Computational Methods and Data Analysis 2003

Ex. 2: Numerical derivatives

Make a new directory CMDA/ex2 and open it for reading.
The first and second derivative of a function can be approximated numerically as finite differences. From Taylor's expansions one can derive the forward difference formula

$$f'(x) = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h) \tag{1}$$

as well as the backwards difference formula

$$f'(x) \approx \frac{f(x) - f(x-h)}{h} \tag{2}$$

Both approximations have truncation error $\mathcal{O}(h)$ linear in the step h.
More accurate is the central difference formula with error $\mathcal{O}(h^2)$

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} \tag{3}$$

The purpose of this exercise is to calculate the numerical derivatives of a few simple functions as a function of the step h and compare to the exact value calculated analytically. By going from very small to very large steps h (from 10^-20 to 10^0) one can also distinguish round off errors from truncation errors. In particular we will calculate the derivatives of the following functions at the given point:

x^2	at	$x = 1$
x^5	at	$x = 1$
$sin(x)$	at	$x = 0$
$sin(x)$	at	$x = pi/4$
$sin(x)$	at	$x = pi/2$

Besides, the exercise is useful to learn other features of Matlab. You want to write functions which calculate the forward, backward and central difference of a generic function. This means that the function itself has to be passed as argument. You have to do :

- create files fx2.m, fx5.m and fsin.m which calculate function and the exact derivatives for the functions x^2, x^5 and $sin(x)$.
- create files forward_difference.m (etcetera) which calculate the forward difference for a generic function fname. For instance:

```
function Forw_Diff = Forward_Diff(fname, x ,h)
% Forward_Diff approximates the derivative of a function f at point x
% using the following two-points difference formula:
%
% derivative = ( f(x+h) - f(x) ) / h
```

```
%
% where h is the stepsize.  The error is of order 1.

Forw_Diff = (feval(fname,x+h)-feval(fname,x))/h;
```

In this example fname is a character string which gives the name of the function, `feval` is a matlab function which evaluates the function fname in x. In the most recent version of matlab one can refer to a function also with @fname where fname is the name of an m-file, e.g. feval(@fname,x).

- create a script derivatives.m which makes use of the previous files. In this file you have to do several things
  - read from the screen the name of the functions of which you want to calculate the derivative and the value of the argument at which it has to be calculated. The name of the function is read as a string by using the command input with 's'. If 's' is not given matlab tries to evaluate the function and gives an error because the argument of the function is undefined. We also read the value x at which the derivative is evaluated.

    ```
    fname = input('type in function of which derivative has to be
    evaluated','s')
    x = input ('type x, value at which the derivative will be evaluated')
    ```
  - create a grid of values of h.
  - create vectors containing the values calculated with the different approximations for all values of h
  - evaluate also the exact derivative for all values of h
  - calculate the relative error defined as

$$\Delta = \frac{f'_{approx}(x) - f'(x)}{f'(x)} \tag{4}$$

  - plot the calculated error with the different approximation, each with a different linetype.
  - functions that can be expressed as exponential or power law are conveniently plotted in logarithmic scale. Taking the logarithm of the error $\Delta \sim Ch^n$ where C is a constant and n is the order of the approximation, one obtains $ln(\Delta) = ln(C) + nln(h)$ so that the slope in logarithmic scale gives directly the order of the approximation. Analogously one would obtain a straight line for an exponential function $y = be^{ax}$ in a semilogarithmic plot since $ln(y) = ln(b) + ax$