct eigenvalues of a matrix A, then any set of associated eigenvectors $\{\mathbf{u}_1, \ldots, \mathbf{u}_k\}$ is linearly independent.

Example 11.3.1. If possible, diagonalize the given matrix A:

$$A = \begin{pmatrix} 0 & 2 & -1 \\ 1 & -1 & 0 \\ 1 & -2 & 0 \end{pmatrix}$$

- **Step 1.** Define the matrix A.

```
In[ ]:= ClearAll["Global`*"]
        A = {{0, 2, -1}, {1, -1, 0}, {1, -2, 0}}; MatrixForm[A]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 & 2 & -1 \\ 1 & -1 & 0 \\ 1 & -2 & 0 \end{pmatrix}$$

- **Step 2.** Find the eigenvalues of the matrix A.

```
In[ ]:= Factor[CharacteristicPolynomial[A, λ]]
```

Out[ ]=

$$-\left((-1+\lambda)\ (1+\lambda)^2\right)$$

```
In[ ]:= Solve[Det[A - λ * IdentityMatrix[3]] == 0, λ]
```

Out[ ]=

$$\{\{\lambda \to -1\}, \{\lambda \to -1\}, \{\lambda \to 1\}\}$$

```
In[ ]:= Eigenvalues[A]
```

Out[ ]=

$$\{-1, -1, 1\}$$

- Results show that the matrix A has **two distinct eigenvalues**, $\lambda_1 = -1$ (multiplicity 2) and $\lambda_2 = 1$ (multiplicity 1).

◆ **Step 3.** Find the corresponding eigenvectors.

◆ Basis for eigenspace of $\lambda_1 = -1$:

```
In[ ]:= λ1 = -1;
        NullSpace[A - λ1 IdentityMatrix[3]] // Transpose // MatrixForm
```

```
Out[ ]//MatrixForm=
        ⎛0⎞
        ⎜1⎟
        ⎝2⎠
```

⚠ Note that although the eigenvalue $\lambda_1 = -1$ has **multiplicity of 2**, its associated eigenspace has **dimension 1** (i.e., a line).

◆ Basis for eigenspace of $\lambda_2 = 1$:

```
In[ ]:= λ2 = 1;
        NullSpace[A - λ2 IdentityMatrix[3]] // Transpose // MatrixForm
```

```
Out[ ]//MatrixForm=
        ⎛2⎞
        ⎜1⎟
        ⎝0⎠
```

◆ **Step 4.** Check the matrix for diagonalizability based on the eigenvalues and eigenvectors.

For the given matrix A, the dimension of eigenspace of $\lambda_1 = -1$ is **less** than the multiplicity of the corresponding eigenvalue, therefore, according to the **Theorem 11.4.3** and **Theorem 11.4.4**, the given matrix A is **non-diagonalizable**.

◆ **Step 5.** Verify the conclusion using **DiagonalizableMatrixQ**.

```
In[ ]:= DiagonalizableMatrixQ[A]
```

```
Out[ ]=
        False
```

Example 11.3.2. If possible, find matrices P and D to diagonalize the given matrix A:

$$A = \begin{pmatrix} 1 & -4 & 3 \\ 0 & 7 & 1 \\ 0 & 0 & 2 \end{pmatrix}$$

◆ **Step 1.** Define the matrix A.

```
In[ ]:= ClearAll["Global`*"]
        A = {{1, -4, 3}, {0, 7, 1}, {0, 0, 2}}; MatrixForm[A]
```

```
Out[ ]//MatrixForm=
        ⎛1  -4  3⎞
        ⎜0   7  1⎟
        ⎝0   0  2⎠
```

- **Step 2.** Find the eigenvalues of the matrix A.

```
In[ ]:= Factor[CharacteristicPolynomial[A, λ]]
Out[ ]=
-((-7 + λ) (-2 + λ) (-1 + λ))

In[ ]:= Solve[Det[A - λ * IdentityMatrix[3]] == 0, λ]
Out[ ]=
{{λ → 1}, {λ → 2}, {λ → 7}}

In[ ]:= Eigenvalues[A]
Out[ ]=
{7, 2, 1}
```

- Results show that the matrix A has three distinct eigenvalues, $\lambda_1 = 7$, $\lambda_2 = 2$, and $\lambda_3 = 1$.
- **Step 3.** Find the corresponding eigenvectors.
- Basis for eigenspace of $\lambda_1 = 7$:

```
In[ ]:= λ1 = 7;
        u1 = NullSpace[A - λ1 IdentityMatrix[3]]; u1 // Transpose // MatrixForm
Out[ ]//MatrixForm=
```

$$\begin{pmatrix} -2 \\ 3 \\ 0 \end{pmatrix}$$

- Basis for eigenspace of $\lambda_2 = 2$:

```
In[ ]:= λ2 = 2;
        u2 = NullSpace[A - λ2 IdentityMatrix[3]]; u2 // Transpose // MatrixForm
Out[ ]//MatrixForm=
```

$$\begin{pmatrix} 19 \\ -1 \\ 5 \end{pmatrix}$$

- Basis for eigenspace of $\lambda_3 = 1$:

```
In[ ]:= λ3 = 1;
        u3 = NullSpace[A - λ3 IdentityMatrix[3]]; u3 // Transpose // MatrixForm
Out[ ]//MatrixForm=
```

$$\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

- **Step 4.** Check the matrix for diagonalizability based on the eigenvalues and eigenvectors.
- Since the matrix A has three distinct eigenvalues and three eigenvectors corresponsing to these eigenvalues, therefore by **Theorem 11.4.3** the set of eigenvectors is linearly idepen-dent and thus forms a basis for $\mathbf{R}^3$.
- According to the **Theorem 11.4.1** and **Theorem 11.4.4**, the given matrix A is **diagonalizable**.

- **Step 5.** Verify the conclusion.
- The following nonzero determinant implies that the eigenvectors are linearly independent.

```
In[ ]:= Det[Eigenvectors[A]]
Out[ ]=
15
```

```
In[ ]:= DiagonalizableMatrixQ[A]
Out[ ]=
True
```

- **Step 6.** Define the matrices P and D based on the **Definition 11.4.**

```
In[ ]:= D1 = DiagonalMatrix[{λ1, λ2, λ3}]; D1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 7 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

```
In[ ]:= P = Transpose[{u1[[1]], u2[[1]], u3[[1]]}]; P // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -2 & 19 & 1 \\ 3 & -1 & 0 \\ 0 & 5 & 0 \end{pmatrix}$$

- **Step 7.** If possible, obtain the inverse of matrix P.
- Calculate the determinant of P.

```
In[ ]:= Det[P] ≠ 0
Out[ ]=
True
```

- The nonzero determinant implies that the matrix P is invertible.

```
In[ ]:= InvP = Inverse[P]; InvP // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 & \frac{1}{3} & \frac{1}{15} \\ 0 & 0 & \frac{1}{5} \\ 1 & \frac{2}{3} & -\frac{11}{3} \end{pmatrix}$$

- **Step 7.1.** Show that $A = PDP^{-1}$.

```
In[ ]:= A == Dot[P, D1, InvP]
Out[ ]=
True
```

- **Step 7.2.** Alternatively, since P is invertible, show that **AP = PD** .

In[ ]:=
```
A.P == P.D1
```
Out[ ]=
```
True
```

Example 11.4: Matrix Powers

Definition 11.5: Matrix Powers

Suppose that $n \times n$ matrix A is diagonalizable with $A = PDP^{-1}$. Then the k^{th} power of A is defined to be

$$A^k = PD^k P^{-1}$$

Find A^5 via diagonalization.

$$A = \begin{pmatrix} \frac{1}{3} & \frac{2}{9} \\ \frac{2}{3} & \frac{7}{9} \end{pmatrix}$$

- **Step 1.** Define the matrix A.

In[ ]:=
```
ClearAll["Global`*"]
A = {{1/3, 2/9}, {2/3, 7/9}}; MatrixForm[A]
```
Out[ ]//MatrixForm=

$$\begin{pmatrix} \frac{1}{3} & \frac{2}{9} \\ \frac{2}{3} & \frac{7}{9} \end{pmatrix}$$

- **Step 2.** Find the eigenvalues and corresponding eigenvectors of the matrix A.

In[ ]:=
```
λ = Eigenvalues[A]
```
Out[ ]=

$$\left\{1, \frac{1}{9}\right\}$$

In[ ]:=
```
u = Eigenvectors[A];
{MatrixForm[u[[1]]], MatrixForm[u[[2]]]}
```
Out[ ]=

$$\left\{\begin{pmatrix} \frac{1}{3} \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix}\right\}$$

- **Step 3.** Check the matrix for diagonalizability based on the eigenvalues and eigenvectors.
- The following nonzero determinant implies that the eigenvectors are linearly independent.

In[ ]:=
```
Det[Eigenvectors[A]]
```
Out[ ]=
$$\frac{4}{3}$$

In[ ]:=
```
DiagonalizableMatrixQ[A]
```
Out[ ]=
```
True
```

- **Step 4.** Define the matrices P and D based on the **Definition 11.4.**

In[ ]:=
```
D1 = DiagonalMatrix[{λ[[1]], λ[[2]]}]; D1 // MatrixForm
```
Out[ ]//MatrixForm=
$$\begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{9} \end{pmatrix}$$

In[ ]:=
```
P = Transpose[u]; P // MatrixForm
```
Out[ ]//MatrixForm=
$$\begin{pmatrix} \frac{1}{3} & -1 \\ 1 & 1 \end{pmatrix}$$

- **Step 5.** If possible, obtain the inverse of matrix P.
- Calculate the determinant of P.

In[ ]:=
```
Det[P] ≠ 0
```
Out[ ]=
```
True
```

- The nonzero determinant implies that the matrix P is invertible.

In[ ]:=
```
InvP = Inverse[P]; InvP // MatrixForm
```
Out[ ]//MatrixForm=
$$\begin{pmatrix} \frac{3}{4} & \frac{3}{4} \\ -\frac{3}{4} & \frac{1}{4} \end{pmatrix}$$

- **Step 6.** Calculate the 5^{th} power of the diagonal matrix.

Theorem 11.5: Powers of a Diagonal Matrix

> The k^{th} power of a diagonal matrix D is a diagonal matrix whose diagonal entries are the k^{th} powers of the corresponsing entries of D.

- Diagonal entries of D:

In[ ]:=
```
d = Diagonal[D1]
```
Out[ ]=
$$\left\{1, \frac{1}{9}\right\}$$

```
In[ ]:= D5 = DiagonalMatrix[Power[d, 5]]; D5 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 0 \\ 0 & \frac{1}{59\,049} \end{pmatrix}$$

- **Step 6.** Compute A^5 based on the **Definition 11.5.** Calculate the 5^{th} power of the diagonal matrix.

```
In[ ]:= A5 = P.D5.InvP; A5 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} \frac{4921}{19\,683} & \frac{14\,762}{59\,049} \\ \frac{14\,762}{19\,683} & \frac{44\,287}{59\,049} \end{pmatrix}$$

- **Step 7.** Verify the obtained solution.

```
In[ ]:= MatrixPower[A, 5] == A5
```

Out[ ]=

```
True
```

Summary

After completing this chapter, you should be able to

- develop SOPs to find the eigensystem (eigenvalues and eigenvectors) of a given square matrix.
- perform step-by-step matrix diagonalization in Mathematica.
- perform matrix power operations in Mathematica.
- develop the habit of always checking your solutions for quality assurance.

Week 12: Linear Algebra and Geometry

Vectors, Vector Operations, and Linear Transformations

Table of Contents

8. **Summary**

Commands list

- Arrowheads
- Arrow
- Show
- Graphics
- Point
- Line
- Dot
- Norm
- EuclideanDistance
- VectorAngle
- Expand
- BezierCurve

Prerequisite: Operations on Vectors | Plane Geometry

The Wolfram Language represents vectors as a lists, and never needs to distinguish between row and column cases. Vectors in the Wolfram Language can always mix numbers and arbitrary symbolic or algebraic elements. There are varoius built-in functions for constructing and displaying the vectors, performing operations on vectors, and for vector analysis.

Operations on Vectors:
https://reference.wolfram.com/language/guide/OperationsOnVectors.html

The Wolfram Language provides fully integrated support for plane geometry, including basic regions such as points, lines, triangles, and disks; functions for computing basic properties such as arc length and area; and nearest points to solvers to find the intersection of regions or integrals over regions. It is a powerful tool of visualization of geometrical figures and graphical images, and consists of explicit list of primitives, directives, wrappers, and options.

Plane Geometry:
https://reference.wolfram.com/language/guide/PlaneGeometry.html

Example 12.1: Vectors and Vector Operations

Definition 12.1: Vector and R^n

A **vector** is an object that has both magnitude and direction. It can be represented by an ordered list of real numbers $u_1,\ u_2,\ \ldots,\ u_n$ expressed as

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix}$$

or as $\mathbf{u} = (u_1,\ u_2,\ \ldots,\ u_n)$.

The set of all vectors with n entries is denoted by $\mathbf{R}^n$ (real coordinate space of dimension n).

Definition 12.2: Vector Arithmetic, Scalar, Euclidean Space

Let u and v be vectors in $\mathbf{R}^n$ given by

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} \quad \text{and} \quad \mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$$

Equality: $\mathbf{u} = \mathbf{v}$ if and only if $u_1 = v_1,\ u_2 = v_2,\ \ldots,\ u_n = v_n$.

Addition:
$$\mathbf{u} + \mathbf{v} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} + \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} u_1 + v_1 \\ u_2 + v_2 \\ \vdots \\ u_n + v_n \end{pmatrix}$$

Scalar Multiplication:
$$c\mathbf{u} = c \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} c \cdot u_1 \\ c \cdot u_2 \\ \vdots \\ c \cdot u_n \end{pmatrix}$$

where c is a real number, called a **scalar**.

The set of all vectors in $\mathbf{R}^n$, taken together with theses definitions of addition and scalar multiplication, is called **Euclidean space**. The Euclidean space is an example of a vector space.

Theorem 12.1: Algebraic Properties of Vectors

Let a and b be scalars, and $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ be vectors in $\mathbf{R}^n$. Then

(a) $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$

(b) $a(\mathbf{u}+\mathbf{v}) = a\mathbf{v} + a\mathbf{u}$

(c) $(a+b)\,\mathbf{u} = a\mathbf{u} + b\mathbf{u}$

(d) $(\mathbf{u}+\mathbf{v})+\mathbf{w} = \mathbf{u}+(\mathbf{v}+\mathbf{w})$

(e) $a(b\mathbf{u}) = (ab)\,\mathbf{u}$

(f) $\mathbf{u}+(-\mathbf{u}) = \mathbf{0}$

(g) $\mathbf{u}+\mathbf{0} = \mathbf{0}+\mathbf{u} = \mathbf{u}$

(h) $1\,\mathbf{u} = \mathbf{u}$

(i) $-\mathbf{u} = (-1)\,\mathbf{u}$

where the zero vector given by $0 = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$.

⚠ Note that two vectors can be equal if they have the same number of components. Similarly, it is impossible to add two vectors that have a different number of components.

Suppose that we have the vectors in $\mathbf{R}^4$. Perform operations on following vectors to find $\mathbf{u}+\mathbf{v}$, $-4\mathbf{v}$, and $2\mathbf{u}-3\mathbf{v}$.

$$\mathbf{u} = \begin{pmatrix} 2 \\ -3 \\ 0 \\ -1 \end{pmatrix} \qquad \mathbf{v} = \begin{pmatrix} -4 \\ 6 \\ -2 \\ 7 \end{pmatrix}$$

◆ **Step 1.** Define vectors u and v.

In[ ]:=
```
ClearAll["Global`*"]
u = {{2}, {-3}, {0}, {-1}}; MatrixForm[u]
```

Out[ ]//MatrixForm=
$$\begin{pmatrix} 2 \\ -3 \\ 0 \\ -1 \end{pmatrix}$$

In[ ]:=
```
v = {{-4}, {6}, {-2}, {7}}; MatrixForm[v]
```

Out[ ]//MatrixForm=
$$\begin{pmatrix} -4 \\ 6 \\ -2 \\ 7 \end{pmatrix}$$

◆ **Step 2.** Check the number of component of the given vectors for equality.

In[ ]:=
```
Dimensions[u] == Dimensions[v]
```
Out[ ]=
```
True
```

- **Step 3.** Find the sum, **u + v**, by adding the corresponding components.

In[ ]:=
```
sol1 = {u[[1]] + v[[1]], u[[2]] + v[[2]], u[[3]] + v[[3]], u[[4]] + v[[4]]}; sol1 // MatrixForm
```
Out[ ]//MatrixForm=

$$\begin{pmatrix} -2 \\ 3 \\ -2 \\ 6 \end{pmatrix}$$

- **Step 4.** Find the scalar multiplication: **−4 v**. Use a **Do loop** to perform vector operations.
- Get the dimension of the vector.

In[ ]:=
```
dim = Dimensions[v]
```
Out[ ]=
```
{4, 1}
```

- Construct a zero vector.

In[ ]:=
```
sol2 = ConstantArray[0, dim]; MatrixForm[sol2]
```
Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

- Run a **do loop** to perform the scalar multiplication.

In[ ]:=
```
Do[sol2[[i, j]] = -4 * v[[i, j]], {i, 1, dim[[1]]}, {j, 1, dim[[2]]}];
MatrixForm[sol2]
```

$$\begin{pmatrix} 16 \\ -24 \\ 8 \\ -28 \end{pmatrix}$$

- **Step 5.** Find the expression: **2 u − 3 v**, which is an exampe of a *linear combination* of vectors. Use a **For loop** to perform vector operations.

Definition 12.3: Linear Combination

If $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m$ are vectors and $c_1, c_2, \ldots, c_m$ are scalars, then

$$c_1 \mathbf{u}_1 + c_2 \mathbf{u}_2 + \cdots + c_m \mathbf{u}_m$$

is a linear combination of $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m$.

⚠ Note that it is possible for scalars to be negative or equal to zero.

- Get the dimension of one of the vectors.

In[ ]:=
```
dim = Dimensions[u]
```
Out[ ]=
```
{4, 1}
```

- Construct a zero vector.

In[ ]:=
```
sol3 = ConstantArray[0, dim]; MatrixForm[sol3]
```
Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

- Run a **for loop** to perform the given vector operation.
- **Difference of two vectors** can be rewritten in the following way:
  2 u − 3 v = 2 u + (−3) v

In[ ]:=
```
sol3 = ConstantArray[0, dim];
For[i = 1, i ≤ dim[[1]], i++,
  For[j = 1, j ≤ dim[[2]], j++,
   sol3[[i, j]] = 2 * u[[i, j]] + (-3) * v[[i, j]]
  ]];
MatrixForm[sol3]
```

$$\begin{pmatrix} 16 \\ -24 \\ 6 \\ -23 \end{pmatrix}$$

- **Step 8.** Verify the results.

In[ ]:=
```
FullSimplify[sol1 == u + v]
```
Out[ ]=
```
True
```

In[ ]:=
```
FullSimplify[sol2 == -4 v]
```
Out[ ]=
```
True
```

In[ ]:=
```
FullSimplify[sol3 == 2 u - 3 v]
```
Out[ ]=
```
True
```

Example 12.2: Geometry of Vectors

Definition 11.4: Tip, Tail of Vector

Vectors have a geometric interpretation that is most easilty understood in $\mathbf{R}^2$. We plot the vector $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$ by drawing an arrow from the origin to the point $(x_1,\ x_2)$ in the plane.

The end of the vector with the arrow is called the **tip**, and the end at the origin is called the **tail**.

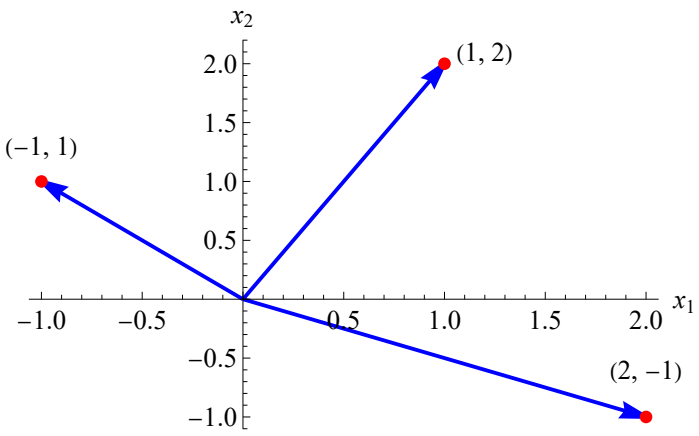
- The built-in command **Arrow** can be used to visualize the vectors.

In[ ]:=
```
u1 = {Arrowheads[Medium], Arrow[{{0, 0}, {-1, 1}}]};
u2 = {Arrowheads[Medium], Arrow[{{0, 0}, {1, 2}}]};
u3 = {Arrowheads[Medium], Arrow[{{0, 0}, {2, -1}}]};
```

- Then, **Graphics** and **Show** functions were used to display them in coordinate system.

In[ ]:=
```
Show[Graphics[{Thick, Blue, u1, u2, u3,
   Red, PointSize[0.02], Point[{-1, 1}], Point[{1, 2}], Point[{2, -1}], Black,
   Text[Style["(-1, 1)", FontFamily → "Times", FontSize → 14], {-1, 1.3}],
   Text[Style["(1, 2)", FontFamily → "Times", FontSize → 14], {1.2, 2.1}],
   Text[Style["(2, -1)", FontFamily → "Times", FontSize → 14], {2, -0.6}]}],
 Axes → True, AxesLabel → {"x₁", "x₂"}, LabelStyle →
  Directive[FontFamily → "Times", FontSize → 14, Black],
 ImageSize → 360, AspectRatio → 1 / GoldenRatio]
```

Out[ ]=



Suppose that we have the vectors in $\mathbf{R}^2$. Perform vector operations and illustrate them graphically.

Theorem 12.2.1: Geometric Procedures for Adding Vectors

1. **The Tip-to-Tail Rule:** Let **u** and **v** be two vectors. Tranlate the graph of **v**, preserving direction, so that its tail is at the tip of **u**. Then the tip of the translated **v** is at the tip of **u** + **v**.

2. The Parallelogram Rule: Let vectors **u** and **v** form two adjacent sides of a parallelogram with vertices at the origin, the tip of **u**, and the tip of **v**. Then the tip of **u** + **v** is at the fourth vertex.

Example 12.2.1. Vector addition

- **Step 1.** Define vectors u and v.

```
In[ ]:= ClearAll["Global`*"]
        u = {2, 5}; u // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 \\ 5 \end{pmatrix}$$

```
In[ ]:= v = {4, 2}; v // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 4 \\ 2 \end{pmatrix}$$

- **Step 2.** Check the number of component of the given vectors for equality.

```
In[ ]:= Dimensions[u] == Dimensions[v]
```

Out[ ]=

```
True
```

- **Step 3.** Find the sum, **u + v**.

```
In[ ]:= sum = u + v; sum // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 6 \\ 7 \end{pmatrix}$$

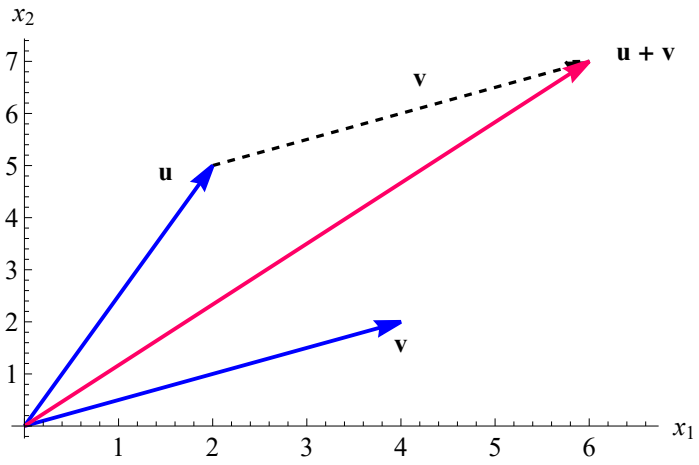
- **Step 4.** Represent the vectors **u**, **v**, and their sum as an arrow.

```
In[ ]:= u1 = {Arrowheads[Medium], Arrow[{{0, 0}, u}]};
        v1 = {Arrowheads[Medium], Arrow[{{0, 0}, v}]};
        sum1 = {Arrowheads[Medium], Arrow[{{0, 0}, sum}]};
```

- **Step 5.** Display the geometric interpretation of the **Tip-to-Tail Rule** for vector addition.

In[ ]:=
```
Show[Graphics[{Black, Dashed, Thickness[0.005], {Arrowheads[Medium], Arrow[{u, sum}]} ,
   Dashing[None], Thick, Blue, u1, v1, RGBColor[1, 0, 0.4], sum1, Black,
   Text[Style["u", Bold, FontFamily → "Times", FontSize → 14], {1.5, 4.9}],
   Text[Style["v", Bold, FontFamily → "Times", FontSize → 14], {4, 1.6}],
   Text[Style["u + v", Bold, FontFamily → "Times", FontSize → 14], {6.6, 7.2}],
   Text[Style["v", Bold, FontFamily → "Times", FontSize → 14], {4.2, 6.7}]}],
 Axes → True, AxesLabel → {"x1", "x2"},
 LabelStyle → Directive[FontFamily → "Times", FontSize → 14, Black],
 ImageSize → 360, AspectRatio → 1 / GoldenRatio]
```

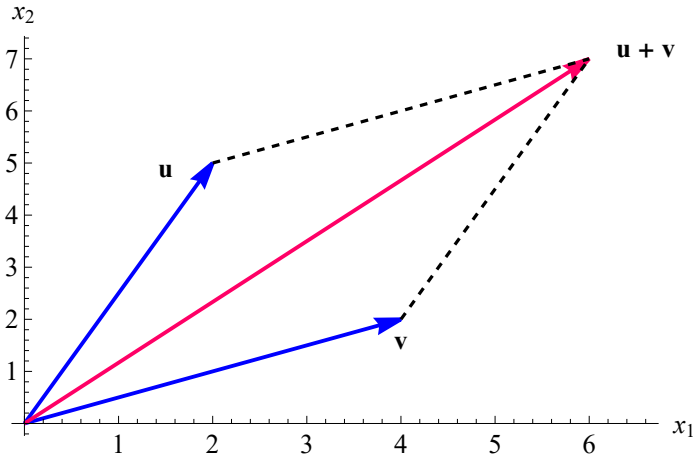
Out[ ]=



- The figure above shows vectors **u**, **v**, the translated **v** (dashed), and **u** + **v**. When we add **v** to **u**, we add each component of **v** to the corresponding component of **u**. Note that we get to the same place if we translate **u** instead of **v**.
- **Step 6.** Display the geometric interpretation of the **Parallelogram Rule** for vector addition.

In[ ]:=
```
Show[Graphics[{Thick, Blue, u1, v1, RGBColor[1, 0, 0.4], sum1,
   Black, Dashed, Thickness[0.005], Line[{u, sum}] , Line[{v, sum}],
   Text[Style["u", Bold, FontFamily → "Times", FontSize → 14], {1.5, 4.9}],
   Text[Style["v", Bold, FontFamily → "Times", FontSize → 14], {4, 1.6}],
   Text[Style["u + v", Bold, FontFamily → "Times", FontSize → 14], {6.6, 7.2}]}],
 Axes → True, AxesLabel → {"x₁", "x₂"},
 LabelStyle → Directive[FontFamily → "Times", FontSize → 14, Black],
 ImageSize → 360, AspectRatio → 1 / GoldenRatio]
```

Out[ ]=



- The figure above shows that the third and fourth sides of the parallelogram are translated copies of **u** and **v**, which shows the connection to the Tip-to-Tail Rule.

Theorem 12.2.2: Geometric Interpretation of the Scalar Multiplication of Vectors

If a vector **u** is multiplied by a scalar c, then the new vector $c\mathbf{u}$ points in the same direction as **u** when $c > 0$ and in the opposite direction when $c < 0$. The length of $c\mathbf{u}$ is equal to the length of **u** multiplied by $|c|$.

Example 12.2.2. Scalar Multiplication

- **Step 1.** Define vector u.

In[ ]:=
```
ClearAll["Global`*"]
u = {-1, 1}; u // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -1 \\ 1 \end{pmatrix}$$

- **Step 2.** Multiply the vector **u** by scalars.

```
In[ ]:= MatrixForm[u2 = -u]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

```
In[ ]:= MatrixForm[u3 = 2.5 * u]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -2.5 \\ 2.5 \end{pmatrix}$$

```
In[ ]:= MatrixForm[u4 = -2 u]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 \\ -2 \end{pmatrix}$$

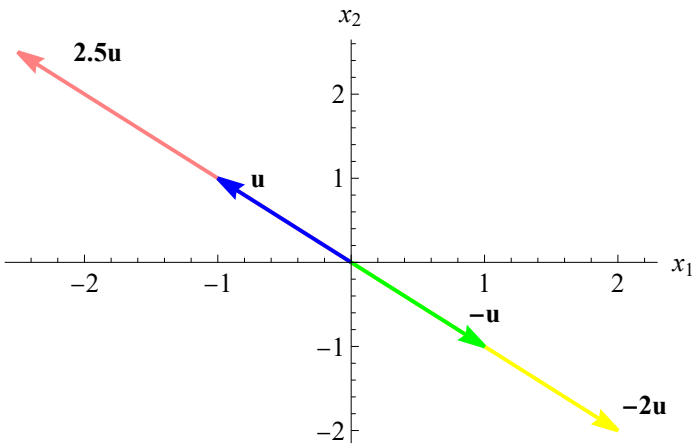
- **Step 3.** Represent all resulting vectors as arrows.

```
In[ ]:= u1 = {Arrowheads[Medium], Arrow[{{0, 0}, u}]};
        u2 = {Arrowheads[Medium], Arrow[{{0, 0}, u2}]};
        u3 = {Arrowheads[Medium], Arrow[{{0, 0}, u3}]};
        u4 = {Arrowheads[Medium], Arrow[{{0, 0}, u4}]};
```

- **Step 4.** Display the geometric interpretation of the **scalar multiples** of the vector **u**.

```
In[ ]:= Show[Graphics[{Thick, Pink, u3, Yellow, u4, Blue, u1, Green, u2, Black,
           Text[Style["u", Bold, FontFamily → "Times", FontSize → 14], {-0.7, 1}],
           Text[Style["-u", Bold, FontFamily → "Times", FontSize → 14], {1, -0.6}],
           Text[Style["2.5u", Bold, FontFamily → "Times", FontSize → 14], {-1.9, 2.5}],
           Text[Style["-2u", Bold, FontFamily → "Times", FontSize → 14], {2.2, -1.7}]}],
         Axes → True, AxesLabel → {"x1", "x2"}, LabelStyle →
          Directive[FontFamily → "Times", FontSize → 14, Black],
         ImageSize → 360, AspectRatio → 1 / GoldenRatio]
```

Out[ ]=



- 2.5**u** points in the same direction as **u** and is 2.5 times as long.
- **-u** points in the opposite direction of **u** and is equal in length.

- -2**u** points in the opposite direction of **u** and is twice as long.

Theorem 12.2.3: Geometric Interpretation of the Subtraction of Vectors

> Draw a vector **w** from the tip of **v** to the tip of **u**. Then translate **w**, preserving direction and placing the tail at the origin. The resulting vector is **u** - **v**.

Example 12.2.3. Vector Subtraction

- **Step 1.** Define vectors u and v.

```
In[ ]:= ClearAll["Global`*"]
        u = {2, 6}; u // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 \\ 6 \end{pmatrix}$$

```
In[ ]:= v = {5, 4}; v // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 5 \\ 4 \end{pmatrix}$$

- **Step 2.** Check the number of component of the given vectors for equality.

```
In[ ]:= Dimensions[u] == Dimensions[v]
```

Out[ ]=

```
True
```

- **Step 3.** Find the difference, $u - v$.

```
In[ ]:= diff = u - v; diff // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -3 \\ 2 \end{pmatrix}$$

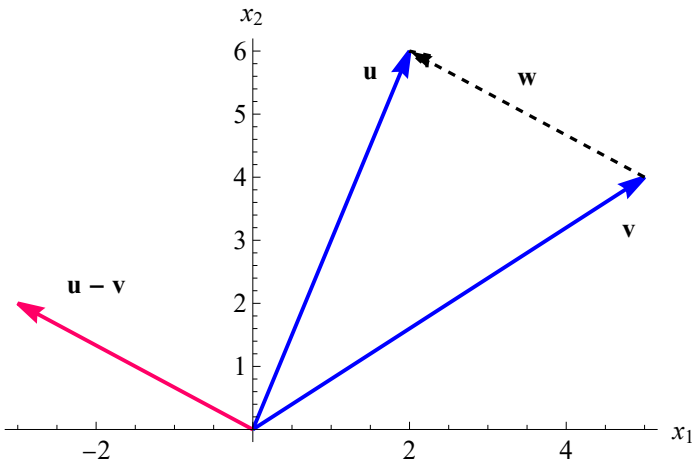
- **Step 4.** Represent the vectors **u**, **v**, and their difference as an arrow.

```
In[ ]:= u1 = {Arrowheads[Medium], Arrow[{{0, 0}, u}]};
        v1 = {Arrowheads[Medium], Arrow[{{0, 0}, v}]};
        diff1 = {Arrowheads[Medium], Arrow[{{0, 0}, diff}]};
```

- **Step 5.** Display the geometric interpretation of the **vector subtraction** procedure.

In[ ]:=
```
Show[Graphics[{Thick, Blue, u1, v1, RGBColor[1, 0, 0.4], diff1,
   Black, Dashed, Thickness[0.005], {Arrowheads[Medium], Arrow[{v, u}]},
   Text[Style["u", Bold, FontFamily → "Times", FontSize → 14], {1.5, 5.7}],
   Text[Style["v", Bold, FontFamily → "Times", FontSize → 14], {4.8, 3.2}],
   Text[Style["u - v", Bold, FontFamily → "Times", FontSize → 14], {-2, 2.3}],
   Text[Style["w", Bold, FontFamily → "Times", FontSize → 14], {3.5, 5.6}]}],
 Axes → True, AxesLabel → {"x1", "x2"},
 LabelStyle → Directive[FontFamily → "Times", FontSize → 14, Black],
 ImageSize → 360, AspectRatio → 1 / GoldenRatio]
```

Out[ ]=



Example 12.3: Span

Definition 12.5: Span

Let $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m\}$ be a set of vectors in $\mathbf{R}^n$. The **span** of this set is denoted span $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m\}$ and is defined as the set of all linear combinations

$$x_1 \mathbf{u}_1 + x_2 \mathbf{u}_2 + \cdots + x_m \mathbf{u}_m$$

where $x_1 + x_2 + \cdots + x_m$ can be any real numbers.

If span $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m\} = \mathbf{R}^n$, then we say that the set $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m\}$ spans $\mathbf{R}^n$.

Show that $\mathbf{v}_1$ is in S = span $\{\mathbf{u}_1, \mathbf{u}_2\}$, and that $\mathbf{v}_2$ is not.

$$\mathbf{u}_1 = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \quad \mathbf{u}_2 = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad \mathbf{v}_1 = \begin{pmatrix} -1 \\ 4 \\ 7 \end{pmatrix} \quad \mathbf{v}_2 = \begin{pmatrix} 8 \\ 2 \\ 1 \end{pmatrix}$$

◆ **Step 1.1.** Specify the given vectors.

```
In[ ]:= ClearAll["Global`*"]
        u1 = {{2}, {1}, {1}}; u1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}$$

```
In[ ]:= u2 = {{1}, {2}, {3}}; u2 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

```
In[ ]:= v1 = {{-1}, {4}, {7}}; v1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -1 \\ 4 \\ 7 \end{pmatrix}$$

Theorem 12.3: Vectors in R^n

Let $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m$ and $\mathbf{v}$ be vectors in $\mathbf{R}^n$. The $\mathbf{v}$ is an element of span $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m\}$ if and only if the linear system with vector equation and augmented matrix

$$x_1\,\mathbf{u}_1 + x_2\,\mathbf{u}_2 + \cdots + x_m\,\mathbf{u}_m = \mathbf{v}$$

$$\left[\begin{array}{cccc|c} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_m & \mathbf{v} \end{array}\right]$$

have a solution.

- **Step 2.1.** Construct the vector equation of the form: $x_1\,\mathbf{u}_1 + x_2\,\mathbf{u}_2 = \mathbf{v}_1$.

```
In[ ]:= x1 * MatrixForm[u1] + x2 * MatrixForm[u2] == MatrixForm[v1]
```

Out[ ]=

$$x2\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} + x1\begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} == \begin{pmatrix} -1 \\ 4 \\ 7 \end{pmatrix}$$

- **Step 3.1.** Construct the augmented matrix of the system, $\mathbf{A} = [\mathbf{u}_1 \quad \mathbf{u}_2 \quad | \quad \mathbf{v}_1]$.

```
In[ ]:= A = ArrayFlatten[{{u1, u2, v1}}]; MatrixForm[A]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 & 1 & -1 \\ 1 & 2 & 4 \\ 1 & 3 & 7 \end{pmatrix}$$

- **Step 4.1.** Transform the augmented matrix to echelon form.

In[ ]:=
```
RowReduce[A] // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & -2 \\ 0 & 1 & 3 \\ 0 & 0 & 0 \end{pmatrix}$$

- **Step 5.1.** Perform the back substitution to find scalars x_1 and x_2 and check the solution using LinearSolve function.

In[ ]:=
```
x2 = 3;
x1 = -2;
```

In[ ]:=
```
LinearSolve[{{2, 1}, {1, 2}, {1, 3}}, {-1, 4, 7}]
```

Out[ ]=

$$\{-2, 3\}$$

- Since the augmented matrix has a solution, $\mathbf{v}_1$ is in $S = \text{span}\{\mathbf{u}_1, \mathbf{u}_2\}$.
- **Step 6.1.** Define the linear combination.

In[ ]:=
```
Clear[u1, u2, v1]
v1 == x1 * u1 + x2 * u2
```

Out[ ]=
```
v1 == -2 u1 + 3 u2
```

- **Step 1.2.** Specify the given vectors.

In[ ]:=
```
ClearAll["Global`*"]
u1 = {{2}, {1}, {1}}; u1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}$$

In[ ]:=
```
u2 = {{1}, {2}, {3}}; u2 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

In[ ]:=
```
v2 = {{8}, {2}, {1}}; v2 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 8 \\ 2 \\ 1 \end{pmatrix}$$

- **Step 2.2.** Construct the vector equation of the form: $x_1\,\mathbf{u}_1 + x_2\,\mathbf{u}_2 = \mathbf{v}_2$.

```
In[ ]:= x1 * MatrixForm[u1] + x2 * MatrixForm[u2] == MatrixForm[v2]
```

Out[ ]=

$$x2\begin{pmatrix}1\\2\\3\end{pmatrix} + x1\begin{pmatrix}2\\1\\1\end{pmatrix} == \begin{pmatrix}8\\2\\1\end{pmatrix}$$

- **Step 3.2.** Construct the augmented matrix of the system, $\mathbf{A} = [\mathbf{u}_1 \quad \mathbf{u}_2 \mid \mathbf{v}_2]$.

```
In[ ]:= A = ArrayFlatten[{{u1, u2, v2}}]; MatrixForm[A]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix}2 & 1 & 8\\1 & 2 & 2\\1 & 3 & 1\end{pmatrix}$$

- **Step 4.2.** Transform the augmented matrix to echelon form.

```
In[ ]:= RowReduce[A] // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{pmatrix}$$

- The third row of the echelon matrix corresponds to the equation 0=1. Thus the system has no solutions and $\mathbf{v}_2$ is ***not*** **in** $\boldsymbol{S}$ = span $\{\mathbf{u}_1, \mathbf{u}_2\}$.
- **Step 5.2.** Check the solution using LinearSolve function.

```
In[ ]:= LinearSolve[{{2, 1}, {1, 2}, {1, 3}}, {8, 2, 1}]
```

```
··· LinearSolve: Linear equation encountered that has no solution.
```

Out[ ]=

```
LinearSolve[{{2, 1}, {1, 2}, {1, 3}}, {8, 2, 1}]
```

- Other theorems on span are listed below.

Theorem 12.4: Theorems on Span

Let $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m$ and $\mathbf{u}$ be vectors in $\mathbf{R}^n$. If $\mathbf{u}$ is in span $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m\}$ then

$$\text{span}\{\mathbf{u}, \mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m\} = \text{span}\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m\}$$

Suppose that $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m$ are in $\mathbf{R}^n$, and let

$$A = [\mathbf{u}_1 \quad \mathbf{u}_2 \quad \cdots \quad \mathbf{u}_m] \sim B$$

where B is in echelon form. Then span $\{\mathbf{u}_1, \ldots, \mathbf{u}_m\} = \mathbf{R}^n$ exactly when B has a pivot position in every row.

Let $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m\}$ be a set of vectors in $\mathbf{R}^n$. If $m < n$, then this set does not span $\mathbf{R}^n$. If $m \geq n$, then the set might span $\mathbf{R}^n$ or it might not. In this case, we cannot say more without additional information about the vectors.

Let $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m$ and $\mathbf{b}$ be vectors in $\mathbf{R}^n$. Then the following statements are equivalent. That is, if one is true, then so are the others, and if one i false then so are others.

(a) b is in span $\{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m\}$.

(b) The vector equation $x_1\,\mathbf{a}_1 + x_2\,\mathbf{a}_2 + \cdots + x_m\,\mathbf{a}_m = \mathbf{b}$ has at least one solution.

(c) The linear system corresponding to $[\ \mathbf{a}_1 \quad \mathbf{a}_2 \quad \cdots \quad \mathbf{a}_m \mid \mathbf{b}\ ]$ has at least one solution.

(d) The equation $A\boldsymbol{x} = \mathbf{b}$, with A and $\mathbf{x}$ given as

$$A = [\ \mathbf{a}_1 \quad \mathbf{a}_2 \quad \cdots \quad \mathbf{a}_m\ ] \qquad \text{and} \qquad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}$$

has at least one solution.

Example 12.4: Linear Independence

Definition 12.6: Linear Independence

Let $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m\}$ be a set of vectors in $\mathbf{R}^n$. If the only solution to the vector equation

$$x_1\,\mathbf{u}_1 + x_2\,\mathbf{u}_2 + \cdots + x_m\,\mathbf{u}_m = \mathbf{0}$$

is the trivial solution given by $x_1 = x_2 = \cdots = x_m = 0$, then the set $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m\}$ is **linearly independent**.

If there are nontrivial solutions, then the set is **linearly dependent**.

Determine if the given set is linearly dependent or linearly independent.

$$\mathbf{u}_1 = \begin{pmatrix} -1 \\ 4 \\ -2 \\ -3 \end{pmatrix} \qquad \mathbf{u}_2 = \begin{pmatrix} 3 \\ -13 \\ 7 \\ 7 \end{pmatrix} \qquad \mathbf{u}_3 = \begin{pmatrix} -2 \\ 1 \\ 9 \\ -5 \end{pmatrix}$$

◆ **Step 1.1.** Specify the given vectors.

```
In[ ]:= ClearAll["Global`*"]
        u1 = {{-1}, {4}, {-2}, {-3}}; u1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -1 \\ 4 \\ -2 \\ -3 \end{pmatrix}$$

```
In[ ]:= u2 = {{3}, {-13}, {7}, {7}}; u2 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 3 \\ -13 \\ 7 \\ 7 \end{pmatrix}$$

```
In[ ]:= u3 = {{-2}, {1}, {9}, {-5}}; u3 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -2 \\ 1 \\ 9 \\ -5 \end{pmatrix}$$

- **Step 2.1.** Construct the vector equation of the form: $x_1\,\mathbf{u}_1 + x_2\,\mathbf{u}_2 + x_3\,\mathbf{u}_3 = 0$.

```
In[ ]:= b = ConstantArray[0, {4, 1}];
        MatrixForm[x1 * u1] + MatrixForm[x2 * u2] + MatrixForm[x3 * u3] == MatrixForm[b]
```

Out[ ]=

$$\begin{pmatrix} -x1 \\ 4\,x1 \\ -2\,x1 \\ -3\,x1 \end{pmatrix} + \begin{pmatrix} 3\,x2 \\ -13\,x2 \\ 7\,x2 \\ 7\,x2 \end{pmatrix} + \begin{pmatrix} -2\,x3 \\ x3 \\ 9\,x3 \\ -5\,x3 \end{pmatrix} == \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

- **Step 3.1.** Construct the corresponding augmented matrix: $\mathbf{A} = [\mathbf{u}_1 \quad \mathbf{u}_2 \quad \mathbf{u}_3 \mid \mathbf{0}]$

```
In[ ]:= A = ArrayFlatten[{{u1, u2, u3, b}}]; MatrixForm[A]
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -1 & 3 & -2 & 0 \\ 4 & -13 & 1 & 0 \\ -2 & 7 & 9 & 0 \\ -3 & 7 & -5 & 0 \end{pmatrix}$$

- **Step 4.1.** Transform the augmented matrix to echelon form.

```
In[ ]:= RowReduce[A] // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

- **Step 5.1.** Perform the back substitution and check the solution using LinearSolve function.

```
x3 = 0;
x2 = 0;
x1 = 0;
```

In[ ]:=
```
LinearSolve[{{-1, 3, -2}, {4, -13, 1}, {-2, 7, 9}, {-3, 7, -5}}, b]
```

Out[ ]=

{{0}, {0}, {0}}

- The results show that the only solution is the trivial one, $x_1 = x_2 = x_3 = 0$. Hence the $\{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3\}$ is **linearly independent.**

- Other theorems on linear independence are listed below.

Theorem 12.5: Theorems on Linear Independence

Suppose that $\{\mathbf{0}, \mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m\}$ is a set of vectors in $\mathbf{R}^n$. Then the set is linearly dependent.

Suppose that $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m\}$ is a set of vectors in $\mathbf{R}^n$. If $n < m$, then the set is linearly dependent.

Let $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m\}$ be a set of vectors in $\mathbf{R}^n$. Then this set is linearly dependent if and only if one of the vectors in the set is in the span of the other vectors.

Let $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m$ be in $\mathbf{R}^n$, and suppose

$$A = [\ \mathbf{u}_1 \quad \mathbf{u}_2 \quad \cdots \quad \mathbf{u}_m\ ] \sim B$$

where B is in echelon form. Then

(a) span $\{\mathbf{u}_1, \ldots, \mathbf{u}_m\} = \mathbf{R}^n$ exactly when B has a pivot position in every row.

(b) $\{\mathbf{u}_1, \ldots, \mathbf{u}_m\}$ is linearly independent exactly when B has a pivot position in every column.

Let $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m$ and $\mathbf{b}$ be vectors in $\mathbf{R}^n$. Then the following statements are equivalent. That is, if one is true, then so are the others, and if one i false then so are others.

(a) The set $\{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m\}$ is linearly independent.

(b) The vector equation $x_1\,\mathbf{a}_1 + x_2\,\mathbf{a}_2 + \cdots + x_m\,\mathbf{a}_m = \mathbf{b}$ has at most one solution for every $\mathbf{b}$.

(c) The linear system corresponding to $[\ \mathbf{a}_1 \quad \mathbf{a}_2 \quad \cdots \quad \mathbf{a}_m \mid \mathbf{b}\ ]$ has at most one solution for every $\mathbf{b}$.

(d) The equation $A\boldsymbol{x} = \mathbf{b}$, with $A = [\ \mathbf{a}_1 \quad \mathbf{a}_2 \quad \cdots \quad \mathbf{a}_m\ ]$, has at most one solution for every $\mathbf{b}$.

Example 12.5: Dot Products and its Applications

Definition 12.7: Dot Product

Suppose that

$$\mathbf{u} = \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} \quad \text{and} \quad \mathbf{v} = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}$$

are both in $\mathbf{R}^n$. Then the **dot product** of $\mathbf{u}$ and $\mathbf{v}$ is given by

$$\mathbf{u} \cdot \mathbf{v} = u_1 v_1 + \cdots + u_n v_n$$

An alternative way to define the dot product of $\mathbf{u}$ and $\mathbf{v}$ is with matrix multiplication

$$\mathbf{u} \cdot \mathbf{v} = \mathbf{u}^T \mathbf{v} = \begin{pmatrix} u_1 & \cdots & u_n \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} = u_1 v_1 + \cdots + u_n v_n$$

⚠ Unlike vector addition, which produces a new vector, the dot product of two vectors yields a scalar.

Theorem 12.6: Properties of the Dot Product

Let $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ be vectors in $\mathbf{R}^n$, and c be a scalar. Then

(a) $\mathbf{u} \cdot \mathbf{v} = \mathbf{v} \cdot \mathbf{u}$

(b) $(\mathbf{u} + \mathbf{v}) \cdot \mathbf{w} = \mathbf{u} \cdot \mathbf{w} + \mathbf{v} \cdot \mathbf{w}$

(c) $(c\mathbf{u}) \cdot \mathbf{v} = \mathbf{u} \cdot (c\mathbf{u}) = c\,(\mathbf{u} \cdot \mathbf{v})$

(d) $\mathbf{u} \cdot \mathbf{u} \geq 0$, and $\mathbf{u} \cdot \mathbf{u} = 0$ only when $\mathbf{u} = 0$

Prove the **property (b)** of **Theorem 12.6** using the given vectors.

$$\mathbf{u} = \begin{pmatrix} 2 \\ 1 \\ -3 \\ 2 \end{pmatrix} \quad \mathbf{v} = \begin{pmatrix} -1 \\ 4 \\ 0 \\ 3 \end{pmatrix} \quad \mathbf{w} = \begin{pmatrix} 5 \\ 0 \\ 1 \\ 2 \end{pmatrix}$$

◆ **Step 1.** Define the given vectors.

In[ ]:=
```
ClearAll["Global`*"]
u = {2, 1, -3, 2}; u // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 \\ 1 \\ -3 \\ 2 \end{pmatrix}$$

In[ ]:=
```
v = {-1, 4, 0, 3}; v // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -1 \\ 4 \\ 0 \\ 3 \end{pmatrix}$$

```
In[ ]:= w = {5, 0, 1, 2}; w // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 5 \\ 0 \\ 1 \\ 2 \end{pmatrix}$$

- Lets start with the right-hand-side of (b).
- **Step 2.** Compute the sum **u** + **v**.

```
In[ ]:= sumUV = u + v; sumUV // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 \\ 5 \\ -3 \\ 5 \end{pmatrix}$$

- **Step 3.** Compute the dot product of **u** + **v** and **w** by multiplying the corresponding entries and adding them together.
- So the right-hand-side of **Theorem 12.6 (b)** equals to:

```
In[ ]:= RHS = sumUV[[1]] * w[[1]] + sumUV[[2]] * w[[2]] + sumUV[[3]] * w[[3]] + sumUV[[4]] * w[[4]]
```

Out[ ]=

```
12
```

- Compute the left-hand-side expression of (b) using a **Do** and **For loops.**
- **Step 4.** Get the dimension of one of the vectors.

```
In[ ]:= dim = Dimensions[u]
```

Out[ ]=

```
{4}
```

- **Step 5.** Assign the two new variables responsible for the dot products **u** · **w** and **v** · **w** a zero value.

```
In[ ]:= dotUW = 0;
        dotVW = 0;
```

- **Step 6.** Run a **do loop** for computing the dot product of the vectors **u** and **w**.

```
In[ ]:= Do[dotUW = dotUW + u[[i]] * w[[i]], {i, 1, dim[[1]]}];
        dotUW
```

Out[ ]=

```
11
```

- **Step 7.** Run a **for loop** for computing the dot product of the vectors **v** and **w**.

```
In[ ]:= For[i = 1, i ≤ dim[[1]], i++,
          dotVW = dotVW + v[[i]] * w[[i]]
        ];
        dotVW
Out[ ]=
        1
```

◆ **Step 8.** Sum upp the two dot products.

```
In[ ]:= LHS = dotUW + dotVW
Out[ ]=
        12
```

◆ **Step 9.** Check the two sides of (b) for equality.

```
In[ ]:= RHS == LHS
Out[ ]=
```

True

◆ **Step 10.** Confirm the results using the built-in function **Dot(.)**.

```
In[ ]:= RHS == Dot[(u + v), w]
Out[ ]=
        True

In[ ]:= LHS == u.w + v.w
Out[ ]=
        True
```

The **dot product** is used to find the **lengths**, **distances**, **angles**, and **orthogonality** of vectors.

Definition 12.8: Norm of a Vector

Let $\mathbf{x}$ be a vector in $\mathbf{R}^n$. Then the **norm** (or **length**) of $\mathbf{x}$ is given by

$$\| \mathbf{x} \| = \sqrt{\mathbf{x} \cdot \mathbf{x}} = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$

For a scalar c and a vector $\mathbf{x}$, it follows from Theorem 12.6c that

$$\| c\mathbf{x} \| = |c| \, \| \mathbf{x} \|$$

Find $\| \mathbf{x} \|$ and $\| -5\mathbf{x} \|$ for the given vector $\mathbf{x}$.

$$\mathbf{x} = \begin{pmatrix} -3 \\ 1 \\ 4 \end{pmatrix}$$

◆ **Step 1.** Define the given vector **x**.

```
In[ ]:= ClearAll["Global`*"]
        x = {-3, 1, 4}; x // MatrixForm
Out[ ]//MatrixForm=
        ⎛-3⎞
        ⎜ 1⎟
        ⎝ 4⎠
```

◆ **Step 2.** Compute the length of **x** using the definition.

```
In[ ]:= norm1 = √(x[[1]]^2 + x[[2]]^2 + x[[3]]^2)
Out[ ]=
        √26
```

◆ **Step 3.** Compute the length of -5**x** using the definition.

```
In[ ]:= norm2 = Abs[-5] * norm1
Out[ ]=
        5 √26
```

◆ **Step 4.** Confirm the results using the built-in functions.

```
In[ ]:= √x.x == norm1
Out[ ]=
        True

In[ ]:= Norm[-5 x] == norm2
Out[ ]=
        True
```

Definition 12.9: Distance Between Vectors

For two vectors **u** and **v** in $\mathbf{R}^n$, the **distance** between **u** and **v** is given by $\|\mathbf{u}-\mathbf{v}\|$.

Compute the distance between the given vectors **u** and **v**.

$$\mathbf{u}=\begin{pmatrix}-1\\3\\2\end{pmatrix} \qquad \mathbf{v}=\begin{pmatrix}7\\1\\-5\end{pmatrix}$$

◆ **Step 1.** Define the given vectors **u** and **v**.

```
In[ ]:= ClearAll["Global`*"]
        u = {-1, 3, 2}; u // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -1 \\ 3 \\ 2 \end{pmatrix}$$

```
In[ ]:= v = {7, 1, -5}; v // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 7 \\ 1 \\ -5 \end{pmatrix}$$

- **Step 2.** Compute the difference of two vectors: **u** - **v**.

```
In[ ]:= diff = u - v; diff // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -8 \\ 2 \\ 7 \end{pmatrix}$$

- **Step 3.** Compute the distance between two vectors using the definition.

```
In[ ]:= d = Sqrt[diff[[1]]^2 + diff[[2]]^2 + diff[[3]]^2]
```

Out[ ]=

$$3\sqrt{13}$$

- **Step 4.** Confirm the results using the built-in functions **Norm** and **EuclideanDistance**.

```
In[ ]:= Norm[u - v] == d
```

Out[ ]=

```
True
```

```
In[ ]:= EuclideanDistance[u, v] == d
```

Out[ ]=

```
True
```

Definition 12.10: Angle Between Vectors

Let **u** and **v** be vectors in $\mathbf{R}^n$. Then the **angle** θ between **u** and **v** is given by

$$\mathbf{u} \cdot \mathbf{v} = \| \mathbf{u} \| \, \| \mathbf{v} \| \cos(\theta)$$

Compute the angle between the given vectors **u** and **v**.

$$\mathbf{u} = \begin{pmatrix} 2 \\ 3 \end{pmatrix} \qquad \mathbf{v} = \begin{pmatrix} -3 \\ 4 \end{pmatrix}$$

◆ **Step 1.** Define the given vectors **u** and **v**.

```
In[ ]:= ClearAll["Global`*"]
        u = {2, 3}; u // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

```
In[ ]:= v = {-3, 4}; v // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -3 \\ 4 \end{pmatrix}$$

◆ **Step 2.** Solve the equation in definition for cos(θ).

```
In[ ]:= cos = u.v / (Norm[u] * Norm[v])
```

Out[ ]=

$$\frac{6}{5\sqrt{13}}$$

◆ **Step 3.** Find the angle θ.

```
In[ ]:= angle = ArcCos[cos]
```

Out[ ]=

$$\text{ArcCos}\left[\frac{6}{5\sqrt{13}}\right]$$

◆ **Step 4.** Convert the angle into degrees.

```
In[ ]:= angleDeg = N[angle] * 180 / Pi
```

Out[ ]=

```
70.56
```

◆ **Step 5.** Confirm the results using the built-in function **VectorAngle**.

```
In[ ]:= VectorAngle[u, v]
```

Out[ ]=

$$\text{ArcCos}\left[\frac{6}{5\sqrt{13}}\right]$$

Definition 12.11: Orthogonal Vectors

> Vectors **u** and **v** in $\mathbf{R}^n$ are **orthogonal** if $\mathbf{u} \cdot \mathbf{v} = 0$.

Determine if any pair among **u**, **v**, and **w** is orthogonal.

$$\mathbf{u} = \begin{pmatrix} 2 \\ -1 \\ 5 \\ -2 \end{pmatrix} \qquad \mathbf{v} = \begin{pmatrix} 3 \\ 2 \\ -4 \\ 0 \end{pmatrix} \qquad \mathbf{w} = \begin{pmatrix} 2 \\ 9 \\ 6 \\ 4 \end{pmatrix}$$

◆ **Step 1.** Define the given vectors **u**, **v**, and **w**.

```
In[ ]:= ClearAll["Global`*"]
        u = {2, -1, 5, -2}; u // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 \\ -1 \\ 5 \\ -2 \end{pmatrix}$$

```
In[ ]:= v = {3, 2, -4, 0}; v // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 3 \\ 2 \\ -4 \\ 0 \end{pmatrix}$$

```
In[ ]:= w = {2, 9, 6, 4}; w // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 \\ 9 \\ 6 \\ 4 \end{pmatrix}$$

◆ **Step 2.** Compute their dot products.

```
In[ ]:= u.v
```

Out[ ]=

```
-16
```

◆ The dot product is not equal to zero, hence **u** and **v** are **not orthogonal**.

```
In[ ]:= u.w
```

Out[ ]=

```
17
```

◆ The dot product is not equal to zero, hence **u** and **w** are **not orthogonal**.

```
In[ ]:= v.w
```

Out[ ]=

```
0
```

◆ The dot product is equal to zero, hence **v** and **w** are **orthogonal**.

◆ **Step 3.** Confirm the results by computing the angles between vectors.

- The term ***orthogonal*** is commonly said to be equivalent to **perpendicular**.

```
In[ ]:= N[VectorAngle[u, v]] * 180/Pi
Out[ ]= 120.633

In[ ]:= N[VectorAngle[u, w]] * 180/Pi
Out[ ]= 75.5766

In[ ]:= N[VectorAngle[v, w]] * 180/Pi
Out[ ]= 90.
```

- The angle between **v** and **w** vectors is 90^o, hence they are **perpendicular~orthogonal**.

Example 12.6: Linear Transformations

Definition 12.12: Linear Transformation

A function T: $\mathbf{R}^m \to \mathbf{R}^n$ is a **linear transformation** if for all vectors **u** and **v** in $\mathbf{R}^m$ and all scalars r and s we have

(a) $T(\mathbf{u}+\mathbf{v}) = T(\mathbf{u}) + T(\mathbf{v})$

(b) $T(r\mathbf{u}) = rT(\mathbf{u})$

Conditions (a) and (b) can be combined into a single condition

$$T(r\mathbf{u}+s\mathbf{v}) = rT(\mathbf{u}) + sT(\mathbf{v})$$

Suppose that T: $\mathbf{R}^3 \to \mathbf{R}^4$ is defined by

$$T\left(\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}\right) = \begin{pmatrix} 2x_1 + x_3 \\ -x_1 + 2x_2 \\ x_1 - 3x_2 + 5x_3 \\ 4x_2 \end{pmatrix}$$

(a) Show that T is a linear transformation.

(b) Is a linear transformation one-to-one?

(c) Is a linear transformation onto?

(a) Approach 1: To show that T is a linear transformation requires the verifying the both conditions (a) and (b) of **Definition 12.12**.

- **Step 1.** Define the vectors u and v, and the given linear transformation.

```
In[ ]:= ClearAll["Global`*"]
        u = {u1, u2, u3}; u // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} u1 \\ u2 \\ u3 \end{pmatrix}$$

```
In[ ]:= v = {v1, v2, v3}; v // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} v1 \\ v2 \\ v3 \end{pmatrix}$$

```
In[ ]:= T[{x1_, x2_, x3_}] := {{2 x1 + x3}, {-x1 + 2 x2}, {x1 - 3 x2 + 5 x3}, {4 x2}}
```

- **Step 2.** For Part (a), verify that $T(\boldsymbol{u}+\boldsymbol{v}) = T(\boldsymbol{u}) + T(\boldsymbol{v})$.
- Left-hand-side: $T(\boldsymbol{u}+\boldsymbol{v})$.

```
In[ ]:= aLHS = T[u + v]; aLHS // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} u3 + 2\,(u1 + v1) + v3 \\ -u1 - v1 + 2\,(u2 + v2) \\ u1 + v1 - 3\,(u2 + v2) + 5\,(u3 + v3) \\ 4\,(u2 + v2) \end{pmatrix}$$

- Expanded form:

```
In[ ]:= Expand[%] // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2\,u1 + u3 + 2\,v1 + v3 \\ -u1 + 2\,u2 - v1 + 2\,v2 \\ u1 - 3\,u2 + 5\,u3 + v1 - 3\,v2 + 5\,v3 \\ 4\,u2 + 4\,v2 \end{pmatrix}$$

- Right-hand-side: $T(\boldsymbol{u}) + T(\boldsymbol{v})$.

```
In[ ]:= aRHS = T[u] + T[v]; aRHS // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2\,u1 + u3 + 2\,v1 + v3 \\ -u1 + 2\,u2 - v1 + 2\,v2 \\ u1 - 3\,u2 + 5\,u3 + v1 - 3\,v2 + 5\,v3 \\ 4\,u2 + 4\,v2 \end{pmatrix}$$

- Checking both sides of (a) for equality:

In[ ]:=
```
FullSimplify[aLHS == aRHS]
```
Out[ ]=
```
True
```

- **Step 3.** For Part (b), verify that $T(r\boldsymbol{u}) = rT(\boldsymbol{u})$.
- Left-hand-side: $T(r\boldsymbol{u})$.

In[ ]:=
```
bLHS = T[r * u]; bLHS // MatrixForm
```
Out[ ]//MatrixForm=
$$\begin{pmatrix} 2\,r\,u1 + r\,u3 \\ -r\,u1 + 2\,r\,u2 \\ r\,u1 - 3\,r\,u2 + 5\,r\,u3 \\ 4\,r\,u2 \end{pmatrix}$$

- Right-hand-side: $rT(\boldsymbol{u})$.

In[ ]:=
```
bRHS = r * T[u]; bRHS // MatrixForm
```
Out[ ]//MatrixForm=
$$\begin{pmatrix} r\,(2\,u1 + u3) \\ r\,(-u1 + 2\,u2) \\ r\,(u1 - 3\,u2 + 5\,u3) \\ 4\,r\,u2 \end{pmatrix}$$

- Expanded form:

In[ ]:=
```
Expand[%] // MatrixForm
```
Out[ ]//MatrixForm=
$$\begin{pmatrix} 2\,r\,u1 + r\,u3 \\ -r\,u1 + 2\,r\,u2 \\ r\,u1 - 3\,r\,u2 + 5\,r\,u3 \\ 4\,r\,u2 \end{pmatrix}$$

- Checking both sides of (a) for equality:

In[ ]:=
```
FullSimplify[bLHS == bRHS]
```
Out[ ]=
```
True
```

The results verify that both parts of the **Definition 12.12** hold, so T is a **linear transformation.**

Theorem 12.7: Matrix Transformation

> Let T: $\mathbf{R}^m \to \mathbf{R}^n$. Then T is a linear transformation if and only if $T(\mathbf{x}) = A\mathbf{x}$ for some $n \times m$ matrix A.

(a) Approach 2: To apply the **Theorem 12.7** and find matrix A such that $T(\mathbf{x}) = A\mathbf{x}$ to show

that T is a linear transformation.

◆ **Step 1.** Define the vector **x** and the given linear transformation.

```
In[ ]:= ClearAll["Global`*"]
        x = {x1, x2, x3}; x // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} x1 \\ x2 \\ x3 \end{pmatrix}$$

```
In[ ]:= T[{x1_, x2_, x3_}] := {{2 x1 + x3}, {-x1 + 2 x2}, {x1 - 3 x2 + 5 x3}, {4 x2}}
```

◆ **Step 2.** Compute the linear transformation of **x**.

```
In[ ]:= T[x] // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2\,x1 + x3 \\ -x1 + 2\,x2 \\ x1 - 3\,x2 + 5\,x3 \\ 4\,x2 \end{pmatrix}$$

◆ **Step 3.** Rewrite the given linear transformation function by adding the missing elements.

```
In[ ]:= newT[{x1_, x2_, x3_}] :=
         {{2 x1 + 0 x2 + x3}, {-x1 + 2 x2 + 0 x3}, {x1 - 3 x2 + 5 x3}, {0 x1 + 4 x2 + 0 x3}}
```

```
In[ ]:= newT[{x1, x2, x3}] // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2\,x1 + x3 \\ -x1 + 2\,x2 \\ x1 - 3\,x2 + 5\,x3 \\ 4\,x2 \end{pmatrix}$$

◆ **Step 4.** Verify that nothing have changed and the linear transformations are the same.

```
In[ ]:= T[{x1, x2, x3}] == newT[{x1, x2, x3}]
```

Out[ ]=

```
True
```

◆ **Step 5.** Find the matrix A so that $T(\mathbf{x}) = A\mathbf{x}$.

```
In[ ]:= newT[{x1_, x2_, x3_}] :=
         {{2 x1, +0 x2, +x3}, {-x1, +2 x2, +0 x3}, {x1, -3 x2, +5 x3}, {0 x1, +4 x2, +0 x3}}

        A = newT[{1, 1, 1}]; A // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 & 0 & 1 \\ -1 & 2 & 0 \\ 1 & -3 & 5 \\ 0 & 4 & 0 \end{pmatrix}$$

◆ **Step 6.** Verify the equality $T(\mathbf{x}) = A\mathbf{x}$.

```
In[ ]:= T[x] == A. {{x1}, {x2}, {x3}} // FullSimplify
Out[ ]= True
```

◆ Since, $T(\mathbf{x}) = A\mathbf{x}$, T is a **linear transformation** by **Theorem 12.7** .

Definition 12.13: One-to-One and Onto Linear Transformations

Let T: $\mathbf{R}^m \to \mathbf{R}^n$ be a linear transformation. Then if for all vectors $\mathbf{u}$ and $\mathbf{v}$ in $\mathbf{R}^m$ and all scalars r and s we have

(a) T is **one-to-one** if for every vector $\mathbf{w}$ in $\mathbf{R}^n$ there exists *at most* one vector $\mathbf{u}$ in $\mathbf{R}^m$ such that $T(\mathbf{u}) = \mathbf{w}$.

(b) T is **onto** if for every vector $\mathbf{w}$ in $\mathbf{R}^n$ there exists *at least* one vector $\mathbf{u}$ in $\mathbf{R}^m$ such that $T(\mathbf{u}) = \mathbf{w}$.

Theorem 12.8: Conditions for One-to-One Linear Transformation

Let T be a linear transformation. Then T is one-to-one if and only if the solution to $T(\mathbf{x}) = 0$ is the trivial solution $\mathbf{x} = 0$.

Let A be a $n \times m$ matrix and define T: $\mathbf{R}^m \to \mathbf{R}^n$ by $T(\mathbf{x}) = A\mathbf{x}$. Then

(a) T is one-to-one if and only if the columns of A are linearly independent.

(b) If $A \sim B$ and B is in echelon form, then T is one-to-one if and only if B has a pivot position in every column.

(c) If $n < m$, then T is *not* one-to-one.

(b) To find whether the linear transformation is one-to-one or not, we shoud apply either part of **Theorem 12.8**.

According to the **Theorem 12.8.1**, we need to find the solution to $T(\mathbf{x}) = 0$, which is equivalent to solving $A\mathbf{x} = 0$.

◆ **Step 1.** Define the given linear transformation and corresponding matrix A.

```
In[ ]:= T[{x1_, x2_, x3_}] :=
         {{2 x1, +0 x2, +x3}, {-x1, +2 x2, +0 x3}, {x1, -3 x2, +5 x3}, {0 x1, +4 x2, +0 x3}}

        A = T[{1, 1, 1}]; A // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 & 0 & 1 \\ -1 & 2 & 0 \\ 1 & -3 & 5 \\ 0 & 4 & 0 \end{pmatrix}$$

- **Step 2.** Construct the corresponding augmented matrix for $A\mathbf{x} = 0$.

```
In[ ]:= b = ConstantArray[0, {4, 1}];
        AugMat = ArrayFlatten[{{A, b}}]; AugMat // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 & 0 & 1 & 0 \\ -1 & 2 & 0 & 0 \\ 1 & -3 & 5 & 0 \\ 0 & 4 & 0 & 0 \end{pmatrix}$$

- **Step 3.** Reduce the augmented matrix to row echelon form.

```
In[ ]:= RowReduce[AugMat] // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

- The echelon form shows that $A\mathbf{x} = 0$ has only the trivial solution.
- **Step 4.** Verify the result using built-in function.

```
In[ ]:= LinearSolve[A, b]
```

Out[ ]=

```
{{0}, {0}, {0}}
```

$T(\mathbf{x}) = 0$ has only the trivial solution and thus T is **one-to-one**.

Theorem 12.9: Conditions for Onto Linear Transformation

Let A be a $n \times m$ matrix and define T: $\mathbf{R}^m \to \mathbf{R}^n$ by $T(\mathbf{x}) = A\mathbf{x}$. Then

(a) T is onto if and only if the columns of A span the codomain R^n.

(b) If $A \sim B$ and B is in echelon form, then T is onto if and only if B has a pivot position in every row.

(c) If $n > m$, then T is *not* onto.

(c) To find whether the linear transformation is onto or not, we shoud apply **Theorem 12.9**.

◆ **Step 1.** Define the given linear transformation and corresponding matrix A.

```
In[ ]:= T[{x1_, x2_, x3_}] :=
         {{2 x1, +0 x2, +x3}, {-x1, +2 x2, +0 x3}, {x1, -3 x2, +5 x3}, {0 x1, +4 x2, +0 x3}}

        A = T[{1, 1, 1}]; A // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 2 & 0 & 1 \\ -1 & 2 & 0 \\ 1 & -3 & 5 \\ 0 & 4 & 0 \end{pmatrix}$$

◆ **Step 2.** Get the dimensions of the matrix A.

```
In[ ]:= dim = Dimensions[A]
```

Out[ ]=

```
{4, 3}
```

```
In[ ]:= n = dim[[1]];
        m = dim[[2]];
```

◆ **Step 3.** Check the condition (c) of the given theorem and compare the number of rows and columns.

```
In[ ]:= n > m
```

Out[ ]=

```
True
```

Since $n = 4 > m = 3$, t is **not onto**.

Example 12.7: Geometry of Linear Transformations

Every linear transformation of the plane has a geometric effects and used to change the position. orientation, or shape of geometric objects. This section discusses the different types of common geometric linear transformations like **reflections**, **rotations**, **shears**, **dilations**, and **projections**.

◆ Define the vector **u**.

```
In[ ]:= ClearAll["Global`*"]
        u = {5, 4}; u // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 5 \\ 4 \end{pmatrix}$$

Refection Across x-Axis

◆ Define the given geometric transformation with the matrix A.

In[ ]:=
```
A = {{1, 0}, {0, -1}}; A // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

◆ Define the corresponding linear transformation using the definition $T(\mathbf{x}) = A\mathbf{x}$.

In[ ]:=
```
T[{x_, y_}] = A.u
```

Out[ ]=

```
{5, -4}
```

◆ Find the reflection of the vector **u** across the x-axis.

In[ ]:=
```
Rx = T[u]
```

Out[ ]=

```
{5, -4}
```

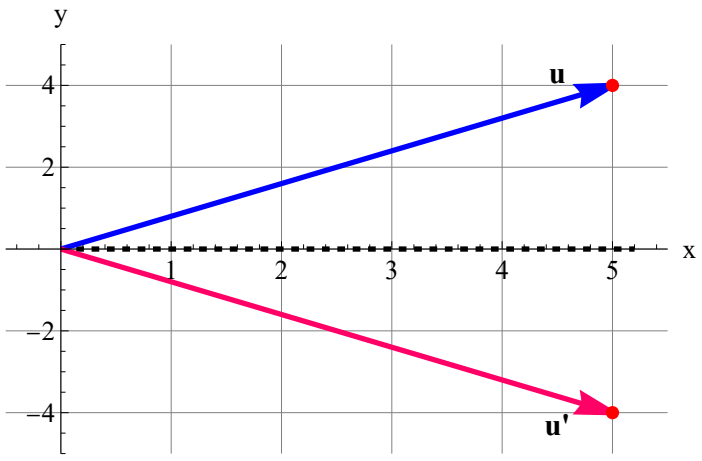
◆ Represent the vector **u** and its image as an arrow.

In[ ]:=
```
u1 = {Arrowheads[0.05], Arrow[{{0, 0}, u}]};
Rx1 = {Arrowheads[0.05], Arrow[{{0, 0}, Rx}]};
```

◆ Show the image of vector **u** under the reflection across x-axis.

In[ ]:=
```
Show[Graphics[{Thickness[0.008], Black, Dashed, Line[{{0, 0}, {5.2, 0}}], Dashing[None],
    Blue, u1, RGBColor[1, 0, 0.4], Rx1, Red, PointSize[0.02], Point[u], Point[Rx], Black,
    Text[Style["u", Bold, FontFamily → "Times", FontSize → 16], {4.5, 4.3}],
    Text[Style["u'", Bold, FontFamily → "Times", FontSize → 16], {4.5, -4.3}]},
  GridLines → Automatic], Axes → True, AxesLabel → {"x", "y"},
 PlotRange → {{-0.5, 5.5}, {-5, 5}},
 LabelStyle → Directive[FontFamily → "Times", FontSize → 14, Black],
 ImageSize → 360, AspectRatio → 1 / GoldenRatio]
```

Out[ ]=



Refection Across y-Axis

◆ Define the given geometric transformation with the matrix A.

In[ ]:=
```
A = {{-1, 0}, {0, 1}}; A // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

◆ Define the corresponding linear transformation using the definition $T(\mathbf{x}) = A\mathbf{x}$.

In[ ]:=
```
T[{x_, y_}] = A.u
```

Out[ ]=

```
{-5, 4}
```

◆ Find the reflection of the vector **u** across the y-axis.

In[ ]:=
```
Ry = T[u]
```

Out[ ]=

```
{-5, 4}
```

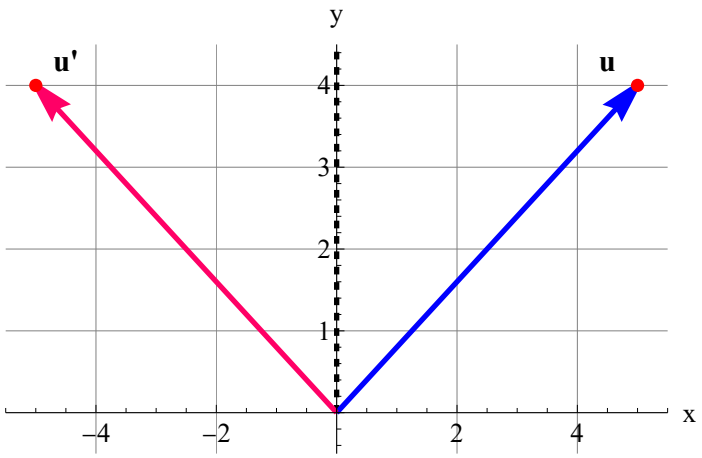
◆ Represent the vector **u** and its image as an arrow.

In[ ]:=
```
u1 = {Arrowheads[0.05], Arrow[{{0, 0}, u}]};
Ry1 = {Arrowheads[0.05], Arrow[{{0, 0}, Ry}]};
```

◆ Show the image of vector **u** under the reflection across y-axis.

In[ ]:=
```
Show[Graphics[{Thickness[0.008], Black, Dashed, Line[{{0, 0}, {0, 4.5}}], Dashing[None],
    Blue, u1, RGBColor[1, 0, 0.4], Ry1, Red, PointSize[0.02], Point[u], Point[Ry], Black,
    Text[Style["u", Bold, FontFamily → "Times", FontSize → 16], {4.5, 4.3}],
    Text[Style["u'", Bold, FontFamily → "Times", FontSize → 16], {-4.5, 4.3}]},
   GridLines → Automatic], Axes → True, AxesLabel → {"x", "y"},
  PlotRange → {{-5.5, 5.5}, {-0.5, 4.5}},
  LabelStyle → Directive[FontFamily → "Times", FontSize → 14, Black],
  ImageSize → 360, AspectRatio → 1 / GoldenRatio]
```

Out[ ]=



Rotation by Angle θ

◆ Define the given geometric transformation with the matrix A.

```
In[ ]:= A = {{Cos[θ], -Sin[θ]}, {Sin[θ], Cos[θ]}}; A // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} \text{Cos}[\theta] & -\text{Sin}[\theta] \\ \text{Sin}[\theta] & \text{Cos}[\theta] \end{pmatrix}$$

◆ Define the corresponding linear transformation using the definition $T(\mathbf{x}) = A\mathbf{x}$.

```
In[ ]:= T[{x_, y_}] = A.{x, y}
```

Out[ ]=

```
{x Cos[θ] - y Sin[θ], y Cos[θ] + x Sin[θ]}
```

◆ Find the rotation of the vector **u** by angle $\theta = 30^o$.

```
In[ ]:= θ = 25 Degree;
        Rθ = T[u]
```

Out[ ]=

```
{5 Cos[25 °] - 4 Sin[25 °], 4 Cos[25 °] + 5 Sin[25 °]}
```

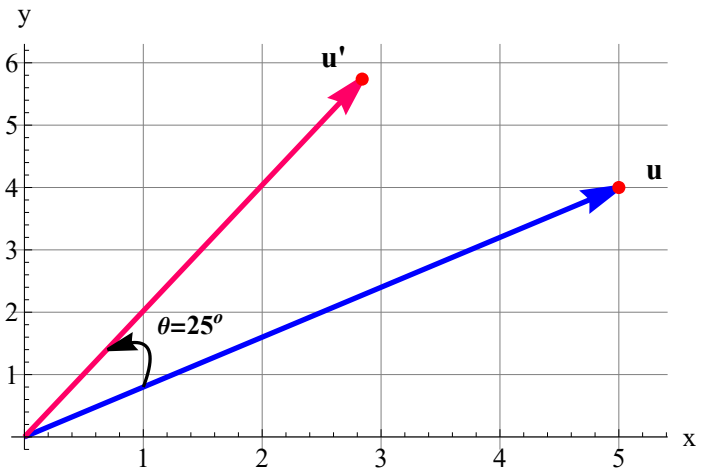
◆ Represent the vector **u** and its image as an arrow.

```
In[ ]:= u1 = {Arrowheads[0.05], Arrow[{{0, 0}, u}]};
        Rθ1 = {Arrowheads[0.05], Arrow[{{0, 0}, Rθ}]};
```

◆ Show the image of vector **u** under the rotation by angle θ.

In[ ]:=
```
Show[Graphics[{Thickness[0.008], Blue, u1, RGBColor[1, 0, 0.4], Rθ1,
   Red, PointSize[0.02], Point[u], Point[Rθ], Black, Thickness[0.005],
   Arrow[BezierCurve[{{1, 0.8}, {1.2, 1.8}, {0.7, 1.4}}]],
   Text[Style["u", Bold, FontFamily → "Times", FontSize → 16], u + 0.3],
   Text[Style["u'", Bold, FontFamily → "Times", FontSize → 16], {2.6, 6.1}],
   Text[Style["θ=25°", Bold, FontFamily → "Times", FontSize → 14], {1.4, 1.8}]},
  GridLines → Automatic], Axes → True, AxesLabel → {"x", "y"},
 LabelStyle → Directive[FontFamily → "Times", FontSize → 14, Black],
 ImageSize → 360, AspectRatio → 1 / GoldenRatio]
```

Out[ ]=



Vertical Shear Transformation

- Define the given geometric transformation with the matrix A.

In[ ]:=
```
A = {{1, 0}, {v, 1}}; A // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 0 \\ v & 1 \end{pmatrix}$$

- where v is a vertical shear factor.
- Define the corresponding linear transformation using the definition $T(\mathbf{x}) = A\mathbf{x}$.

In[ ]:=
```
T[{x_, y_}] = A.{x, y}
```

Out[ ]=
```
{x, v x + y}
```

- Find the vertical shear of the vector **u** by a factor of 3.

In[ ]:=
```
v = 3;
vSht = T[u]
```

Out[ ]=
```
{5, 19}
```

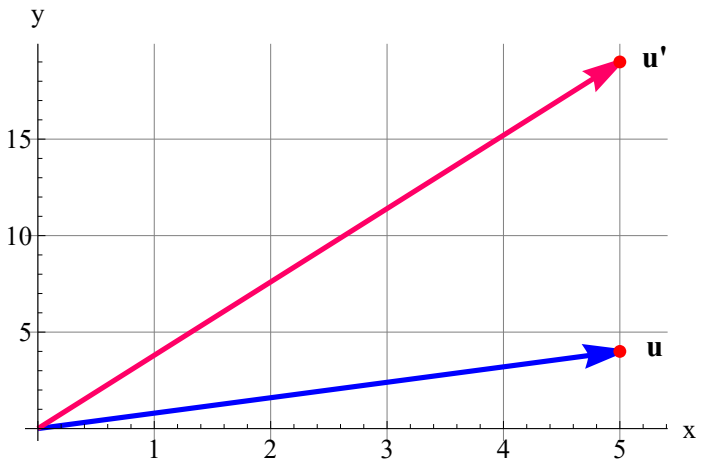
◆ Represent the vector **u** and its image as an arrow.

In[ ]:=
```
u1 = {Arrowheads[0.05], Arrow[{{0, 0}, u}]};
vSht1 = {Arrowheads[0.05], Arrow[{{0, 0}, vSht}]};
```

◆ Show the image of vector **u** under the vertical shearing.

In[ ]:=
```
Show[Graphics[{Thickness[0.008], Blue, u1, RGBColor[1, 0, 0.4],
    vSht1, Red, PointSize[0.02], Point[u], Point[vSht], Black,
    Text[Style["u", Bold, FontFamily → "Times", FontSize → 16], u + 0.3],
    Text[Style["u'", Bold, FontFamily → "Times", FontSize → 16], vSht + 0.3]},
  GridLines → Automatic], Axes → True, AxesLabel → {"x", "y"},
 LabelStyle → Directive[FontFamily → "Times", FontSize → 14, Black],
 ImageSize → 360, AspectRatio → 1 / GoldenRatio]
```

Out[ ]=



Horizontal Shear Transformation

◆ Define the given geometric transformation with the matrix A.

In[ ]:=
```
A = {{1, h}, {0, 1}}; A // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & h \\ 0 & 1 \end{pmatrix}$$

◆ where h is a horizontal shear factor.

◆ Define the corresponding linear transformation using the definition $T(\mathbf{x}) = A\mathbf{x}$.

In[ ]:=
```
T[{x_, y_}] = A.{x, y}
```

Out[ ]=

```
{x + h y, y}
```

◆ Find the horizontal shear of the vector **u** by a factor of 2.

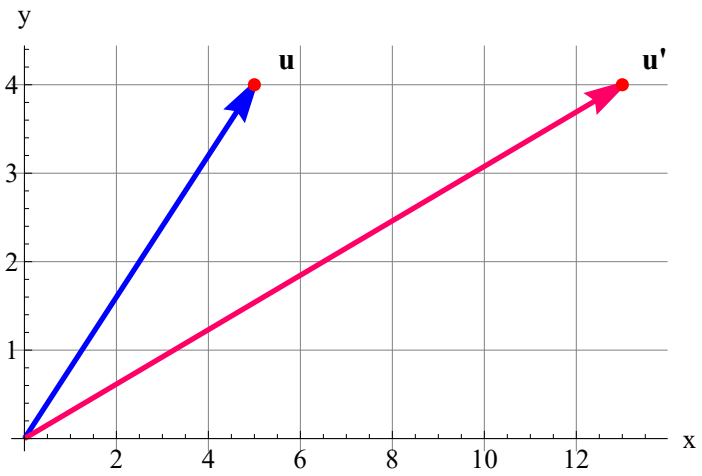
In[ ]:=
```
h = 2;
hSht = T[u]
```
Out[ ]=
```
{13, 4}
```

- Represent the vector **u** and its image as an arrow.

In[ ]:=
```
u1 = {Arrowheads[0.05], Arrow[{{0, 0}, u}]};
hSht1 = {Arrowheads[0.05], Arrow[{{0, 0}, hSht}]};
```

- Show the image of vector **u** under the horizontal shearing.

In[ ]:=
```
Show[Graphics[{Thickness[0.008], Blue, u1, RGBColor[1, 0, 0.4],
    hSht1, Red, PointSize[0.02], Point[u], Point[hSht], Black,
    Text[Style["u", Bold, FontFamily → "Times", FontSize → 16], {5.7, 4.3}],
    Text[Style["u'", Bold, FontFamily → "Times", FontSize → 16], {13.7, 4.3}]},
  GridLines → Automatic], Axes → True, AxesLabel → {"x", "y"},
 LabelStyle → Directive[FontFamily → "Times", FontSize → 14, Black],
 ImageSize → 360, AspectRatio → 1 / GoldenRatio]
```
Out[ ]=



Dilation

- Define the given geometric transformation with the matrix A.

In[ ]:=
```
A = {{d, 0}, {0, d}}; A // MatrixForm
```
Out[ ]//MatrixForm=

$$\begin{pmatrix} d & 0 \\ 0 & d \end{pmatrix}$$

- where d is a scale factor which determines how much larger or smaller the image will be compared to the original geometric object.
- Define the corresponding linear transformation using the definition $T(\mathbf{x}) = A\mathbf{x}$.

In[ ]:=
```
T[{x_, y_}] = A.{x, y}
```

Out[ ]=
```
{d x, d y}
```

- Find the dilation of the vector **u** by a factor of 1.5.

In[ ]:=
```
d = 1.5;
Du = T[u]
```

Out[ ]=
```
{7.5, 6.}
```

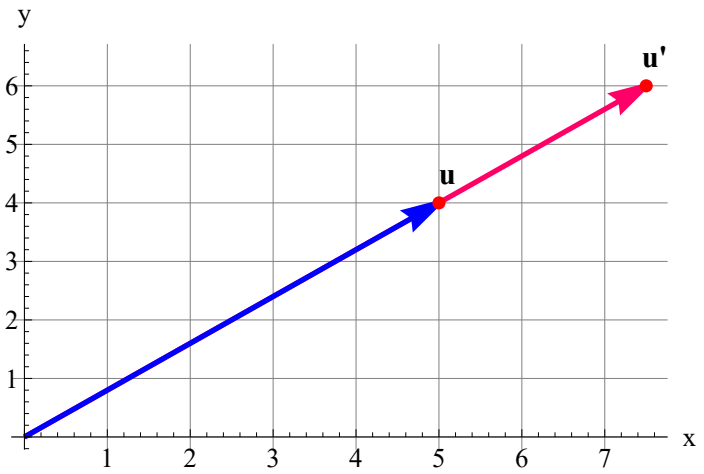
- Represent the vector **u** and its image as an arrow.

In[ ]:=
```
u1 = {Arrowheads[0.05], Arrow[{{0, 0}, u}]};
Du1 = {Arrowheads[0.05], Arrow[{{0, 0}, Du}]};
```

- Show the image of vector **u** under the dilation.

In[ ]:=
```
Show[Graphics[{Thickness[0.008], RGBColor[1, 0, 0.4],
    Du1, Blue, u1, Red, PointSize[0.02], Point[u], Point[Du], Black,
    Text[Style["u", Bold, FontFamily → "Times", FontSize → 16], {5.1, 4.5}],
    Text[Style["u'", Bold, FontFamily → "Times", FontSize → 16], {7.6, 6.5}]},
  GridLines → Automatic], Axes → True, AxesLabel → {"x", "y"},
 LabelStyle → Directive[FontFamily → "Times", FontSize → 14, Black],
 ImageSize → 360, AspectRatio → 1 / GoldenRatio]
```

Out[ ]=



Projection onto the x-Axis

- Define the given geometric transformation with the matrix *A*.

In[ ]:=
```
A = {{1, 0}, {0, 0}}; A // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

◆ Define the corresponding linear transformation using the definition $T(\mathbf{x}) = A\mathbf{x}$.

In[ ]:=
```
T[{x_, y_}] = A.{x, y}
```

Out[ ]=
```
{x, 0}
```

◆ Find the projetion of the vector **u** onto the x-axis.

In[ ]:=
```
Px = T[u]
```

Out[ ]=
```
{5, 0}
```

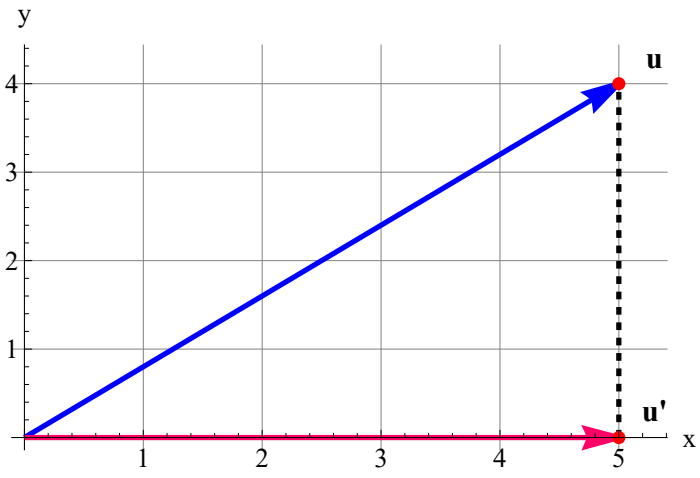
◆ Represent the vector **u** and its image as an arrow.

In[ ]:=
```
u1 = {Arrowheads[0.05], Arrow[{{0, 0}, u}]};
Px1 = {Arrowheads[0.05], Arrow[{{0, 0}, Px}]};
```

◆ Show the image of vector **u** under the projection onto the x-axis.

In[ ]:=
```
Show[Graphics[{Thickness[0.008], Dashed, Line[{{5, 0}, {5, 4}}], Dashing[None], Blue, u1,
    RGBColor[1, 0, 0.4], Px1, Red, PointSize[0.02], Point[u], Point[Px], Black,
    Text[Style["u", Bold, FontFamily → "Times", FontSize → 16], u + 0.3],
    Text[Style["u'", Bold, FontFamily → "Times", FontSize → 16], Px + 0.3]},
  GridLines → Automatic], Axes → True, AxesLabel → {"x", "y"},
 LabelStyle → Directive[FontFamily → "Times", FontSize → 14, Black],
 ImageSize → 360, AspectRatio → 1 / GoldenRatio]
```

Out[ ]=



Projection onto the y-Axis

◆ Define the given geometric transformation with the matrix A.

In[ ]:=
```
A = {{0, 0}, {0, 1}}; A // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

◆ Define the corresponding linear transformation using the definition $T(\mathbf{x}) = A\mathbf{x}$.

In[ ]:=
```
T[{x_, y_}] = A.{x, y}
```

Out[ ]=
```
{0, y}
```

◆ Find the projetion of the vector **u** onto the y-axis.

In[ ]:=
```
Py = T[u]
```

Out[ ]=
```
{0, 4}
```

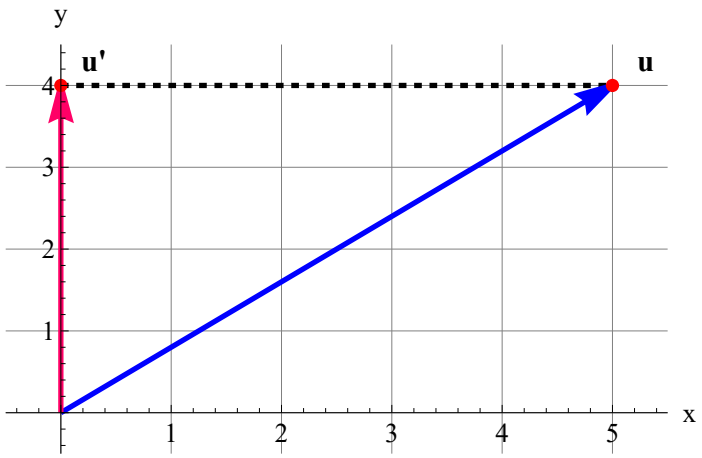
◆ Represent the vector **u** and its image as an arrow.

In[ ]:=
```
u1 = {Arrowheads[0.05], Arrow[{{0, 0}, u}]};
Py1 = {Arrowheads[0.05], Arrow[{{0, 0}, Py}]};
```

◆ Show the image of vector **u** under the projection onto the y-axis.

In[ ]:=
```
Show[Graphics[{Thickness[0.008], Black, Dashed, Line[{{0, 4}, {5, 4}}], Dashing[None],
    Blue, u1, RGBColor[1, 0, 0.4], Py1, Red, PointSize[0.02], Point[u], Point[Py], Black,
    Text[Style["u", Bold, FontFamily → "Times", FontSize → 16], u + 0.3],
    Text[Style["u'", Bold, FontFamily → "Times", FontSize → 16], Py + 0.3]},
  GridLines → Automatic], Axes → True, PlotRange → {{-0.5, 5.5}, {-0.5, 4.5}},
 AxesLabel → {"x", "y"}, LabelStyle → Directive[FontFamily → "Times", FontSize → 14, Black],
 ImageSize → 360, AspectRatio → 1 / GoldenRatio]
```

Out[ ]=



In[ ]:=

Summary

After completing this chapter, you should be able to

- analyze vectors, simple vector operations, and geometry of vectors in Mathematica.
- analyze span and linear independence of vectors in Mathematica.

- analyze the dot product of two vectors and related applications in Mathematica.
- analyze linear transformations in Mathematica.
- learn and use information, tools, and technology to solve problems.

Week 13: Linear Systems of ODEs

How to solve a system of differential equations?

Table of Contents

Commands list

- Integrate
- Solve
- DSolve
- LaplaceTransform
- InverseLaplaceTransform
- RowReduce
- Transpose
- Join
- Inverse
- CharacteristicPolynomial
- Eigenvalues
- Eigenvectors
- Eigensystem
- Wronskian
- DiagoalizableMatrixQ

Prerequisite: Eigenvalues and Eigenvectors

The Wolfram Language includes built-in functions that are helpful in solving the systems of differential equations. This section dicusses the 3 methods for solving the homogeneous linear

systems ODEs of first order with initial condition. Therefore it is recommended to visit and read the sections for Weeks 1, 6, and 11 of this guidebook to learn more about these methods.

Week 1 | 1st Order ODEs
Week 6 | Laplace Transforms-2 (Solving ODEs)
Week 11 | Eigenvalues and Eigenvectors

Homogeneous First-Order Linear System of ODEs with IVP

Example 13.1: Consider the reaction network of two irreversible (one-way), first-order reaction in series:

$$A \xrightarrow{k_1} B \xrightarrow{k_2} C$$

Suppose at time $t = 0$, we have initial conditions $[A] = [A_0]$, $[B] = 0$, $[C] = 0$, where $[A]$, $[B]$, and $[C]$ denote the concentrations of species A, B, and C, respectively. Using the Guldberg-Waage form of the reaction rates to describes the network $A \to B \to C$ gives for volume:

$$\frac{d[A]}{dt} = -k_1[A]$$

$$\frac{d[B]}{dt} = k_1[A] - k_2[B]$$

$$\frac{d[C]}{dt} = k_2[B]$$

Solve the above rate equations (system of ODEs) to determine the concentrations $[A]$, $[B]$, ar $[C]$ as a function of time.

Solution: Instaed of using notations like $[A]$, $[B]$, and $[C]$, let's introduce $y_1 = [A]$, $y_2 = [B]$, and $y_3 = [C]$ and rewrite the system of ODEs as

$$\frac{dy_1}{dt} = y_1' = -k_1 y_1$$

$$\frac{dy_2}{dt} = y_2' = k_1 y_1 - k_2 y_2$$

$$\frac{dy_3}{dt} = y_3' = k_2\, y_2$$

with initial conditions $y_1(0) = [A_0]$, $y_2(0) = 0$, and $y_3(0) = 0$.

Method 1: Sepataion of Variables

Since the first ODE are separable equations, we will solve it by the method of separation of variables.

- **Step 1.** The first ODE is separable: $y_1^{-1}\, dy_1 = -k_1\, dt$.

```
In[ ]:= ClearAll["Global`*"]
        expr = y1'[t] + k1 * y1[t]
Out[ ]=
        k1 y1[t] + y1'[t]
```

- **Step 2.** Integrate the left-hand-side with respect to y_1.

```
In[ ]:= LHS = Integrate[y1^-1, y1]
Out[ ]=
        Log[y1]
```

- **Step 3.** Integrate the right-hand-side with respect to t.

```
In[ ]:= RHS = Integrate[-k1, t]
Out[ ]=
        -k1 t
```

- **Step 4.** By integration we got: $\ln y_1 = -k_1\, t + C$. Solve the expression to get the general solution to ODE.

```
In[ ]:= Solve[LHS == RHS + C, y1, Reals]
Out[ ]=
        {{y1 → ℯ^(C-k1 t)}}

In[ ]:= y1genSoln = C * Exp[-k1 * t]
Out[ ]=
        C ℯ^(-k1 t)
```

- **Step 5.** Use initial value condition $y_1(0) = A_0$ to find the particular solution.

```
In[ ]:= y1₀ = y1genSoln /. t → 0
Out[ ]=
        C
```

- Solve for the arbitrary constant.

```
In[ ]:= Solve[y1_0 == A0, C]
Out[ ]=
{{C → A0}}
```

- Substitute the arbitrary constant to the general solution.

```
In[ ]:= y1partSol = y1genSoln /. C → A0;
        y_1[t] == y1partSol
Out[ ]=
```

$$y_1[t] == A0\, e^{-k1\, t}$$

- **Step 6.** Verify the obtained solution.

```
In[ ]:= dsol1 = DSolve[{expr == 0, y1[0] == A0}, y1[t], t]
Out[ ]=
{{y1[t] → A0 e^(-k1 t)}}

In[ ]:= FullSimplify[y1partSol == A0 e^(-k1 t)]
Out[ ]=
True
```

Now we move to the second ODE. Substituting the solution $y_1(t)$ into the second ODE, $y_2' = k_1\, y_1 - k_2\, y_2$, we get a non-homogeneous linear ODE of first order.

- **Step 1.** Define the second ODE and substitute the solution $y_1(t)$.

```
In[ ]:= expr2 = y2'[t] - k1 * y1[t] + k2 * y2[t]
Out[ ]=
-k1 y1[t] + k2 y2[t] + y2'[t]

In[ ]:= expr2 = expr2 /. y1[t] → A0 * Exp[-k1 * t]
Out[ ]=
-A0 e^(-k1 t) k1 + k2 y2[t] + y2'[t]
```

- Now we have a non-homogeneous linear ODE of the form $y' + p(t)\, y = r(t)$.
- **Step 2.** Define the $p(t)$ and $r(t)$.

```
In[ ]:= p = k2;
        r = A0 * Exp[-k1 * t] * k1;
```

- **Step 3.** Calculate the integrting factor h as: $h = \int p(t)\, dt$.

```
In[ ]:= h = Integrate[p, t]
Out[ ]=
k2 t
```

- **Step 4.** Find the general solution to given ODE: $y(t) = e^{-h} \int e^{h}\, r\, dt + c\, e^{-h}$.

```
In[ ]:= y2genSol = Exp[-h] * Integrate[Exp[h] * r, t] + Exp[-h] * C
```

Out[ ]=

$$C\, e^{-k2\, t} + \frac{A0\, e^{-k2\, t + (-k1+k2)\, t}\, k1}{-k1 + k2}$$

- **Step 5.** Find the particular solution by the initial condition $y_2(0) = 0$.

```
In[ ]:= y2₀ = y2genSol /. t → 0
```

Out[ ]=

$$C + \frac{A0\, k1}{-k1 + k2}$$

- Solve for the arbitrary constant.

```
In[ ]:= Solve[y2₀ == 0, C]
```

Out[ ]=

$$\left\{\left\{C \to \frac{A0\, k1}{k1 - k2}\right\}\right\}$$

- Substitute the arbitrary constant to the general solution.

```
In[ ]:= y2partSol = y2genSol /. C → k1/(k1 - k2) A0;
        y2[t] == FullSimplify[y2partSol]
```

Out[ ]=

$$y_2[t] == \frac{A0\, e^{-k2\, t}\left(-1 + e^{(-k1+k2)\, t}\right) k1}{-k1 + k2}$$

- **Step 6.** Verify the obtained solution.

```
In[ ]:= dsol2 = DSolve[{expr2 == 0, y2[0] == 0}, y2[t], t]
```

Out[ ]=

$$\left\{\left\{y2[t] \to \frac{A0\, e^{-k2\, t}\left(-1 + e^{(-k1+k2)\, t}\right) k1}{-k1 + k2}\right\}\right\}$$

```
In[ ]:= FullSimplify[y2partSol == A0 e^(-k2 t) (-1 + e^((-k1+k2) t)) k1/(-k1 + k2)]
```

Out[ ]=

```
True
```

Substituting the solution $y_2(t)$ into the third ODE, $y_3' = k_2\, y_2$, we get again a separable equation which can be solved by separation of variables.

- **Step 1.** Define the third ODE and substitute the solution $y_2(t)$.

```
In[ ]:= expr3 = y3'[t] - k2 * y2[t]
```

Out[ ]=

```
-k2 y2[t] + y3'[t]
```

```
In[ ]:= expr3 = expr3 /. y2[t] → A0 * k1 * Exp[-k2 * t] (-1 + Exp[-k1 * t + k2 * t]) / (-k1 + k2)
```

```
Out[ ]=
-(A0 e^(-k2 t) (-1 + e^(-k1 t+k2 t)) k1 k2)/(-k1 + k2) + y3'[t]
```

```
In[ ]:= expr3 // FullSimplify
```

```
Out[ ]=
(A0 (e^(-k1 t) - e^(-k2 t)) k1 k2)/(k1 - k2) + y3'[t]
```

- Now we have a separable equation: $dy_3 = -\frac{A_0\left(e^{-k_1 t}-e^{-k_2 t}\right) k_1 k_2}{k_1-k_2}\, dt$.

- **Step 2.** Integrate the left-hand-side with respect to y_3.

```
In[ ]:= LHS = Integrate[1, y3]
```

```
Out[ ]=
y3
```

- **Step 3.** Integrate the right-hand-side with respect to t.

```
In[ ]:= RHS = Integrate[-(A0 * (Exp[-k1 * t] - Exp[-k2 * t]) * k1 * k2)/(k1 - k2), t]
```

```
Out[ ]=
-(A0 k1 (-e^(-k1 t)/k1 + e^(-k2 t)/k2) k2)/(k1 - k2)
```

```
In[ ]:= RHS = FullSimplify[RHS]
```

```
Out[ ]=
-(A0 (e^(-k2 t) k1 - e^(-k1 t) k2))/(k1 - k2)
```

- **Step 4.** By integration we got: $y_3 = -\frac{A_0\left(e^{-k_2 t}\, k_1-e^{-k_1 t}\, k_2\right)}{k_1-k_2} + C$. Solve the expression for y_3 to get the general solution to ODE.

```
In[ ]:= Solve[LHS == RHS + C, y3]
```

```
Out[ ]=
{{y3 → C - (A0 (e^(-k2 t) k1 - e^(-k1 t) k2))/(k1 - k2)}}
```

```
In[ ]:= y3genSoln = C - (A0 * (Exp[-k2 * t] * k1 - Exp[-k1 * t] * k2))/(k1 - k2)
```

```
Out[ ]=
C - (A0 (e^(-k2 t) k1 - e^(-k1 t) k2))/(k1 - k2)
```

◆ **Step 5.** Use initial value condition $y_3(0) = 0$ to find the particular solution.

```
In[ ]:= y3₀ = y3genSoln /. t → 0
Out[ ]=
        -A0 + C
```

◆ Solve for the arbitrary constant.

```
In[ ]:= Solve[y3₀ == 0, C]
Out[ ]=
        {{C → A0}}
```

◆ Substitute the arbitrary constant to the general solution.

```
In[ ]:= y3partSol = y3genSoln /. C → A0;
        y3 == y3partSol
Out[ ]=
```

$$y_3 == A0 - \frac{A0\left(e^{-k2\,t}\,k1 - e^{-k1\,t}\,k2\right)}{k1 - k2}$$

◆ **Step 6.** Verify the obtained solution.

```
In[ ]:= dsol3 = FullSimplify[DSolve[{expr3 == 0, y3[0] == 0}, y3[t], t]]
Out[ ]=
```

$$\left\{\left\{y3[t] \to \frac{A0\left(k1 - e^{-k2\,t}\,k1 + \left(-1 + e^{-k1\,t}\right)k2\right)}{k1 - k2}\right\}\right\}$$

```
In[ ]:= FullSimplify[y3partSol == A0 (k1 - e^(-k2 t) k1 + (-1 + e^(-k1 t)) k2)/(k1 - k2)]
Out[ ]=
        True
```

Method 2: Laplace Transform

Now lets solve the system od ODEs using Laplace transforms.

$$\frac{dy_1}{dt} = y_1' = -k_1\,y_1$$

$$\frac{dy_2}{dt} = y_2' = k_1\,y_1 - k_2\,y_2$$

$$\frac{dy_3}{dt} = y_3' = k_2\,y_2$$

with initial conditions $y_1(0) = [A_0]$, $y_2(0) = 0$, and $y_3(0) = 0$.

◆ **Step 1.1.** Define the first ODE as an equation and its intial condition as a substitution.

```
In[ ]:= ClearAll["Global`*"]
        ode1 = y1'[t] == -k1 * y1[t]
Out[ ]=
        y1'[t] == -k1 y1[t]

In[ ]:= IC1 = {y1[0] → A0}
Out[ ]=
        {y1[0] → A0}
```

- **Step 2.1.** Compute the Laplace transforms of both sides of the ODE and substitute the initial condition.

```
In[ ]:= LT1 = LaplaceTransform[ode1, t, s] /. IC1
Out[ ]=
        -A0 + s LaplaceTransform[y1[t], t, s] == -k1 LaplaceTransform[y1[t], t, s]
```

- Replace $\mathcal{L}\{y_1\}$ with Y_1.

```
In[ ]:= eqnforY1 = LT1 /. LaplaceTransform[y1[t], t, s] → Y1[s]
Out[ ]=
        -A0 + s Y1[s] == -k1 Y1[s]
```

- **Step 1.2.** Define the second ODE as an equation and its intial condition as a substitution.

```
In[ ]:= ode2 = y2'[t] == k1 * y1[t] - k2 * y2[t]
Out[ ]=
        y2'[t] == k1 y1[t] - k2 y2[t]

In[ ]:= IC2 = {y2[0] → 0}
Out[ ]=
        {y2[0] → 0}
```

- **Step 2.2.** Compute the Laplace transforms of both sides of the ODE and substitute the initial condition.

```
In[ ]:= LT2 = LaplaceTransform[ode2, t, s] /. IC2
Out[ ]=
        s LaplaceTransform[y2[t], t, s] ==
         k1 LaplaceTransform[y1[t], t, s] - k2 LaplaceTransform[y2[t], t, s]
```

- Replace $\mathcal{L}\{y_1\}$ with Y_1 and $\mathcal{L}\{y_2\}$ with Y_2.

```
In[ ]:= eqnforY2 =
         LT2 /. {LaplaceTransform[y1[t], t, s] → Y1[s], LaplaceTransform[y2[t], t, s] → Y2[s]}
Out[ ]=
        s Y2[s] == k1 Y1[s] - k2 Y2[s]
```

- **Step 1.3.** Define the third ODE as an equation and its intial condition as a substitution.

```
In[ ]:= ode3 = y3'[t] == k2 * y2[t]
Out[ ]=
        y3'[t] == k2 y2[t]
```

```
In[ ]:= IC3 = {y3[0] → 0}
Out[ ]=
{y3[0] → 0}
```

◆ **Step 2.3.** Compute the Laplace transforms of both sides of the ODE and substitute the initial condition.

```
In[ ]:= LT3 = LaplaceTransform[ode3, t, s] /. IC3
Out[ ]=
s LaplaceTransform[y3[t], t, s] == k2 LaplaceTransform[y2[t], t, s]
```

◆ Replace $\mathcal{L}\{y_2\}$ with Y_2 and $\mathcal{L}\{y_3\}$ with Y_3.

```
In[ ]:= eqnforY3 =
 LT3 /. {LaplaceTransform[y2[t], t, s] → Y2[s], LaplaceTransform[y3[t], t, s] → Y3[s]}
Out[ ]=
s Y3[s] == k2 Y2[s]
```

◆ **Step 3.** Solve the obtained algebraic system of equations.

```
In[ ]:= sys = {eqnforY1, eqnforY2, eqnforY3}; sys // Column
Out[ ]=
-A0 + s Y1[s] == -k1 Y1[s]
s Y2[s] == k1 Y1[s] - k2 Y2[s]
s Y3[s] == k2 Y2[s]
```

```
In[ ]:= soln = Solve[sys, {Y1[s], Y2[s], Y3[s] }]
Out[ ]=
{{Y1[s] → A0/(k1 + s), Y2[s] → A0 k1/((k1 + s) (k2 + s)), Y3[s] → A0 k1 k2/(s (k1 + s) (k2 + s))}}
```

◆ The Laplace transforms of the solutions are:

```
In[ ]:= Y1sol[s_] := Y1[s] /. soln[[1, 1]]; Y1sol[s]
Out[ ]=
A0/(k1 + s)
```

```
In[ ]:= Y2sol[s_] := Y2[s] /. soln[[1, 2]]; Y2sol[s]
Out[ ]=
A0 k1/((k1 + s) (k2 + s))
```

```
In[ ]:= Y3sol[s_] := Y3[s] /. soln[[1, 3]]; Y3sol[s]
Out[ ]=
A0 k1 k2/(s (k1 + s) (k2 + s))
```

◆ **Step 4.** Take the inverse transforms of Y_1, Y_2, and Y_3 to get the solution to the given system of ODEs.

In[ ]:=
```
y1Soln[t_] = InverseLaplaceTransform[Y1sol[s], s, t]; y1 == y1Soln[t]
```

Out[ ]=

$$y_1 == A0\, e^{-k1\, t}$$

In[ ]:=
```
y2Soln[t_] = InverseLaplaceTransform[Y2sol[s], s, t]; y2 == y2Soln[t]
```

Out[ ]=

$$y_2 == -\frac{A0\left(e^{-k1\, t} - e^{-k2\, t}\right) k1}{k1 - k2}$$

In[ ]:=
```
y3Soln[t_] = InverseLaplaceTransform[Y3sol[s], s, t]; y3 == FullSimplify[y3Soln[t]]
```

Out[ ]=

$$y_3 == \frac{A0\left(k1 - e^{-k2\, t}\, k1 + \left(-1 + e^{-k1\, t}\right) k2\right)}{k1 - k2}$$

◆ **Step 5.** Verify the obtained results.

In[ ]:=
```
dsol = FullSimplify[
  DSolve[{ode1, ode2, ode3, y1[0] == A0, y2[0] == 0, y3[0] == 0}, {y1[t], y2[t], y3[t]}, t]]
```

Out[ ]=

$$\left\{\left\{y1[t] \to A0\, e^{-k1\, t},\ y2[t] \to -\frac{A0\left(e^{-k1\, t} - e^{-k2\, t}\right) k1}{k1 - k2},\ y3[t] \to \frac{A0\left(k1 - e^{-k2\, t}\, k1 + \left(-1 + e^{-k1\, t}\right) k2\right)}{k1 - k2}\right\}\right\}$$

In[ ]:=
```
FullSimplify[y1Soln[t] == y1[t] /. dsol[[1, 1]]]
```

Out[ ]=
```
True
```

In[ ]:=
```
FullSimplify[y2Soln[t] == y2[t] /. dsol[[1, 2]]]
```

Out[ ]=
```
True
```

In[ ]:=
```
FullSimplify[y3Soln[t] == y3[t] /. dsol[[1, 3]]]
```

Out[ ]=
```
True
```

Method 3: Eigenvalues and Eigenvectors

The given system of linear first-order ODEs can be expressed as $\mathbf{y}' = A\mathbf{y}$:

$$\begin{pmatrix} y_1' \\ y_2' \\ y_3' \end{pmatrix} = \begin{pmatrix} -k_1 & 0 & 0 \\ k_1 & -k_2 & 0 \\ 0 & k_2 & 0 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

◆ **Step 1.** Define the coefficient matrix A.

```
In[ ]:= ClearAll["Global`*"]
        A = {{-k1, 0, 0}, {k1, -k2, 0}, {0, k2, 0}}; A // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -k1 & 0 & 0 \\ k1 & -k2 & 0 \\ 0 & k2 & 0 \end{pmatrix}$$

- **Step 2.** Determine the eigenvalues of A by calculating the characteristic polynomial, **det (A – λI$_3$)**.

```
In[ ]:= Factor[CharacteristicPolynomial[A, λ]]
```

Out[ ]=

$$-\lambda\ (k1+\lambda)\ (k2+\lambda)$$

- The eigenvalues for a matrix A are given by the roots of the characteristic equation.

```
In[ ]:= Solve[Det[A - λ * IdentityMatrix[3]] == 0 == 0, λ]
```

Out[ ]=

$$\{\{\lambda \to 0\}, \{\lambda \to -k1\}, \{\lambda \to -k2\}\}$$

- Results show that the matrix A has three distinct eigenvalues, $\lambda_1 = 0$, $\lambda_2 = -k_1$, and $\lambda_3 = -k_2$.

```
In[ ]:= {λ1, λ2, λ3} = Eigenvalues[A]
```

Out[ ]=

$$\{0, -k1, -k2\}$$

- **Step 3.** Find the eigenvectors associated with corresponding eigenvalues.

```
In[ ]:= {u1, u2, u3} = Eigenvectors[A];
        {MatrixForm[u1], MatrixForm[u2], MatrixForm[u3]}
```

Out[ ]=

$$\left\{\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} -\frac{-k1+k2}{k2} \\ -\frac{k1}{k2} \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix}\right\}$$

Theorem 13.1: General Solution to the 1st Order Linear System of ODEs.

Suppose that $\mathbf{y}' = A\mathbf{y}$ is a first-order linear system of differential equations. If A is an $n \times n$ diagonalizable matrix, then the general solution to the system is given by

$$\mathbf{y} = c_1 e^{\lambda_1 t}\mathbf{u}_1 + \cdots + c_n e^{\lambda_n t}\mathbf{u}_n$$

where $\mathbf{u}_1, \ldots, \mathbf{u}_n$ are n linearly independent eigenvectors with associayed eigenvalues $\lambda_1, \ldots, \lambda_n$ and $c_1, \ldots, c_n$ are constants.

- **Step 4.** Check the matrix for diagonalizability.

In[ ]:=
```
DiagonalizableMatrixQ[A]
```
Out[ ]=
```
True
```

- **Step 5.** By **Theorem 13.1**, the corresponding solutions of the differential equations are:

In[ ]:=
```
sol1 = u1 * Exp[λ1 * t]; sol1 // MatrixForm
```
Out[ ]//MatrixForm=
$$\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

In[ ]:=
```
sol2 = u2 * Exp[λ2 * t]; sol2 // MatrixForm
```
Out[ ]//MatrixForm=
$$\begin{pmatrix} -\frac{e^{-k1\,t}\,(-k1+k2)}{k2} \\ -\frac{e^{-k1\,t}\,k1}{k2} \\ e^{-k1\,t} \end{pmatrix}$$

In[ ]:=
```
sol3 = u3 * Exp[λ3 * t]; sol3 // MatrixForm
```
Out[ ]//MatrixForm=
$$\begin{pmatrix} 0 \\ -e^{-k2\,t} \\ e^{-k2\,t} \end{pmatrix}$$

- **Step 6.** The **Wronskian** determinant can be used to check if these functions form a fundamental solution set:

In[ ]:=
```
Wronskian[{sol1, sol2, sol3}, t]
```
Out[ ]=
$$\frac{e^{-((k1+k2)\,t)}\,(-k1+k2)}{k2}$$

- Since the Wronskian detrminant is not equal to zero for real values of ***t***, these fu form a fundamental solution set.
- **Step 7.** By **Theorem 13.1**, the general solution of the system is:

$$\mathbf{y} = c_1\, e^{\lambda_1 t}\, \mathbf{u}_1 + c_2\, e^{\lambda_2 t}\, \mathbf{u}_2 + c_3\, e^{\lambda_3 t}\, \mathbf{u}_3$$

In[ ]:=
```
genSol = c1 * sol1 + c2 * sol2 + c3 * sol3;
MatrixForm[{y1, y2, y3}] ==
 c1 * MatrixForm[sol1] + c2 * MatrixForm[sol2] + c3 * MatrixForm[sol3]
```
Out[ ]=
$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} == c1 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} + c3 \begin{pmatrix} 0 \\ -e^{-k2\,t} \\ e^{-k2\,t} \end{pmatrix} + c2 \begin{pmatrix} -\frac{e^{-k1\,t}\,(-k1+k2)}{k2} \\ -\frac{e^{-k1\,t}\,k1}{k2} \\ e^{-k1\,t} \end{pmatrix}$$

◆ **Step 8.** Find the particular solution with initial conditions: $y_1(0) = A_0$, $y_2(0) = 0$, $y_3(0) = 0$.

◆ Define $\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}_0 = \begin{pmatrix} A_0 \\ 0 \\ 0 \end{pmatrix}$:

```
In[ ]:= IC = {{A0}, {0}, {0}}; IC // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} A0 \\ 0 \\ 0 \end{pmatrix}$$

◆ Substitute the value $t_0 = 0$:

```
In[ ]:= sol1 = {{sol1[[1]]}, {sol1[[2]]}, {sol1[[3]]}} /. t → 0;
        sol1 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

```
In[ ]:= sol2 = {{sol2[[1]]}, {sol2[[2]]}, {sol2[[3]]}} /. t → 0;
        sol2 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -\frac{-k1+k2}{k2} \\ -\frac{k1}{k2} \\ 1 \end{pmatrix}$$

```
In[ ]:= sol3 = {{sol3[[1]]}, {sol3[[2]]}, {sol3[[3]]}} /. t → 0;
        sol3 // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix}$$

◆ Define the corresponding augmented matrix of the system with initial conditions.

```
In[ ]:= AugMat =
         Transpose[Join[Transpose[sol1], Transpose[sol2], Transpose[sol3], Transpose[IC]]];
        AugMat // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 & -\frac{-k1+k2}{k2} & 0 & A0 \\ 0 & -\frac{k1}{k2} & -1 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

◆ Reduce the augmented matrix to row echelon form.

```
In[ ]:= RowReduce[AugMat] // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 0 & A0 \\ 0 & 1 & 0 & \frac{A0\,k2}{k1-k2} \\ 0 & 0 & 1 & -\frac{A0\,k1}{k1-k2} \end{pmatrix}$$

- Perform a back substitution and find the arbitrary constants.

```
In[ ]:= c3 = -k1/(k1 - k2) A0;
        c2 = k2/(k1 - k2) A0;
        c1 = A0;
```

- Obtain the particular solution.

```
In[ ]:= partSol = Simplify[genSol /. {c1 → A0, c2 → k2/(k1 - k2) A0, c3 → -k1/(k1 - k2) A0}];
        MatrixForm[{y1, y2, y3}] == MatrixForm[partSol]
```

Out[ ]=

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} == \begin{pmatrix} A0\,e^{-k1\,t} \\ -\frac{A0\,(e^{-k1\,t}-e^{-k2\,t})\,k1}{k1-k2} \\ \frac{A0\,(k1-e^{-k2\,t}\,k1+(-1+e^{-k1\,t})\,k2)}{k1-k2} \end{pmatrix}$$

- Or alternatively:

```
In[ ]:= MatrixForm[{y1, y2, y3}] == A0 * MatrixForm[partSol /. A0 → 1]
```

Out[ ]=

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} == A0 \begin{pmatrix} e^{-k1\,t} \\ -\frac{\left(e^{-k1\,t}-e^{-k2\,t}\right)\,k1}{k1-k2} \\ \frac{k1-e^{-k2\,t}\,k1+\left(-1+e^{-k1\,t}\right)\,k2}{k1-k2} \end{pmatrix}$$

- **Step 9.** Verify the solution using **DSolve** function.

```
In[ ]:= ClearAll["Global`*"]
        dsol = FullSimplify[DSolve[{y1'[t] == -k1 * y1[t], y2'[t] == k1 * y1[t] - k2 * y2[t],
            y3'[t] == k2 * y2[t], y1[0] == A0, y2[0] == 0, y3[0] == 0}, {y1[t], y2[t], y3[t]}, t]]
```

Out[ ]=

$$\left\{\left\{y1[t] \to A0\,e^{-k1\,t},\ y2[t] \to -\frac{A0\,\left(e^{-k1\,t}-e^{-k2\,t}\right)\,k1}{k1-k2},\ y3[t] \to \frac{A0\,\left(k1-e^{-k2\,t}\,k1+\left(-1+e^{-k1\,t}\right)\,k2\right)}{k1-k2}\right\}\right\}$$

Another technique for solving initial-value problems and ample solution were taken from the book **"Differential Equations with Mathematica, 5th Ed."** by **Martha L. Abell** and **James P. Braselton**, Chapter 6.

Theorem 13.2: Solving Initial-Value Problems

Let $\mathbf{\Phi}(t)$ be a fundamnetal matrix for the system of equations $\mathbf{X}'(t) = \mathbf{A}(t)\,\mathbf{X}(t)$ and define it as follows

$$\mathbf{\Phi} = \begin{pmatrix} e^{\lambda_1 t}\,\mathbf{u}_1 & e^{\lambda_1 t}\,\mathbf{u}_1 & \cdots & e^{\lambda_n t}\,\mathbf{u}_n \end{pmatrix}$$

Then, a general solution is $\mathbf{X}'(t) = \mathbf{\Phi}(t)\,\mathbf{C}$, where $\mathbf{C}$ is a constant vector. If the initial condition $\mathbf{X}(0) = \mathbf{X}_0$ is given, then

$$\mathbf{X}(0) = \mathbf{\Phi}(0)\,\mathbf{C},$$

$$\mathbf{X}_0 = \mathbf{\Phi}(0)\,\mathbf{C},$$

$$\mathbf{C} = \mathbf{\Phi}^{-1}(0)\,\mathbf{X}_0.$$

Therefore, the the solution to the initial-value problem $\begin{cases} \mathbf{X}'(t) = \mathbf{A}(t)\,\mathbf{X}(t) \\ \mathbf{X}(0) = \mathbf{X}_0 \end{cases}$ is $\mathbf{X}(t) = \mathbf{\Phi}(t) \times \mathbf{\Phi}^{-1}(0)\,\mathbf{X}_0$.

◆ **Step 1.** Define the coefficient matrix A.

In[ ]:=
```
ClearAll["Global`*"]
A = {{-k1, 0, 0}, {k1, -k2, 0}, {0, k2, 0}}; A // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -k1 & 0 & 0 \\ k1 & -k2 & 0 \\ 0 & k2 & 0 \end{pmatrix}$$

◆ **Step 2.** Compute the eigenvalues and corresponding eigenvectors of A.

In[ ]:=
```
s1 = Eigensystem[A]
```

Out[ ]=

$$\left\{\{0, -k1, -k2\}, \left\{\{0, 0, 1\}, \left\{-\frac{-k1 + k2}{k2}, -\frac{k1}{k2}, 1\right\}, \{0, -1, 1\}\right\}\right\}$$

◆ Results show that eigenvalues are $\lambda_1 = 0$, $\lambda_2 = -k_1$, and $\lambda_3 = -k_2$ with corresponding eigenvectors, respectively:

$$\mathbf{u}_1 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad \mathbf{u}_2 = \begin{pmatrix} -\frac{-k_1+k_2}{k_2} \\ -\frac{k_1}{k_2} \\ 1 \end{pmatrix}, \quad \text{and} \quad \mathbf{u}_3 = \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix}.$$

◆ **Step 3.** Define the fundamental matrix as follows: $\mathbf{\Phi} = \begin{pmatrix} e^{\lambda_1 t}\,\mathbf{u}_1 & e^{\lambda_1 t}\,\mathbf{u}_1 & \cdots & e^{\lambda_n t}\,\mathbf{u}_n \end{pmatrix}$.

```
In[ ]:= phi[t_] = {s1[[2, 1]] * Exp[s1[[1, 1]] * t],
        s1[[2, 2]] * Exp[s1[[1, 2]] * t], s1[[2, 3]] * Exp[s1[[1, 3]] * t]} // Transpose;
      phi[t] // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 & -\frac{e^{-k1\,t}\,(-k1+k2)}{k2} & 0 \\ 0 & -\frac{e^{-k1\,t}\,k1}{k2} & -e^{-k2\,t} \\ 1 & e^{-k1\,t} & e^{-k2\,t} \end{pmatrix}$$

- **Step 4.** Calculate Φ^{-1} with **Inverse** function.

```
In[ ]:= Inverse[phi[t]] // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 1 & 1 \\ -\frac{e^{-k2\,t}}{e^{-k1\,t-k2\,t}-\frac{e^{-k1\,t-k2\,t}\,k1}{k2}} & 0 & 0 \\ \frac{e^{-k1\,t}\,k1}{\left(e^{-k1\,t-k2\,t}-\frac{e^{-k1\,t-k2\,t}\,k1}{k2}\right)k2} & \frac{-e^{-k1\,t}+\frac{e^{-k1\,t}\,k1}{k2}}{e^{-k1\,t-k2\,t}-\frac{e^{-k1\,t-k2\,t}\,k1}{k2}} & 0 \end{pmatrix}$$

- **Step 5.** Calculate the solution to the initial-value problem.

```
In[ ]:= IC = {A0, 0, 0};
```

```
In[ ]:= sol = Dot[phi[t], Inverse[phi[0]], IC] // Simplify // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} A0\,e^{-k1\,t} \\ -\frac{A0\left(e^{-k1\,t}-e^{-k2\,t}\right)k1}{k1-k2} \\ \frac{A0\left(k1-e^{-k2\,t}\,k1+\left(-1+e^{-k1\,t}\right)k2\right)}{k1-k2} \end{pmatrix}$$

- **Step 6.** Verify the solution using **DSolve** function.

```
In[ ]:= ClearAll["Global`*"]
      dsol = FullSimplify[DSolve[{y1'[t] == -k1 * y1[t], y2'[t] == k1 * y1[t] - k2 * y2[t],
          y3'[t] == k2 * y2[t], y1[0] == A0, y2[0] == 0, y3[0] == 0}, {y1[t], y2[t], y3[t]}, t]]
```

Out[ ]=

$$\left\{\left\{y1[t] \to A0\,e^{-k1\,t},\ y2[t] \to -\frac{A0\left(e^{-k1\,t}-e^{-k2\,t}\right)k1}{k1-k2},\ y3[t] \to \frac{A0\left(k1-e^{-k2\,t}\,k1+\left(-1+e^{-k1\,t}\right)k2\right)}{k1-k2}\right\}\right\}$$

Summary

After completing this chapter, you should be able to

- improve problem-solving skills by practicing different methods.
- develop SOPs and streamline your workflow after you are familar with the methods.
- always check/verify your solutions for quality assurance.

- Remember, “to learn and not to do is really not to learn. To know and not to do is really not to know.” - Stephen R. Covey.

References and Suggested Readings

Table of Contents

Mathematica-Related Books

- An Elementary Introduction to the Wolfram Language, 2nd Ed., by Stephen Wolfram, Wolfram Media, Inc., 2017. URL: https://www.wolfram.com/language/elementary-introduction/2nd-ed/index.html

- The Student's Introduction to Mathematica and the Wolfram Language, 3rd Ed., by Bruce F. Torrence and Eve A. Torrence, Cambridge University Press, 2019. URL: https://doi.org/10.1017/9781108290937

- Hands-on Start to Wolfram Mathematica and Programming with the Wolfram Language, 2nd Ed., by Cliff Hastings, Kelvin Mischo, Michael Morrison, Wolfram Media, Inc., 2016

- Differential Equations with Mathematica, 5th Ed., by Martha L. Abell and James P. Braselton, Academic Press, 2022. URL: https://doi.org/10.1016/C2020-0-00005-8

- Advanced Engineering Mathematics with Mathematica, by Edward B. Magrab, CRC Press, 2020

- Wolfram Documentation Center: Symbolic Differential Equation Solving. URL: https://reference.wolfram.com/language/tutorial/DSolveOverview.html

- Wolfram Documentation Center: Advanced Numerical Differential Equation Solving in the Wolfram Language. URL: https://reference.wolfram.com/language/tutorial/NDSolveOverview.html

- Wolfram Documentation Center: Linear Algebra. URL: https://reference.wolfram.com/language/tutorial/LinearAlgebra.html

- Wolfram Documentation Center: Lists. URL: https://reference.wolfram.com/language/tutorial/Lists.html

Wolfram U Interactive Courses

- Wolfram U - FULL INTERACTIVE COURSE: An Elementary Introduction to the Wolfram Language. URL: https://www.wolfram.com/wolfram-u/an-elementary-introduction-to-the-wolfram-language/
- Wolfram U - FULL INTERACTIVE COURSE: Introduction to Notebooks. URL: https://www.wolfram.com/wolfram-u/introduction-to-notebooks/
- Wolfram U - FULL INTERACTIVE COURSE: Introduction to Calculus. URL: https://www.wolfram.com/wolfram-u/introduction-to-calculus/
- **Wolfram U - FULL INTERACTIVE COURSE: Introduction to Differential Equations**. URL: https://www.wolfram.com/wolfram-u/introduction-to-differential-equations/
- **Wolfram U - FULL INTERACTIVE COURSE: Introduction to Linear Algebra**. URL: https://www.wolfram.com/wolfram-u/introduction-to-linear-algebra/

Books on Engineering Mathematics (ODE & Linear Algebra)

- Kreyszig, Erwin, Advanced engineering mathematics, 10th Ed., International ed., John Wiley & Sons, Inc, 2011
- Holt, Jeffrey, Linear algebra with applications, 2nd Ed., W.H. Freeman and Company, 2017
- Pauls Online Math Notes - Differential Equations, by Paul Dawkins, 2018. URL: https://tutorial.math.lamar.edu/Classes/DE/DE.aspx
- The LibreTexts libraries - Differential Equations. URL: https://math.libretexts.org/Bookshelves/Differential_Equations
- The LibreTexts libraries - Linear Algebra. URL: https://math.libretexts.org/Bookshelves/Linear_Algebra

- Interactive Linear Algebra, by Dan Margalit and Joseph Rabinoff, Georgia Institute of Technology, 2019. URL: https://textbooks.math.gatech.edu/ila/
- MIT OpenCourseWare - Linear Algebra (Instructor: Prof. Gilbert Strang), 2010. URL: https://ocw.mit.edu/courses/18-06-linear-algebra-spring-2010/
- MIT OpenCourseWare - Differential Equations and Linear Algebra (Instructor: Prof. Gilbert Strang and Dr. Cleve Moler), 2015. URL: https://ocw.mit.edu/courses/res-18-009-learn-differential-equations-up-close-with-gilbert-strang-and-cleve-moler-fall-2015/
- Differential Equations and Linear Algebra, by Gilbert Strang, Wellesley-Cambridge Press, 2014. URL: https://math.mit.edu/~gs/dela/