Numerical Analysis 1

Assignment 2

1. Use MATLAB to create plots of the functions $\cos(1.7x)$ and $\sin(1.7x)$ for $x \in [0, 2\pi]$. Enter:

   x=0:0.1:2*pi;
   y1=cos(1.7*x);
   y2=sin(1.7*x);

   The first command creates the vector **x** with the values $0, 0.1, 0.2, ...$ up to 2π. The second and third commands create the vectors **y1, y2** of the same length as **x**, such that $y1(i) = cos(1.7x(i))$ and $y2(i) = sin(1.7x(i))$. Now use subplot and plot to prepare three graphs, *all in the same graphical window*:

   (a) A plot of the graph $(x, \cos(1.7x))$ in **subplot(2,2,1)**.

   (b) A plot of the graph $(x, \sin(1.7x))$ in **subplot(2,2,2)**.

   (c) A plot of $(x, \cos(1.7x))$ and $(x, \sin(1.7x))$ in **subplot(2,2,3)**. Use **legend** to show which is which.

   The **plot** command creates a smooth graph that passes through all the points with coordinates $(x(i), y(i))$. Hence, this graph is only a discrete **approximation** of the accurate (i.e., continuous) graph. The finer spacing we use (say, 0.01 instead of 0.1 in the definition of **x**) the smoother (and more accurate) the graph we get.

   Save your output as an JPG file. For more information use **help print**. Create a hardcopy of your output by printing the file to a printer. If you experience technical problems (e.g., the printer ran out of paper) please refer to the system staff.

   Note: one thing that is emphasized in this class is a clear presentation of numerical and graphical results. Therefore:

   - All plots should be created using **subplot** (remember the rain forests!).
   - To make your plots clear, always use the commands **xlabel, ylabel** and **title**.
   - If you have more then one graph in the same plot use **legend**. You can also add text anywhere inside your plots using **text** ot **gtext**.

2. **Computer Representation of Numbers**

   A floating point number can be represented by

   $$Fl(x) = \pm(1.d_1 d_2 ... d_t)_2 2^e \ ,$$

   where the sequence $(.d_1 d_2 ... d_t)_2$ is called *mantissa*, t is the number of digits in the mantissa, 2 is the *base* (or *radix*) and e is the *exponent*.

(a) For a DEC-VAX using **single precision** we have $t = 24$ and $-128 \le e \le 127$. What are the smallest and largest numbers, m and M, that can be stored in this computer?

(b) Repeat (a) for the case of **double precision**, where $t = 52$ and $-1024 \le e \le 1023$.

(c) What is the smallest value of x in (a) and (b) for which x^{100} will overflow (i.e., $x^{100} > M$)?

3. To approximate the function $f(x) = \sqrt{x}$, we will use the points (1,1), (4,2) and (9,3).

   (a) Write down the linear algebra system that determine interpolation Polynomial $P_2(x)$.

   (b) Use MATLAB to find $P_2(x)$ using direct method, i.e. solve linear algebra system that you just defined.

   (c) Write a .m function file that implements $P_2(x)$ using the MATLAB polynomial commands.

   (d) To see how well the approximation is:

      i. Plot f and P_2 in the interval $[0.01, \;\; 12]$.

      ii. Plot $f - P_2$ in the interval $[0.01, \;\; 12]$.

      iii. Plot the relative error $abs((f - P_2)./f)$ in the interval $[0.01, \;\; 12]$.

      Use the subplot command as much as possible!

   (e) Using the plots determine where the approximation is better/worse.

   (f) Add the point (0,0) and repeat (a)–(c). Do you see any improvement? Deterioration? Where? Why?

THE MATLAB DIGEST

Here are some functions in MATLAB and some useful tips:

- **max**(**x**), **min**(**x**) find the maximal and minimal values of a vector x. See help on them for more details. They are useful in many situations, e.g., calculating the maximal value of an error vector.

- **prod**(**x**) calculates the product (i.e., multiplication) of all the elements of the vector **x**.

- **sum**(**x**) adds the elements of the vector **x**.

- The command **format long** tells MATLAB to print 15 digits on the screen. The default is **format short**, which prints only the 5 significant digits. This command can be convenient when inspecting an error vector. For additional format options see **help format**.

- **poly(A)** finds the coefficients of the characteristic polynomial of the matrix **A**. For example, **poly(eye(2))** returns the vector [**1 -2 1**], because the characteristic polynomial of $I_{2\times 2}$ is $P(x) = (x-1)^2 = 1 * x^2 - 2 * x + 1$. Note: this MATLAB command is actually based on the symbolic program MAPLE, which is available on several UNIX servers in our faculty.

- **roots** finds the roots of a polynomial. For vectors, **roots** and **poly** are inverse functions of each other, up to ordering, scaling, and roundoff error.

- **polyval(P,val)** evaluate polynomial P(x) at x=val. The polynomials $P(x) = 2x^2 + 2x - 4$ and $Q(x) = x^2 - 6$ are represented in MATLAB by:

  ```
  P = [2 2 -4];
  Q = [1 0 -6];
  ```

  $P(4.7)$ is evaluated, for example, using:

  ```
  polyval(P,4.7)
  ```

  You can plot $Q(x)$ in the interval $[-6, \;\; 6]$ using:

  ```
  x = [-6:0.1:6];
  plot(x,polyval(Q,x))
  ```

  The polynomial $S(x) = P(x) + Q(x)$ is calculated using

  ```
  S = P+Q;
  ```

  (but addition or subtraction of polynomials with different degrees takes somewhat more effort). Multiplication is easy and the degrees do not have to be equal. The multiplication $T(x) = P(x) * Q(x)$ is represented by

  ```
  T = conv(P,Q);
  ```