(Custom CSS files are not reliable for controlling Jupyter font style. To establish the same appearance as the original notebook, depend on the browser to control the font, by setting the desired font faces in the browser settings. For example, Chrome 135 or Firefox 134 can do this. In this notebook series, Bookerly font is for markdown and Monaco is for code.)

Chapter 15 Matrices: Matrix functions and operations.

15.1 Show that $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$ for

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

Show that $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$ for the two matrices in the description.

The Wolfram Alpha entry lines go like:
!| {{5, 6}, {7, 8}} + {{1, 2}, {3, 4}} |!
and
!| {{1, 2}, {3, 4}} + {{5, 6}, {7, 8}} |!

$$\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A} = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$$

15.2 Find $\mathbf{A} - \frac{1}{2}\mathbf{B}$ for

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

A subtraction example.

The Wolfram Alpha entry line goes like:
!| {{1, 2}, {3, 4}} - (1/2) * {{5, 6}, {7, 8}} |!

$$\begin{bmatrix} -\frac{3}{2} & -1 \\ -\frac{1}{2} & 0 \end{bmatrix}$$

15.3 Find $\mathbf{AB}$ and $\mathbf{BA}$ for

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

Commutation of multiplication trial.

The Wolfram Alpha entry line goes like:
!| {{1, 2}, {3, 4}} * {{5, 6}, {7, 8}} |!
and
!| {{5, 6}, {7, 8}} * {{1, 2}, {3, 4}} |!

$$\mathbf{A} \times \mathbf{B} = \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix} \qquad \mathbf{B} \times \mathbf{A} = \begin{bmatrix} 23 & 24 \\ 31 & 46 \end{bmatrix}$$

15.4 Find $(2\mathbf{A} - \mathbf{B})^2$ for

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$$

Polynomial squared trial.

For some reason, the Wolfram Alpha entry line needs to be split up into two parts:
!| (2 * {{1, 2}, {3, 4}} - {{5, 6}, {7, 8}}) |!

to get the intermediate result of {{-3, -2}, {-1, 0}}, then

!| {{-3, -2}, {-1, 0}} ^2 |! to get the final answer, which is shown.

$$\mathbf{A} \times \mathbf{B} = \begin{bmatrix} 11 & 6 \\ 3 & 2 \end{bmatrix}$$

15.5 Find $\mathbf{AB}$ and $\mathbf{BA}$ for

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 7 & 0 \\ 8 & -1 \end{bmatrix}$$

The operation of multiplication is not defined for 3-column matrix times a 2-row matrix. However, looking at $\mathbf{BA}$, below is the outcome.

The entry into Wolfram Alpha goes like:
{{7, 0}, {8, -1}} * {{1, 2, 3}, {4, 5, 6}}

$$\mathbf{B} \times \mathbf{A} = \begin{bmatrix} 7 & 14 & 21 \\ 4 & 11 & 18 \end{bmatrix}$$

15.6 Find $\mathbf{AB}$ and $\mathbf{AC}$ if

$$\mathbf{A} = \begin{bmatrix} 4 & 2 & 0 \\ 2 & 1 & 0 \\ -2 & -1 & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 2 & 3 & 1 \\ 2 & -2 & -2 \\ -1 & 2 & 1 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 3 & 1 & -3 \\ 0 & 2 & 6 \\ -1 & 2 & 1 \end{bmatrix}$$

In [48]:

```
from scipy import linalg
import numpy as np

a = np.array([[4, 2, 0], [2, 1, 0], [-2, -1, 1]])
b = np.array([[2, 3, 1], [2, -2, -2], [-1, 2, 1]])
c = np.array([[3, 1, -3], [0, 2, 6], [-1, 2, 1]])

#of the 3 ways to multiply arrays in python,
#"matmul" gives the "matrix product"
#"multipy" returns element-wise multiplication
#"dot" returns dot product


res1 = np.matmul(a, b)
res2 = np.matmul(a, c)


#for line in res:
    #print ('  '.join(map(str, line)))
print(res1)
print()
print(res2)
```

```
[[12  8  0]
 [ 6  4  0]
 [-7 -2  1]]

[[12  8  0]
 [ 6  4  0]
 [-7 -2  1]]
```

15.7 Find $\mathbf{Ax}$ if

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} 9 \\ -1 \\ -2 \\ 0 \end{bmatrix}$$

In [51]:

```
from scipy import linalg
import numpy as np

a = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
x = np.array([[9], [-1], [-2], [0]])


#of the 3 ways to multiply arrays in python,
#"matmul" gives the "matrix product"
#"multipy" returns element-wise multiplication
#"dot" returns dot product


res1 = np.matmul(a, x)


#for line in res:
    #print ('  '.join(map(str, line)))
print(res1)
print()
```

```
[[ 1]
 [25]]
```

15.8 Find $\frac{d\mathbf{A}}{dt}$ if

$$\mathbf{A} = \begin{bmatrix} t^2+1 & e^{2t} \\ \sin t & 45 \end{bmatrix}$$

The derivative of the matrix can be found with Wolfram Alpha. Using the entry

!| d[{{t^2 + 1, e^(2 * t)}, {sin(t), 45}}]/dt |!

the result is returned:

$$\mathbf{A} = \begin{bmatrix} 2t & 2e^{2t} \\ \cos t & 0 \end{bmatrix}$$

Also offered are expressions for the eigenvalues and eigenvectors, for which there is no present need.

15.9 Find $\frac{d\mathbf{x}}{dt}$ if

$$\mathbf{x} = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix}$$

The derivative of the matrix can be found with Wolfram Alpha. Using the entry

!| differentiate {{x1(t)}, {x2(t)}, {x3(t)}} with respect to t |!

the result is returned:

$$\text{d/dt } \{\{\text{x1(t)}\}, \{\text{x2(t)}\}, \{\text{x3(t)}\}\} = \begin{bmatrix} x1'(t) \\ x2'(t) \\ x3'(t) \end{bmatrix}$$

Here W|A is prone to ignore the underbar or subscript form, but will cooperate if the condensed form shown above is used.

15.10 Find $\int \mathbf{A}\, dt$ for $\mathbf{A}$ as given in Problem 15.8.

Wolfram Alpha flatly refuses to integrate over the matrix. Luckily it is an easy matter to integrate component-wise by hand. The matrix that emerges is:

$$\int \mathbf{A}dt = \begin{bmatrix} t^3/3 + t + c_1 & (e^{2t})/2 + c_2 \\ \text{-cos t} + c_3 & 45t + c_4 \end{bmatrix}$$

15.11 Find $\int_0^1 \mathbf{x}\, dt$ if

$$\mathbf{x} = \begin{bmatrix} 1 \\ e^t \\ 0 \end{bmatrix}$$

Again W|A refuses to do the integration. But again it is an easy process and results in:

$$\int \mathbf{x}dt = \begin{bmatrix} t + c_1 \\ e^t + c_2 \\ c_3 \end{bmatrix}$$

The above needs to be evaluated at the upper bound of 1 and at the lower bound of 0, with the answer being the difference between the two. That turns out to be:

$$\textbf{evaluated} = \begin{bmatrix} 1 \\ e - 1 \\ 0 \end{bmatrix}$$

15.12 Find the eigenvalues of

$$\mathbf{A} = \begin{bmatrix} 1 & 3 \\ 4 & 2 \end{bmatrix}$$

In [7]:
```
import numpy as np


a = np.array([[1, 3], [4, 2]])
a = np.array([[2*t, 5*t], [-1*t, -2*t]])
w, v = np.linalg.eig(a)

print(w)
print(v)
```

```
[-2.  5.]
[[-0.70710678 -0.6       ]
 [ 0.70710678 -0.8       ]]
```

Note the two variables w and v assigned to the output of numpy.linalg.eig(). The first variable w is assigned an array of computed eigenvalues and the second variable v is assigned the matrix whose columns are the normalized eigenvectors corresponding to the eigenvalues in that order.

15.13 Find the eigenvalues of $\mathbf{A}t$ if

$$\mathbf{A} = \begin{bmatrix} 2 & 5 \\ -1 & -2 \end{bmatrix}$$

The eigenvalues can be found by Wolfram Alpha. If the entry is made:
!| eigenvalues {{2 * t, 5 * t}, {-1 * t, -2 * t}} |!

then the response is

Results: $\lambda = it$ and $\lambda = -it$
Corresponding eigenvectors: $v_1 = (-2 - i, 1)$ and $(-2 + i, 1)$

The t factor in the problem expression makes getting an answer more difficult when Python is used than when W|A is used.

15.14 Find the eigenvalues of $\mathbf{A}$ if

$$\mathbf{A} = \begin{bmatrix} 4 & 1 & 0 \\ -1 & 2 & 0 \\ 2 & 1 & -3 \end{bmatrix}$$

The eigenvalues can be found by Wolfram Alpha. If the entry is made:
!| eigenvalues {{4, 1, 0}, {-1, 2 , 0}, {2, 1, -3}} |!

then the response is

Results: $\lambda_1 = -3$ and $\lambda_2 = 3$
Corresponding eigenvectors: $v_1 = (0, 0, 1)$ and $v_2 = (6, -6, 1)$

15.15 Find the eigenvalues of $\mathbf{A}$ if

$$\mathbf{A} = \begin{bmatrix} 5 & 7 & 0 & 0 \\ -3 & -5 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

The eigenvalues can be found by Wolfram Alpha. If the entry is made:
!| eigenvalues {{5, 7, 0, 0}, {-3, 5, 0, 0}, {0, 0, 2, 1}, {0, 0, 0, 2}} |!

then the response is

Results: $\lambda_1 = -2$ and $\lambda_2 = 2$ and $\lambda_3 = 2$

Corresponding eigenvectors: $v_1 = (-1, 1, 0, 0)$ and $v_2 = (0, 0, 1, 0)$ and $v_3 = (-7, 3, 0, 0)$

In [ ]:

In [ ]: