9.6 Exercise 6

The theory for numerical integration is discussed in chapter 6.1. Here we will deal with an example of solving integrals numerically, for which no analytical solution exists. The Gaussian bell shape function shows up as solution in many problems including probability theory and many physical problems like the solution of diffusion equations. For $\sigma = 1$ it is

$$F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-y^2}{2}\right) dy \tag{9.10}$$

The normalization is chosen to have $F(+\infty) = 1$, i.e. to represent a probability density.
For numerically solving this problem we will used the MATLAB function `quadgk`.

- Check MATLAB HELP and write a small program for calculating and plotting $F(x)$ between $x_{min} = -10$ and $x_{max} = +10$
- Make sure that you properly incorporate the information in MATLAB HELP about the function to be integrated: `The function y = fun(x) should accept a vector argument x and return a vector result y`
- Check and discuss the correctness of your result
- Although `quadgk` can deal with limits $\pm\infty$ avoid this in your function and use analytical reasons for finding a numerically correct solution.

Numerical integration consumes much CPU time. Thus for many highly relevant integrals numerical efficient approximation exist to reduce the computational effort. For our example it is the error function. As for many examples, the error function does not directly solve our problem but needs for some parameter adaptation. Check MATLAB HELP for the definition and implementation of the error function. Find the necessary parameter transformation to calculate $F(x)$ using the error function.
Next we will use a set of $Z_i(t_i)$ data points from a TD-exercise to repeat fitting and interpolation. Subsequently we use the `MATLAB` routine `quadgk` to perform the integration for calculating

$$\gamma = \exp\left(\int_0^p \frac{Z-1}{p} dp\right) \tag{9.11}$$

The most easiest way to do this is just to open the old examples

- `function fit_test`
- `function my_cubic_spline_test`

combine them and use a nested function function `y=y_from_spline(t)` on which `quadgk` is applied to calculate numerically the integral from 0 to 100.

```
function [c, gamma] = td_fit_int
   p = [1.0 4.0 7.0 10.0 40.0 70.0 100.0]';
   Z = [0.9971 0.98796 0.9788 0.96956 0.8734 0.7764 0.6871]';
   y = (Z-1)./p;
   E = [ones(size(p)) p p.^2];
   c = E\y;
   P = (1:1:100)';
   Y = [ones(size(P)) P P.^2]*c;
   cs = spline(p,y);
   YY = ppval(cs,P);
   gamma = quadgk(@y_from_spline,0,100);
   plot(P,Y,'-',p,y,'o',P,YY,'-r');

       function y=y_from_spline(p)
           y= ppval(cs,p);
       end
end
```