Computational Mathematics

Hilary Term 2019
Lecture 1

Alberto Paganini

Outline for Today

- Functions in Matlab
  - Anonymous functions
  - Optimization
  - Writing m-files

- Algorithm complexity
  - Example: sparse matrices

Writing your own function

Suppose we wish to study the function $f(x) = \frac{e^x}{1+e^{2x}}$.

MATLAB does not have a built-in function of this form.
We may write our own as an *anonymous function.*

Anonymous functions have this structure:

```
myFunction =  @(x,y,z,...)  x+y-2*x+... ;
```

function name   arguments   function definition

Anonymous function - Example

The function $f(x) = \frac{e^x}{1+e^{2x}}$ can be coded as

```
f = @(x) exp(x)./(1+exp(2*x));
```

To call this function in the command window, type

```
>> f(2)
>> f([2 3 9])
```

Anonymous functions in use: integration

Anonymous functions are useful to compute integrals.

For instance, to compute $\int_0^1 \frac{e^x}{1+e^{2x}}\,dx$, define

```
>> f = @(x) exp(x)./(1+exp(2*x));
```

and use Matlab’s built-in function `integral`

```
>> integral(f, 0, 1)
```

Compare with exact solution $\arctan(e) - \arctan(1)$

Optimization tools

Matlab has 3 very useful optimization/roof finding built-in functions

- `fminsearch` find local minimum of nonlinear function
- `fsolve` solve system of nonlinear equations
- `roots` find roots of a polynomial

Using `fminsearch`

This function takes two arguments:

1) a function to minimize,
2) and a starting value.

Example: find a local minimum of $f(x) = \sin(\cos(x) - x^2)$ near $x = 5$.

```
>> f = @(x) sin(cos(x) - x.^2);
>> fminsearch(f, 5);
```

`What happens with fminsearch(f, 4) ?`

Using `fsolve`

The function `fsolve` solves equations of the form $f(x) = 0$.

Example: solve $x^3 + x^2 = e^{-x}$.

```
>> f = @(x) x.^3 + x.^2 - exp(-x);
>> fsolve(f, 1)
```

But be careful, if you evaluate `f(ans)` …

Polynomials

The built-in function `roots` takes a vector of polynomial coefficients.

Example: compute the roots of $x^5 - 2x^2$.

Recall: $x^5 - 2x^2$ = $1x^5 + 0x^4 + 0x^3 \; -2x^2 + 0x^1 + 0x^0$

```
>> roots([1 0 0 -2 0 0])
```

Function files

If a function is too complicate to be written as an anonymous function, we can save it in an M-file, and call it in the usual way.

```
function [out1, out2, ...] = fctName(arg1, arg2,...)
     %statements
     …
     out1 = …;
     out2 = …;
end
```

Function files - Example

```
function out = sum(a, b)

    out = a + b;

end
```

Function files - Example

```
function [out1, out2] = sumandprod(a, b)

    out1 = a + b;
    out2 = a.*b;
end
```

Functions - Scope

The only variables a function can "see" and use are the input argument.

Example:

```
function out = faultyfct()
     out = x;
end
```

Try:

```
>> faultyfct;
>> x = 3;
>> faultyfct;
```

Functions - Scope

Be careful with anonymous functions.

```
>> x = 3;
>> f = @(x) x.^2;
>> f(2)
>> g = @(y) x + y;
>> g(2)
```

Sparse matrices

Many applications lead to matrices that are sparse.

A matrix is sparse if most of its entries are zero.

In such cases, we can save storage space and gain in efficiency by saving these matrices in `sparse` format.

Sparse matrices - Example

```matlab
function playWithSparseMatrices

n = 5e3;

A = sprand(n, n, 1e-5);
B = full(A);

tic, A*A; toc
tic, B*B; toc

end
```

Sparse matrices - Counterexample

```
function playWithSparseMatrices

n = 5e3;

A = sprand(n, n, 0.1);
B = full(A);

tic, A*A; toc
tic, B*B; toc

end
```