Matlab: Matrix Manipulation

Hendrik Speleers

Matlab: Matrix Manipulation

Università di Roma Tor Vergata

- Overview
  - Code vectorization
    - Matrix indexing
    - Matrix creation and rearrangements
    - Mathematical operations
  - Input and output
    - Console input/output
    - File input/output: binary and text files

Matlab: Matrix Manipulation

Università di Roma Tor Vergata

- Matrix manipulation
  - Matlab = Matrix laboratory
  - Matrices and vectors are the fundamental building blocks
  - Code vectorization: Avoid loops
    - Appearance: vectorized code appears more like mathematical expressions, making the code often easier to understand
    - Often less error prone: without loops, vectorized code is mostly shorter; fewer lines of code mean fewer opportunities to introduce programming errors
    - Performance: vectorized code often runs faster than the corresponding code containing loops

Matlab: Matrix Manipulation

Università di Roma Tor Vergata

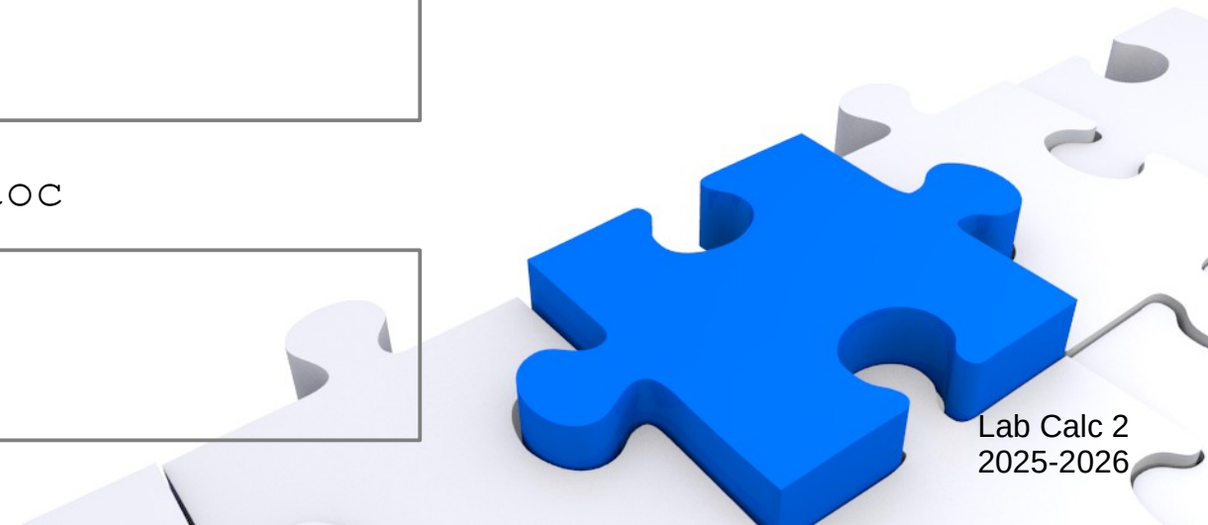
- Matrix manipulation
  - Example of code vectorization

```
function y1 = fun1()

i = 0;
for x = 0:.01:100000
    i = i + 1;
    y1(i) = sin(x);
end
```

```
function y2 = fun2()

x = 0:.01:100000;
y2 = sin(x);
```

- Time your code with the keywords `tic` and `toc`

```
>> tic; y1 = fun1(); toc;
>> tic; y2 = fun2(); toc;
```



Matlab: Matrix Manipulation

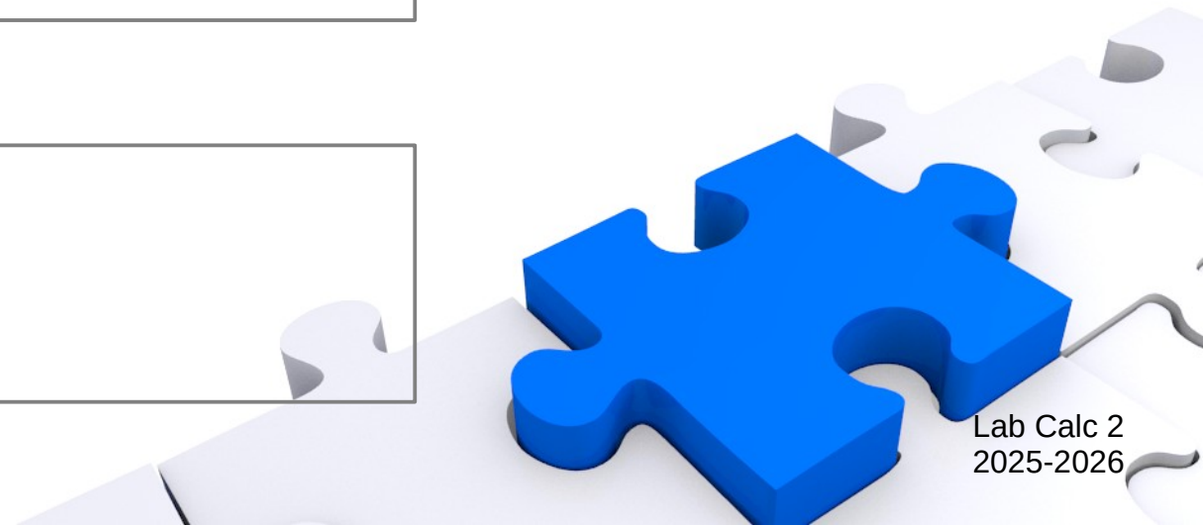
Università di Roma Tor Vergata

- Matrix indexing
  - Element selection
    - Vectors

```
>> v = [1, 3, 5, 4];
>> a = v(2);
>> v(end) = 6;
```

    - Matrices

```
>> M = [1, 3; 5, 4];
>> a = M(1, 2);
>> b = M(3);
```



Matlab: Matrix Manipulation

Università di Roma Tor Vergata

- Matrix indexing
  - Element block selection
    - Vectors

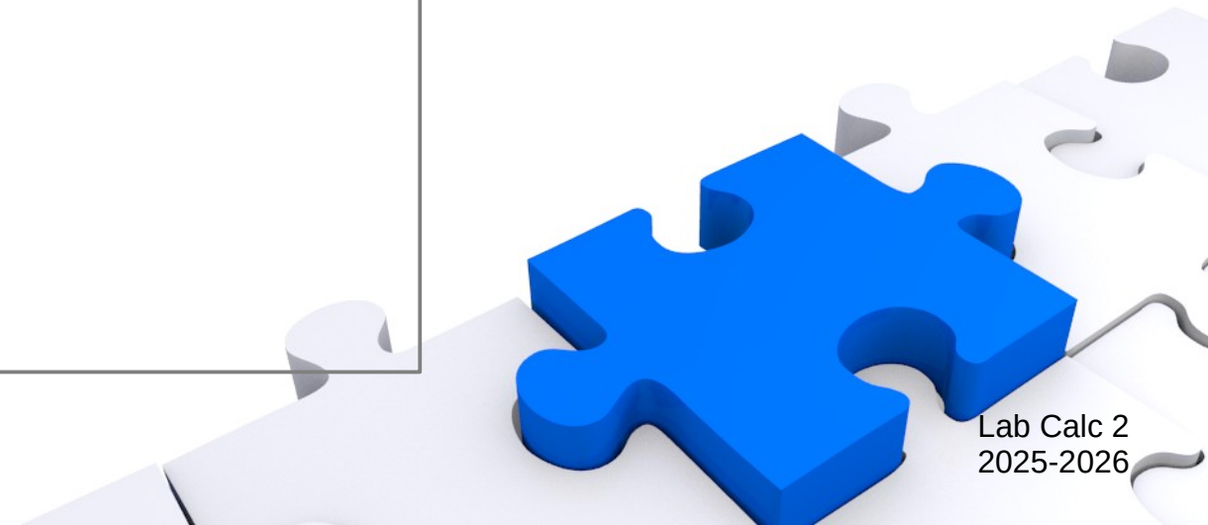
```
>> v = [1, 3, 5, 4];
>> a = v(1:3);
>> b = v(1:2:end);
>> c = v(end:-1:1);
>> d = v([1, 4]);
>> v([1, 2]) = [8, 9];
>> v([3, 4]) = 5;
```

Matlab: Matrix Manipulation

Università di Roma Tor Vergata

- Matrix indexing
  - Element block selection
    - Matrices

```
>> M = [1, 3, 2; 5, 4, 6; 8, 7, 9];
>> a = M(2:3, 1:2);
>> b = M(3, :);
>> c = M([1, 3], [2, 2]);
>> M(end, :) = [-1, -2, -3];
>> M([2, 3], [2, 3]) = 0;
>> d = M([1, 2, 5]);
>> M([2, 3]) = [-9, -8];
```



Matlab: Matrix Manipulation

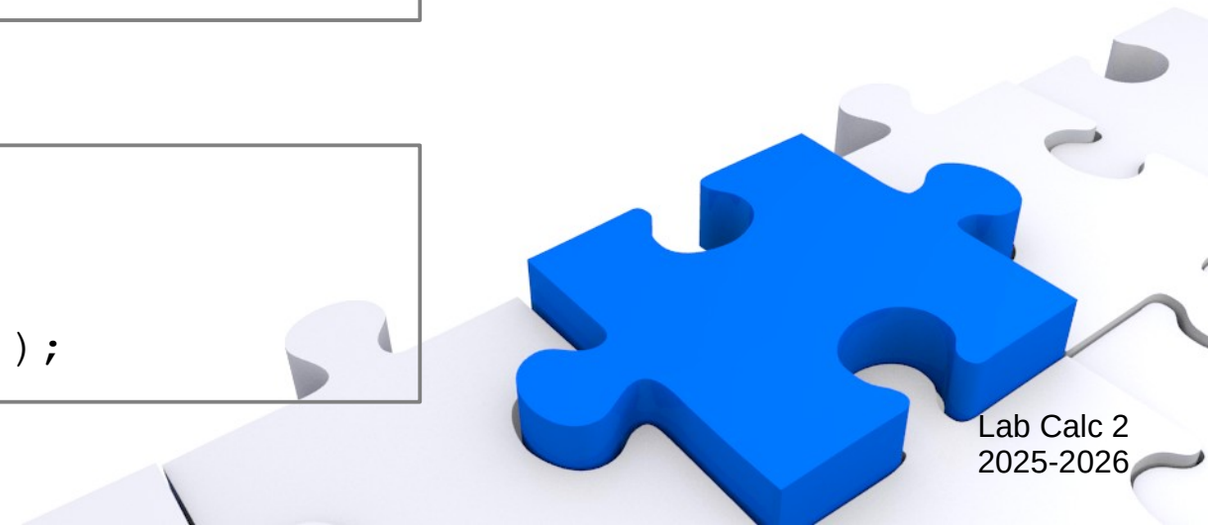
Università di Roma Tor Vergata

- Matrix indexing
  - Boolean selection
    - Vectors

```
>> v = [1, 3, 5, 4];
>> bidx = [true, false, true, false]
>> v(bidx) = 0;
```

    - Matrices

```
>> M = [1, 3; 5, 4];
>> a = M(M > 3);
>> b = M([true, false], [false, true]);
```

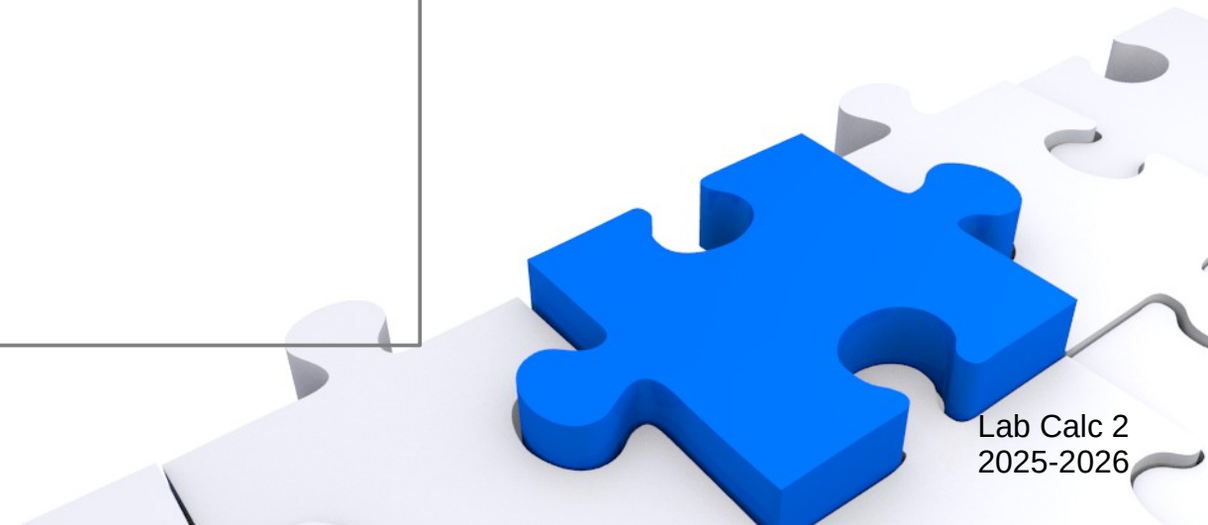


Matlab: Matrix Manipulation

Università di Roma Tor Vergata

- Matrix indexing
  - Example: shift the columns of a matrix by k positions

```
>> A = [1, 2, 3, 4, 5; 6, 7, 8, 9, 10]
A =
     1     2     3     4     5
     6     7     8     9    10
>> [m, n] = size(A);
>> k = 2;
>> B = A(:, [n-k+1:n, 1:n-k])
B =
     4     5     1     2     3
     9    10     6     7     8
```



Matlab: Matrix Manipulation

Università di Roma Tor Vergata

- Matrix creation
  - Common matrices
    - Zero/one matrix: `zeros(m, n)`, `ones(m, n)`
    - Logical matrix: `false(m, n)`, `true(m, n)`
    - Identity matrix: `eye(m, n)`
    - Random matrix: `rand(m, n)`, `randi(imax, m, n)`
  - Diagonal matrices
    - `diag(v, k)`
    - `v` is matrix: returns `k`-th diagonal of `v` in vector
    - `v` is vector: returns matrix with `v` on `k`-th diagonal
  - Many others: `magic(n)`, `pascal(n)`, `...`

Matlab: Matrix Manipulation

Università di Roma Tor Vergata

- Matrix rearrangements
  - Change shape
    - Flatting: `M(:)`
    - Reshaping: `reshape(M, m, n)`
  - Reordering
    - Transpose: `M'`
    - Flipping: `flipud(M), fliplr(M), flip(v)`
    - Shifting: `circshift(M, [m, n])`
  - Repetition
    - Matrix copies: `repmat(M, m, n)`
    - Element copies: `repelem(M, m, n)`

Matlab: Matrix Manipulation

Università di Roma Tor Vergata

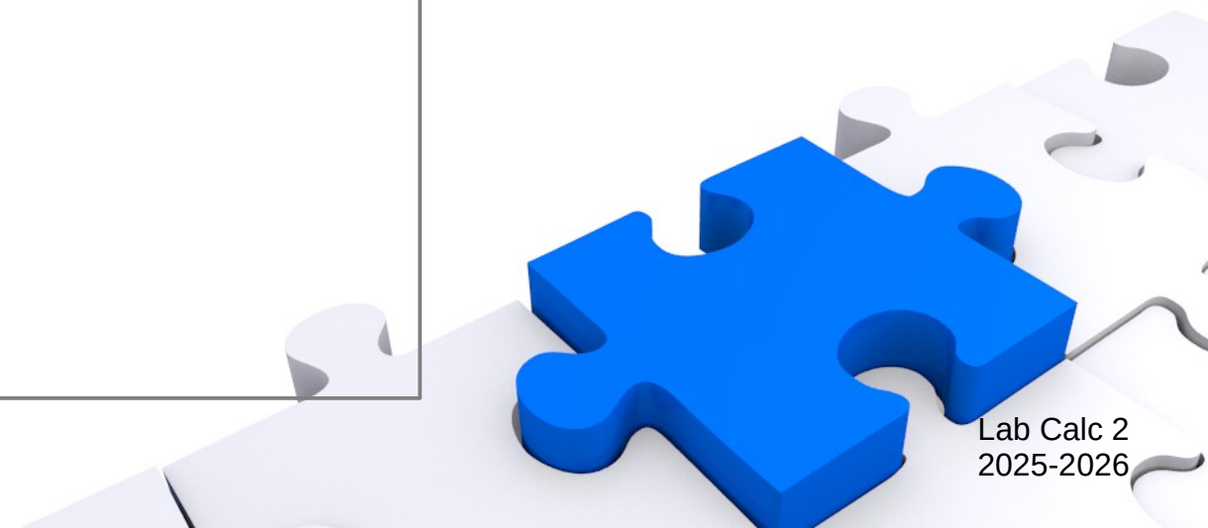
- Mathematical operations
  - Matrix operations
    - Addition: `M1 + M2` (elementwise)
    - Substraction: `M1 – M2` (elementwise)
    - Multiplication: `M1 * M2` (row by column)
    - Power: `M^p` (square matrix)
  - Other elementwise operations
    - Multiplication: `M1 .* M2`
    - Division: `M1 ./ M2`
    - Power: `M.^p, M1.^M2`

Matlab: Matrix Manipulation

Università di Roma Tor Vergata

- Mathematical operations
  - Broadcasting (since R2016b)
    - Implicit expansion for elementwise operations, so that matrices have same size
    - Size of matrices must be compatible: be the same or one of them is 1

```
>> M1 = [1, 2, 3; 1, 2, 3];
>> M2 = [5, 5, 5; 2, 2, 2];
>> M = M1 + M2;
>> M = [1, 2, 3] + [5; 2];
>> M
M =
      6      7      8
      3      4      5
```



Matlab: Matrix Manipulation

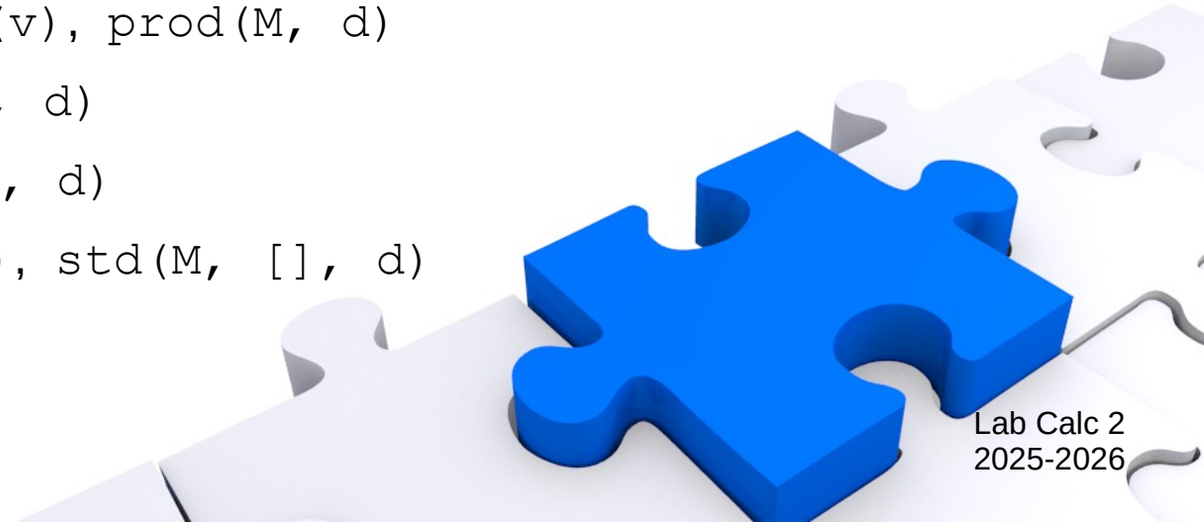
Università di Roma Tor Vergata

- Mathematical operations
  - Logical operations
    - Scalars: `s1 && s2` (and), `s1 || s2` (or), `~s` (not)
    - Elementwise: `M1 & M2` (and), `M1 | M2` (or), `~M` (not)
    - Vectorwise: `any(v)`, `any(M, d)`, `all(v)`, `all(M, d)`
  - Comparison operations
    - Elementwise: `M1 == M2` (eq), `M1 ~= M2` (ne),
      `M1 < M2` (lt), `M1 <= M2` (le),
      `M1 > M2` (gt), `M1 >= M2` (ge)
    - Vectorwise: `isequal(M1, M2)`

Matlab: Matrix Manipulation

Università di Roma Tor Vergata

- Mathematical operations
  - Mathematical functions
    - Power functions: `exp(M), log(M), log10(M), sqrt(M)`
    - Trig functions: `cos(M), sin(M), tan(M), cosh(M), sinh(M), tanh(M)`
    - Complex functions: `abs(M), conj(M), real(M), imag(M)`
    - Rounding functions: `ceil(M), floor(M), fix(M), round(M)`
    - Sum/product: `sum(v), sum(M, d), prod(v), prod(M, d)`
    - Min: `min(v), min(M1, M2), min(M, [], d)`
    - Max: `max(v), max(M1, M2), max(M, [], d)`
    - Statistics: `mean(v), mean(M, d), std(v), std(M, [], d)`
    - `...`

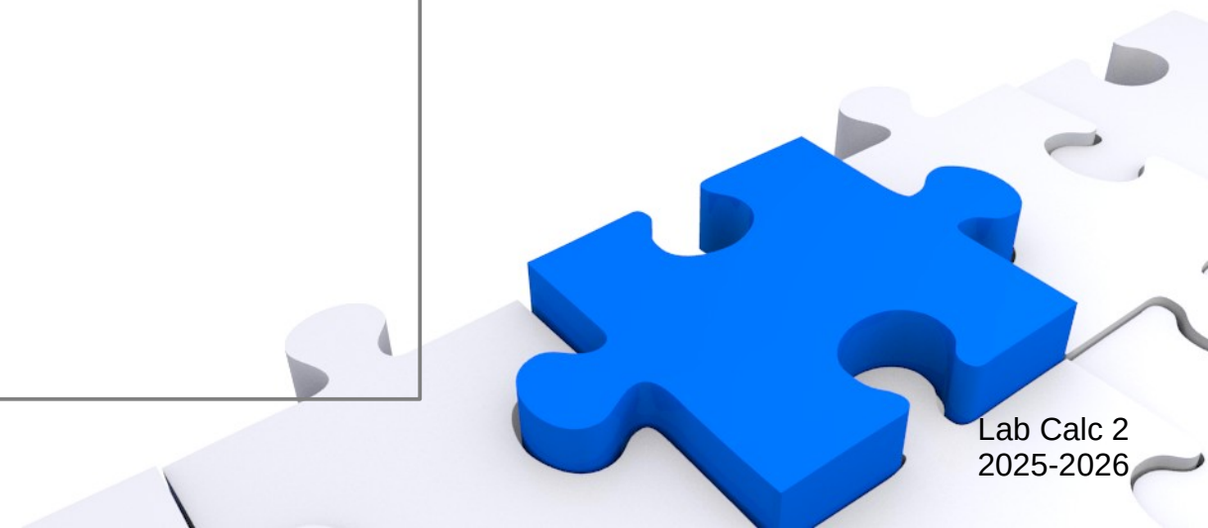


Matlab: Matrix Manipulation

Università di Roma Tor Vergata

- Mathematical operations
  - Example
    - Compute the max value of a matrix
    - Replace all negative entries by zero and the others by the max value

```
>> A = [-2, 3, 4; 1, -3, 2];
>> maxA = max(A(:));
>> A(A<0) = 0;
>> A(A>0) = maxA;
>> A
A =
     0     4     4
     4     0     4
```



Matlab: Matrix Manipulation

Università di Roma Tor Vergata

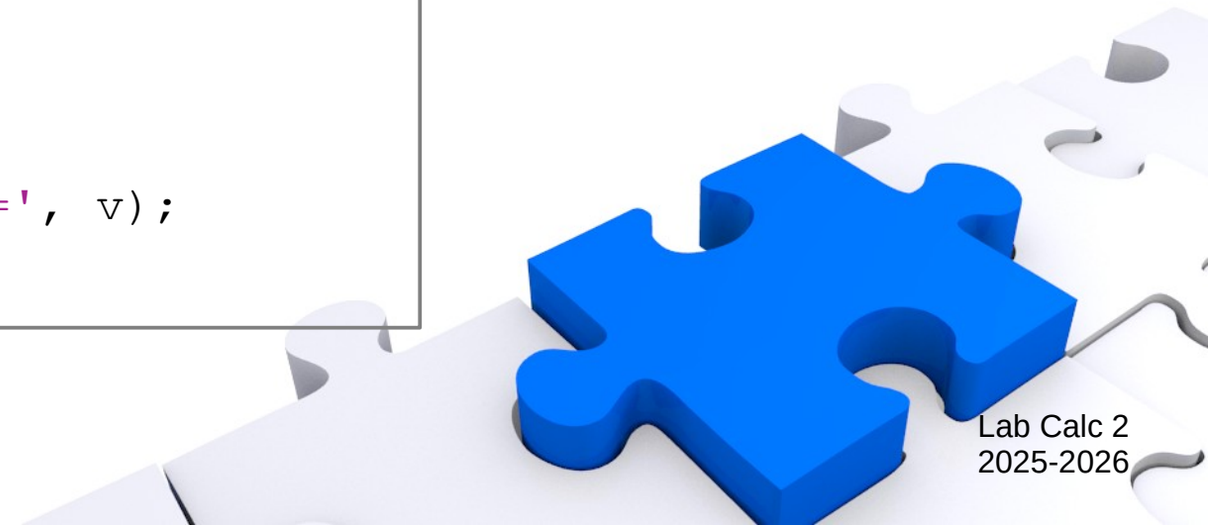
- Input and output
  - Console input
    - Keyboard: `v = input('text'), s = input('text', 's')`
    - Menu: `i = menu('title', 'item1', 'item2', __)`
  - Console output
    - Plain display: `disp(v)`
      - Format of numbers is controlled by `format`
      - Convertion of numbers to string by `num2str`
    - Formatted display: `fprintf('format', a, b, __)`
      - Conversion placeholders: `%d`, `%f`, `%s`, `...`
      - Escape characters: `%%`, `\n`, `\t`, `\\`, `...`

Matlab: Matrix Manipulation

Università di Roma Tor Vergata

- Input and output
  - Example

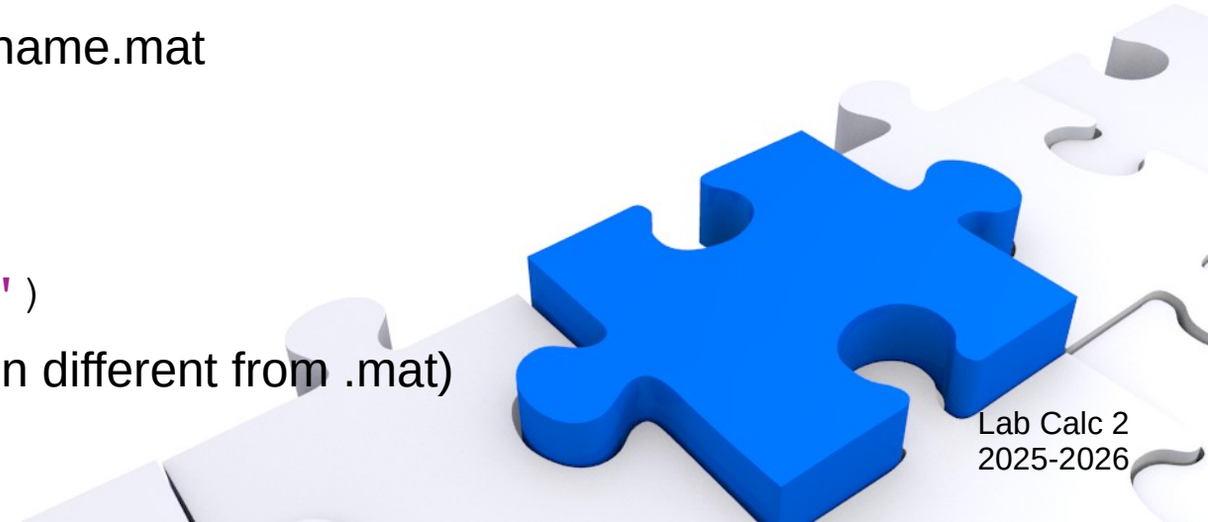
```
>> v = pi;
>> disp(['disp: ', num2str(v)]);
disp: 3.1416
>> fprintf('printf: %f\n', v);
printf: 3.141593
>> fprintf('printf: %.4f\n', v);
printf: 3.1416
>> fprintf('printf: %s %.4f\n', 'pi =', v);
printf: pi = 3.1416
```



Matlab: Matrix Manipulation

Università di Roma Tor Vergata

- Input and output
  - File types
    - .mat: Matlab binary file (platform independent)
    - ascii: text file (Matlab independent)
  - Basic file output
    - `save('filename')`
      - Save all variables from workspace in filename.mat
    - `save('filename', 'a', 'b', __)`
      - Save variables `a` and `b` in filename.mat
    - `save('filename.dat', 'a', '-ascii')`
      - Save variable `a` in filename.dat (extension different from .mat)



Matlab: Matrix Manipulation

Università di Roma Tor Vergata

- Input and output
  - Basic file input
    - `load('filename')`
      - Load all variables from filename.mat
    - `load('filename', 'a', 'b', __)`
      - Load variables `a` and `b` from filename.mat
    - `load('filename.dat', '-ascii')`
      - Load all data from filename.dat into variable `filename`
      - Data in file must be separated by blanks, commas, semicolons
  - There is also a command form of the syntax (no quotes)
    - For example: `load -ascii filename.dat`

Matlab: Matrix Manipulation

Università di Roma Tor Vergata

- Input and output
  - Advanced input and output
    - Open file: `fid = fopen('filename.ext', 'permission')`
      - The file gets an identification number `fid`
      - Permission:
        - `r`, `rt` : open file for reading (`r` for binary / `rt` for ascii)
        - `w`, `wt` : open or create file for writing (binary/ascii); erase existing content
        - `a`, `at` : open or create file for writing (binary/ascii); append data at the end
        - `r+`, `rt+` : open file for reading and writing (binary/ascii)
        - `w+`, `wt+`, `a+`, `at+` : same as `w`, `wt`, `a`, `at`, but also reading
      - Ready to read/write from/to file: fwrite, fprintf, fread, fscanf
    - Close file: `status = fclose(fid)`

Matlab: Matrix Manipulation

Università di Roma Tor Vergata

- Input and output
  - Advanced input and output
    - `count = fwrite(fid, M, 'precision')`
      - Write the values of matrix `M` (linearized) to the binary file related to `fid`
      - Return the number of successfully processed values
    - `[M, count] = fread(fid, size, 'precision')`
      - Read values from the binary file related to `fid` into matrix `M` of size `size`
      - The size can be of the form `n`, `Inf`, or `[m,n]`
    - Precision: `char`, `int8`, `int32`, `int64`, `float32`, `float64`

Matlab: Matrix Manipulation

Università di Roma Tor Vergata

- Input and output
  - Example

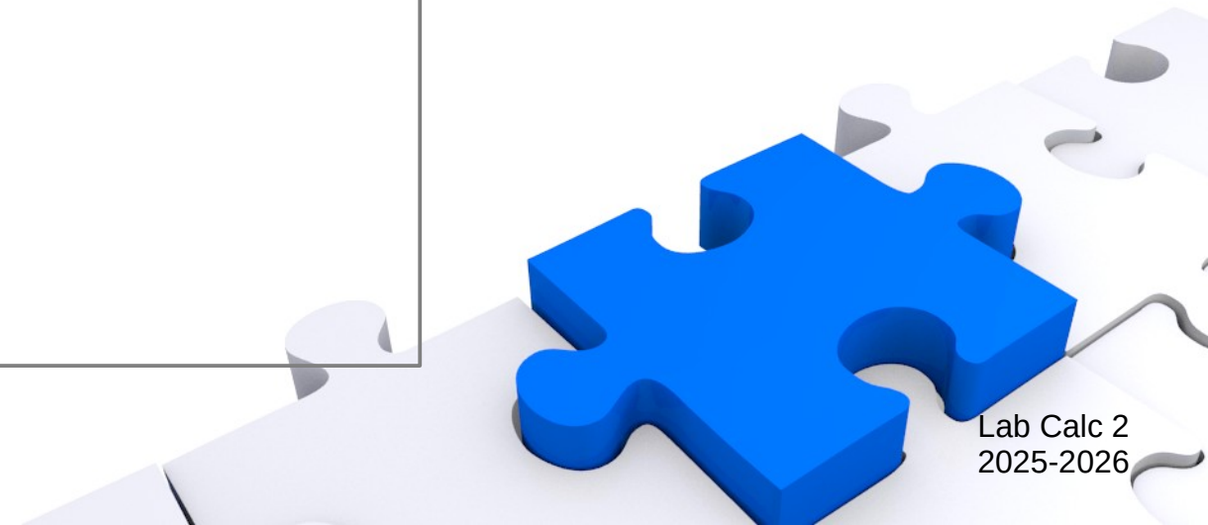
```
>> R = [1, 3, 4, 5, 2];
>> file = input('enter file name: ', 's');
   fid = fopen(file, 'w');
   if fid > 0
       cnt = fwrite(fid, R, 'float64');
       disp([num2str(cnt) ' values written']);
       status = fclose(fid);
   else
       disp('file error');
   end
```

Matlab: Matrix Manipulation

Università di Roma Tor Vergata

- Input and output
  - Example

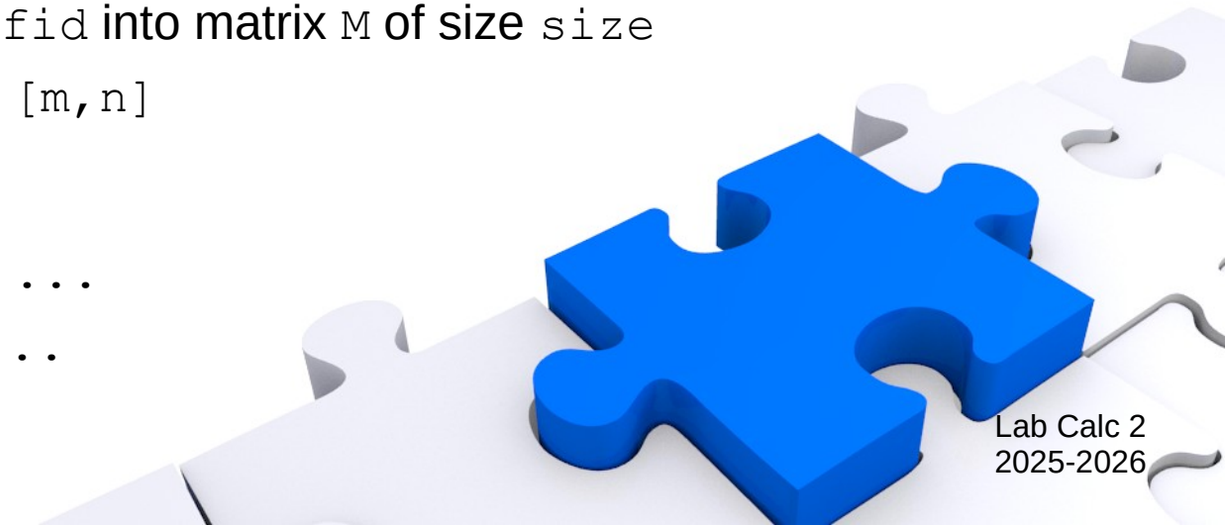
```
>> file = input('enter file name: ', 's');
   fid = fopen(file, 'r');
   if fid > 0
       [R, cnt] = fread(fid, [1, 5], 'float64');
       disp([num2str(cnt) ' values read']);
       status = fclose(fid);
   else
       disp('file error');
   end
>> R
R = [1, 3, 4, 5, 2];
```



Matlab: Matrix Manipulation

Università di Roma Tor Vergata

- Input and output
  - Advanced input and output
    - `count = fprintf(fid, 'format', M)`
      - Write the values of matrix `M` (linearized) to the text file related to `fid`
      - Return the number of successfully processed values
    - `[M, count] = fscanf(fid, 'format', size)`
      - Read values from the text file related to `fid` into matrix `M` of size `size`
      - The size can be of the form `n`, `Inf`, or `[m,n]`
    - Format is the same as `fprintf` before
      - Conversion placeholders: `%d`, `%f`, `%s`, `...`
      - Escape characters: `%%`, `\n`, `\t`, `\\`, `...`



Matlab: Matrix Manipulation

Università di Roma Tor Vergata

- Input and output
  - Example

```
>> x = 0:0.1:1; y = [x; exp(x)];
>> fid = fopen('exp.dat', 'wt');
   if fid > 0
       fprintf(fid, '%6.2f %12.8f\n', y);
       status = fclose(fid);
   end
>> fid = fopen('exp.dat', 'rt');
   if fid > 0
       z = fscanf(fid, '%f', [2, 11]);
       status = fclose(fid);
   end
```

0.00	1.00000000
0.10	1.10517092
...	...