EXERCISES

1. **Entering matrices:** Enter the following three matrices.

$$A = \begin{bmatrix} 2 & 6 \\ 3 & 9 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad C = \begin{bmatrix} -5 & 5 \\ 5 & 3 \end{bmatrix}.$$

2. **Check some linear algebra rules:**

   - **Is matrix addition commutative?** Compute `A+B` and then `B+A`. Are the results the same?
   - **Is matrix addition associative?** Compute `(A+B)+C` and then `A+(B+C)` in the order prescribed. Are the results the same?
   - **Is multiplication with a scalar distributive?** Compute α`(A+B)` and α`A+`α`B`, taking $\alpha = 5$ and show that the results are the same.
   - **Is multiplication with a matrix distributive?** Compute `A*(B+C)` and compare with `A*B+A*C`.
   - **Matrices are different from scalars!** For scalars, $ab = ac$ implies that $b = c$ if $a \neq 0$. Is that true for matrices? Check by computing `A*B` and `A*C` for the matrices given above.

     In general, matrix products do not commute either (unlike scalar products). Check if `A*B` and `B*A` give different results.

3. **Create matrices with `zeros`, `eye`, and `ones`:** Create the following matrices with the help of the matrix generation functions `zeros`, `eye`, and `ones`. See the on-line help on these functions, if required (e.g., `help eye`).

$$D = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad E = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix}, \quad F = \begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix}.$$

4. **Create a big matrix with submatrices:** The following matrix G is created by putting matrices A, B, and C, given above, on its diagonal. In how many ways can you create this matrix using submatrices A, B, and C (that is, you are not allowed to enter the non-zero numbers explicitly)?

$$G = \begin{bmatrix} 2 & 6 & 0 & 0 & 0 & 0 \\ 3 & 9 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 3 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & -5 & 5 \\ 0 & 0 & 0 & 0 & 5 & 3 \end{bmatrix}.$$

5. **Manipulate a matrix:** Do the following operations on matrix G created above in (4).

   - Delete the last row and last column of the matrix.
   - Extract the first 4×4 submatrix from G.
   - Replace $G(5,5)$ with 4.
   - What do you get if you type `G(13)` and hit return? Can you explain how MATLAB got that answer?
   - What happens if you type `G(12,1)=1` and hit return?

6. **See the structure of a matrix:** Create a 20×20 matrix with the command `A=ones(20)`. Now replace the 10×10 submatrix between rows 6:15 and columns 6:15 with zeros. See the structure of the matrix (in terms of nonzero entries with the command `spy(A)`. Set the 5×5 submatrices in the top right corner and bottom left corner to zeros and see the structure again.

7. **Create a symmetric matrix:** Create an upper triangular matrix with the following command:

```
A = diag(1:6) + diag(7:11,1) + diag(12:15,2).
```

   Make sure you understand how this command works (see the on-line help on `diag` if required). Now use the upper off-diagonal terms of A to make A a symmetric matrix with the following command:

```
A = A + triu(A,1)'.
```

   This is a somewhat *loaded* command. It takes the upper triangular part of A above the main diagonal, flips it (transposes), and adds to the original matrix A, thus creating a symmetric matrix A. See the on-line help on `triu`.

8. **Do some cool operations:** Create a 10×10 random matrix with the command `A=rand(10)`. Now do the following operations.

   - Multiply all elements by 100 and then round off all elements of the matrix to integers with the command `A=fix(A)`.
   - Replace all elements of A < 10 with zeros.
   - Replace all elements of A > 90 with infinity (`inf`).
   - Extract all $30 \leq a_{ij} \leq 50$ in a vector b, that is, find all elements of A that are between 30 and 50 and put them in a vector b.

9. **How about some fun with plotting?**

   - Plot the parametric curve $x(t) = t$, $y(t) = e^{-t/2} \sin t$ for $0 < t < \pi/2$ using `ezplot`.
   - Plot the cardioid $r(\theta) = 1 + \cos\theta$ for $0 < \theta < 2\pi$ using `ezpolar`.
   - Plot the contours of $x^2 + \sin(xy) + y^2 - z^2 = 0$ using `ezcontour` over the domain $-\pi/2 < x < \pi/2$, $-\pi/2 < y < \pi/2$.
   - Create a surface plot along with contours of the function $H(x, y) = \frac{x^2}{2} + (1 - \cos y)$ for $-\pi < x < \pi$, $-2 < y < 2$.

EXERCISES

1. **A script file to compute sine series:** Write a script file named sineseries.m that computes the value of $\sin(x)$ at a given x using n terms of the series expansion of the sine function:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots = \sum_{k=1}^{n} (-1)^{k-1} \frac{x^{2k-1}}{(2k-1)!}$$

   Follow the steps given below.

   - First, query MATLAB to see if the name *sineseries* is already taken by some variable or function with the command `exist('sineseries')`. What does the MATLAB response mean? [Hint, see on-line help on `exist`.]
   - Include the following line as the header (H1 line) of your script file.
     `%SINESERIES: computes sin(x) from series expansion`
     Now code the formula so that it computes the sum of the series for a given scalar x and a given integer n.
   - Save the file. Type `help sineseries` to see if MATLAB can access your script file. Now, compute $\sin(\pi/6)$ with $n = 1, 5, 10$, and 20. Compare the results. Do the same for some other x of your choice.

2. **A function file to compute sine series:** Take the script file written above and convert it into a function file using the following steps.

   - Name the function `sine_series` and modify the H1 line appropriately.
   - Let x and n be the input to your function and y (the sum) be the output.
   - Save the function and execute it to see that it works and gives the same output as the script file above.
   - Modify the function to include more on-line help on how to run the function.
   - Modify the function so that it can accept a vector x and give an appropriate y.
   - Modify the function to include a check on the input n. The function should proceed only if $n > 0$ is an integer, otherwise it should display an error message.
   - Provide for an optional output `err`, which gives the % error in y when compared to `sin(x)`. [Hint: use conditional statement on `nargout` for optional output.]
   - Modify the function so that it takes a default value of $n = 10$ if the user does not specify n. [Hint: use `nargin`.]
   - Execute the function to check all features you have added.

3. **A function as an input to another function:** There are several ways in which a function can be passed in the input argument list of another function. The function to be used in the input list can be written as an `inline` function (see Section 3.4.1 on page 70) or it can be coded in a function file. How the function is passed in the input list depends on how it is coded.

   Code the function $y(x) = \frac{\sin(x)}{x}$ as an inline function `sinc` and in a function file called sincfun.m. You will use this function in the input list of `ezplot` (see page 80 for `ezplot`) in various ways in the following instructions.

   - Use the inline function `sinc` in the input list of `ezplot` to plot the function over the default domain.

- Use the function `sincfun` as a string in the input list of `ezplot` to plot the function over the default domain.
- Create a function handle for `sincfun` and use the handle in the input list of `ezplot` to plot the function over the default domain.

4. **Write subfunctions:** In this exercise you will write a simple function to find $\sin(x)$ and/or $\cos(x)$ from series expansion using two subfunctions—`sine_series` and `cosine_series`.
   - Write a function named `cosine_series` to evaluate the cosine series (follow Exercise 2 above)

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots = \sum_{k=1}^{n} (-1)^{k-1} \frac{x^{2(k-1)}}{2(k-1)!}$$

     Execute and test the function to make sure it works correctly.
   - Write a new function named `trigseries` as follows.
     - The input list of the function should include x, n, and another string variable *options*. The user can specify `'sin'`, `'cos'`, or `'both'` for *options*. The default should be `'both'`. The output list should include y and *err*, as discussed in Exercise 2.
     - The function `trigseries` should call `sine_series` and `cosine_series` as appropriate to compute y and *err*, depending on what the user specifies in *options*. Implement this call sequence with `switch` using `options` as the switch.
     - Program the output y and *err* to be two column arrays if the user asks for `'both'` in *options* or as the default.
   - Copy and paste the functions `sine_series` and `cosine_series` as subfunctions below the function `trigseries` in the same file.
   - Delete the original files sine_series2.m and cosine_series2.m or rename them something else. Test `trigseries` with various input and *options*. Make sure it works correctly. Now execute `trigseries` with only one input `x=[0 pi/6 pi/4 pi/3 pi/2]`. Do you get reasonable answers?

5. **Profile a function:** Profile the function `trigseries`, developed above in Exercise 4, taking a vector x of 100 equally spaced points between 0 and π as the only input.

6. **Recursion:** Write a function to compute $n!$ using recursion. (Note that this is not the most efficient way to compute $n!$. However, it is conceptually a recursive calculation that is easy to implement and test recursion in MATLAB.)

EXERCISES

1. **Linear algebraic equations:** Find the solution of the following set of linear algebraic equations, as advised below.

$$\begin{aligned} x + 2y + 3z &= 1 \\ 3x + 3y + 4z &= 1 \\ 2x + 3y + 3z &= 2. \end{aligned}$$

   - Write the equation in matrix form and solve for $\mathbf{x} = [x\ y\ z]^T$ using the left division \.
   - Find the solution again using the function `rref` on the augmented matrix.
   - Can you use the LU decomposition to find the solution? [Hint: Since $[\mathbf{LU}]\mathbf{x} = \mathbf{b}$, let $[\mathbf{U}]\mathbf{x} = \mathbf{y}$, so that $[\mathbf{L}]\mathbf{y} = \mathbf{b}$. Now, first solve for $\mathbf{y}$ and then for $\mathbf{x}$.]

2. **Eigenvalues and eigenvectors:** Consider the following matrix.

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

   - Find the eigenvalues and eigenvectors of $\mathbf{A}$.
   - Show, by computation, that the eigenvalues of $\mathbf{A}^2$ are square of the eigenvalues of $\mathbf{A}$.
   - Compute the square of the eigenvalues of $\mathbf{A}^2$. You have now obtained the eigenvalues of $\mathbf{A}^4$. From these eigenvalues, can you guess the structure of $\mathbf{A}^4$?
   - Compute $\mathbf{A}^4$. Can you compute $\mathbf{A}^{-1}$ without using the `inv` function?

3. **Linear and quadratic curve fits:** The following data is given to you.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 & 0.15 & 0.2 & 0.30 & 0.4 & 0.52 & 0.6 & 0.70 & 0.8 & 0.90 & 1 \\ 0 & 1.61 & 2.2 & 3.45 & 4.8 & 6.55 & 7.8 & 9.45 & 11.2 & 13.05 & 15 \end{bmatrix}$$

   - Enter vectors x and y and plot the raw data using `plot(x,y,'o')`.
   - Click on the figure window. Select Basic Fitting from the Tools menu of the figure window. Once you get the Basic Fitting window, check the box for linear fit. In addition, check the boxes for Show equations, Plot residuals (select a *scatter* plot rather than a *bar* plot option), and Show norm of residuals.
   - Now, do a quadratic fit by checking the quadratic fit box and all other boxes as you did for the linear case.
   - Compare the two fits. Which fit is better? There isn't really a competition, is there?

4. **A nonlinear curve fit:** Enter the following experimental data in MATLAB workspace.

```
t = [0 1.40 2.79 4.19 5.58 6.98 8.38 9.77 11.17 12.57];
x = [0 1.49 0.399 -0.75 -0.42 0.32 0.32 -0.10 -0.21 0];
```

   You are told that this data comes from measuring the displacement x of a damped oscillator at time instants t. The response x is, therefore, expected to have the following form:

$$x = Ce^{-\lambda_1 t}\sin(\lambda_2 t)$$

   where the constants C, λ_1, and λ_2 are to be determined from the experimental data. Thus, your job is to fit a curve of the given form to the data and find the constants that give the best fit.