`peak` and `mountains`

1) The function `peak` takes as input three positive integers: m, n and h;
as output, it returns the "peak" matrix $P(m, n, h)$, as defined in the next slides.
Intuitively, m and n identify the shape of the peak in the middle of the matrix, while h identifies its height.

2) The function `peak_MV` takes as input the integers m, n and h (as for `peak`), and a vector v of "suitable" size. As output, `peak_MV` returns the vector y resulting from the multiplication of the peak matrix $P(m, n, h)$ by the input vector v:
$y = P(m, n, h) * v$.

3) The function `mountains` takes as input three integers: m, n and h; as output, it returns the "mountain" matrix $M(m, n, h)$, as defined in the next slides.
Intuitively, m and n identify the number (grid) of peaks in the matrix, while h identifies their height.

4) The function `mountains_MV` takes as input the integers m, n and h (as for `mountains`), and a vector w of "suitable" size. As output, `moutains_MV` returns the vector z resulting from the multiplication of the mountains matrix $M(m, n, h)$ by the input vector w: $z = M(m, n, h) * w$.

Definition of $P(m, n, h)$ (by examples)

```
             1 1
P(3,2,1) =   1 1
             1 1
```

```
P(2,3,1) =   1 1 1
             1 1 1
```

```
             1 1 1 1
             1 2 2 1
P(3,2,2) =   1 2 2 1
             1 2 2 1
             1 1 1 1
```

```
             1 1 1 1 1
P(2,3,2) =   1 2 2 2 1
             1 2 2 2 1
             1 1 1 1 1
```

```
             1 1 1 1 1 1
             1 2 2 2 2 1
P(2,2,3) =   1 2 3 3 2 1
             1 2 3 3 2 1
             1 2 2 2 2 1
             1 1 1 1 1 1
```

```
             1 1 1 1 1 1 1 1 1
             1 2 2 2 2 2 2 2 1
             1 2 3 3 3 3 3 2 1
P(1,3,4) =   1 2 3 4 4 4 3 2 1
             1 2 3 3 3 3 3 2 1
             1 2 2 2 2 2 2 2 1
             1 1 1 1 1 1 1 1 1
```

Definition of $M(m, n, h)$ (by examples)

```
M(3,2,1) =  1 1
            1 1
            1 1
```

```
M(2,3,1) =  1 1 1
            1 1 1
```

```
M(3,2,2) =  1 1 1 1 1 1
            1 2 1 1 2 1
            1 1 1 1 1 1
            1 1 1 1 1 1
            1 2 1 1 2 1
            1 1 1 1 1 1
            1 1 1 1 1 1
            1 2 1 1 2 1
            1 1 1 1 1 1
```

```
M(2,3,2) =  1 1 1 1 1 1 1 1 1
            1 2 1 1 2 1 1 2 1
            1 1 1 1 1 1 1 1 1
            1 1 1 1 1 1 1 1 1
            1 2 1 1 2 1 1 2 1
            1 1 1 1 1 1 1 1 1
```

```
M(1,2,3) =  1 1 1 1 1 1 1 1 1 1
            1 2 2 2 1 1 2 2 2 1
            1 2 3 2 1 1 2 3 2 1
            1 2 2 2 1 1 2 2 2 1
            1 1 1 1 1 1 1 1 1 1
```

```
M(3,1,3) =  1 1 1 1 1
            1 2 2 2 1
            1 2 3 2 1
            1 2 2 2 1
            1 1 1 1 1
            1 1 1 1 1
            1 2 2 2 1
            1 2 3 2 1
            1 2 2 2 1
            1 1 1 1 1
            1 1 1 1 1
            1 2 2 2 1
            1 2 3 2 1
            1 2 2 2 1
            1 1 1 1 1
```

```
M(3,2,4) =  1 1 1 1 1 1 1 1 1 1 1 1 1 1
            1 2 2 2 2 2 1 1 2 2 2 2 2 1
            1 2 3 3 3 2 1 1 2 3 3 3 2 1
            1 2 3 4 3 2 1 1 2 3 4 3 2 1
            1 2 3 3 3 2 1 1 2 3 3 3 2 1
            1 2 2 2 2 2 1 1 2 2 2 2 2 1
            1 1 1 1 1 1 1 1 1 1 1 1 1 1
            1 1 1 1 1 1 1 1 1 1 1 1 1 1
            1 2 2 2 2 2 1 1 2 2 2 2 2 1
            1 2 3 3 3 2 1 1 2 3 3 3 2 1
            1 2 3 4 3 2 1 1 2 3 4 3 2 1
            1 2 3 3 3 2 1 1 2 3 3 3 2 1
            1 2 2 2 2 2 1 1 2 2 2 2 2 1
            1 1 1 1 1 1 1 1 1 1 1 1 1 1
            1 1 1 1 1 1 1 1 1 1 1 1 1 1
            1 2 2 2 2 2 1 1 2 2 2 2 2 1
            1 2 3 3 3 2 1 1 2 3 3 3 2 1
            1 2 3 4 3 2 1 1 2 3 4 3 2 1
            1 2 3 3 3 2 1 1 2 3 3 3 2 1
            1 2 2 2 2 2 1 1 2 2 2 2 2 1
            1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Assignment

1) Write the Matlab functions `peak` and `mountains`.
   Goal: Style, readability.   Let the code speak!

2) Write the Matlab functions `peak_MV` and `mountains_MV`.
   Goal: Do NOT form the `peak` and `mountains` matrices.
   Assume that the input vectors are of the correct size.

Definitions

$$F_m = P_{m,\frac{m}{k}} \left(I_k \otimes F_{\frac{m}{k}}\right) D_{k,\frac{m}{k}} \left(F_k \otimes I_{\frac{m}{k}}\right)$$

Identity matrix

$$I_k = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & & \\ \vdots & & \ddots & \\ 0 & & & 1 \end{bmatrix}_{[k \times k]}$$

Definitions

$$F_m = P_{m,\frac{m}{k}} \left(I_k \otimes F_{\frac{m}{k}}\right) D_{k,\frac{m}{k}} \left(F_k \otimes I_{\frac{m}{k}}\right)$$

Fourier matrix

$$F_m = e^{\frac{-2\pi i}{m} kj}, \quad k, j \in [0, 1, \ldots, m-1]$$

Definitions

$$F_m = P_{m,\frac{m}{k}} \left(I_k \otimes F_{\frac{m}{k}}\right) D_{k,\frac{m}{k}} \left(F_k \otimes I_{\frac{m}{k}}\right)$$

Kronecker Product

$$A = \begin{bmatrix} a_{00} & a_{01} & \dots & \\ a_{10} & a_{11} & & \\ \vdots & & \ddots & \\ & & & \end{bmatrix}_{[m \times n]} , \quad A \otimes B = \begin{bmatrix} a_{00}B & a_{01}B & \dots & \\ a_{10}B & a_{11}B & & \\ \vdots & & \ddots & \\ & & & \end{bmatrix}_{[? \times ?]}$$

Definitions

$$F_m = P_{m,\frac{m}{k}} \left(I_k \otimes F_{\frac{m}{k}}\right) D_{k,\frac{m}{k}} \left(F_k \otimes I_{\frac{m}{k}}\right)$$

Twiddle Factors

$$D_{k,\frac{m}{k}} = \begin{bmatrix} (D^0 = I) & 0 & & \cdots & 0 \\ 0 & D & & & \\ & & D^2 & & \\ \vdots & & & \ddots & \\ 0 & & & & D^{k-1} \end{bmatrix}_{[m \times m]}$$

$$D = \mathtt{diag}(e^{\frac{-2\pi i}{m} j}), \quad j \in [0, \ldots, \frac{m}{k} - 1]$$

Definitions

$$F_m = P_{m,\frac{m}{k}} \left(I_k \otimes F_{\frac{m}{k}}\right) D_{k,\frac{m}{k}} \left(F_k \otimes I_{\frac{m}{k}}\right)$$

Permutation

$$P_{m,\frac{m}{k}} = \text{stride}$$

$$P_{m,\frac{m}{2}} \begin{bmatrix} \times & \times & \times & \times & \dots & \times & \times \\ \circ & \circ & \circ & \circ & \dots & \circ & \circ \\ + & + & + & + & \dots & & \\ \vdots & \vdots & & & & & \\ \hline \hat{\times} & \hat{\times} & \hat{\times} & \dots & & & \\ \hat{\circ} & \hat{\circ} & \hat{\circ} & \dots & & & \\ \hat{+} & \hat{+} & \hat{+} & \dots & & & \\ \vdots & \vdots & & & & & \end{bmatrix}_{[m\times m]} \rightarrow \begin{bmatrix} \times & \times & \times & \times & \dots & \times & \times \\ \hat{\times} & \hat{\times} & \hat{\times} & \hat{\times} & \dots & \hat{\times} & \hat{\times} \\ \hline \circ & \circ & \circ & \circ & \dots & \circ & \circ \\ \hat{\circ} & \hat{\circ} & \hat{\circ} & \hat{\circ} & \dots & \hat{\circ} & \hat{\circ} \\ \hline + & + & + & + & \dots & & \\ \hat{+} & \hat{+} & \hat{+} & \hat{+} & \dots & & \\ \hline \vdots & \vdots & & & & & \end{bmatrix}_{[m\times m]}$$

Definitions

$$F_m = P_{m,\frac{m}{k}} \left(I_k \otimes F_{\frac{m}{k}}\right) D_{k,\frac{m}{k}} \left(F_k \otimes I_{\frac{m}{k}}\right)$$

Permutation

$$P_{12,3} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \hline \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \hline \alpha_6 \\ \alpha_7 \\ \alpha_8 \\ \hline \alpha_9 \\ \alpha_{10} \\ \alpha_{11} \end{bmatrix} \rightarrow \begin{bmatrix} \alpha_0 \\ \alpha_3 \\ \alpha_6 \\ \alpha_9 \\ \hline \alpha_1 \\ \alpha_4 \\ \alpha_7 \\ \alpha_{10} \\ \hline \alpha_2 \\ \alpha_5 \\ \alpha_8 \\ \alpha_{11} \end{bmatrix}$$

$$P_{12,2} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \hline \alpha_2 \\ \alpha_3 \\ \hline \alpha_4 \\ \alpha_5 \\ \hline \alpha_6 \\ \alpha_7 \\ \hline \alpha_8 \\ \alpha_9 \\ \hline \alpha_{10} \\ \alpha_{11} \end{bmatrix} \rightarrow \begin{bmatrix} \alpha_0 \\ \alpha_2 \\ \alpha_4 \\ \alpha_6 \\ \alpha_8 \\ \alpha_{10} \\ \hline \alpha_1 \\ \alpha_3 \\ \alpha_5 \\ \alpha_7 \\ \alpha_9 \\ \alpha_{11} \end{bmatrix}$$

Problem 1: 1d-FFT

$$y := Fv = P_{m,\frac{m}{k}}\,(I_k \otimes F_{\frac{m}{k}})\,D_{k,\frac{m}{k}}\,(F_k \otimes I_{\frac{m}{k}})v$$

- Write the Matlab function `my1dFFT`(v, p).
  The input arguments are a column vector v of size m and an integer $p > 1$.
  The output is the column vector y, containing v's FFT.
- The objective is **smart** and **clear** code!

Rules

- y must be computed by the recursive formula above, with $k = p$.
- If `mod`$(m, p) <> 0$, then pad v with zeros.
- $1 < p < \sqrt{m}$.
- No matrix of size $m \times m$ can be used.
- The command `kron` cannot be used.
- No need to further decompose F_k and $F_{\frac{m}{k}}$.
- If needed, write the functions for D, P and F yourself;
  include them in the same file as `my1dFFT`.

Problem 2: 2d-FFT

$$Y := P_{m,\frac{m}{2}}(I_2 \otimes F_{\frac{m}{2}})D_{2,\frac{m}{2}} \left(\begin{array}{c|c} I_{\frac{m}{2}} & I_{\frac{m}{2}} \\ \hline I_{\frac{m}{2}} & -I_{\frac{m}{2}} \end{array}\right) X \left(\begin{array}{c|c} I_{\frac{n}{2}} & I_{\frac{n}{2}} \\ \hline I_{\frac{n}{2}} & -I_{\frac{n}{2}} \end{array}\right) D_{2,\frac{n}{2}}(I_2 \otimes F_{\frac{n}{2}})P^T_{n,\frac{n}{2}}$$

- Write the Matlab function `my2dFFT`(X).
  The input argument is a matrix X of size $m \times n$.
  The output is the matrix Y, containing X's 2d-FFT.
- The objective is **smart** and **clear** code!

Rules

- Y must be computed by the recursive formula above.
- The same rules for `my1dFFT` apply. The only "full" matrix allowed is Y.
- Hint: Work your way from X towards the boundaries, until you reach

$$P_{m,\frac{m}{2}}(I_2 \otimes F_{m/2})\tilde{X}(I_2 \otimes F_{n/2})P^T_{n,\frac{n}{2}}.$$

  Then use math, NOT brute force.