# Numerical Analysis 1

## Assignment 3

1. In this exercise you will analyze the problems that arise when the interpolation polynomial is evaluated using the Vandermonde matrix. We will find the interpolation polynomial $P_n$ for the smooth function $\sin(x)$ and see that as $n$ increases problems arise. To do so, run the following program (available as Vandermun.m from virtual-$\tau$)

```
%  Inverting the Vandermoonde matrix in order to find the
%  interpolation polynomial Pn for sin(x) in the interval [1 2]
%  Pn(x) = c_1 x^n + ..+ c_n x + c_{n+1}
%
clear all
for ip = 0:3
   N  = 10*3^ip;      % Number of interpolation points
   dx = 1/N;
   x  = [1:dx:2]';
% Find the interpolation polynomial using
% the Vandermonde matrix
   V  = vander(x);    % Matlab's command to produce the vandermonde matrix
   C  = V\sin(x);     % Warning: use \, not /
% Plot the results at grid values other than the
% ones used for the interpolation.
   y = x + 0.1*dx*rand(size(x));
   subplot(2,2,ip+1);
   plot(y,polyval(C,y)-sin(y));
   xlabel('x');
   ylabel('Pn(x)-f(x)');
   title(['N = ',num2str(N)]);
end
shg;
```

*Do not submit* the plots or the program, but answer the following questions:

(a) Using the error formula derived in class, show that $|\sin(x) - P_n(x)| \to 0$ as $n \to \infty$ for $x \in [0, 2]$.

(b) Does this agree with your plots?

(c) In practice, is larger $n$ (i.e., more interpolation points) good or bad?

(d) What is the source of the problem with large $n$? (Hint: run the program again but use the interpolation points $x$ rather than the $y$ points in the plot command. Also note the MATLAB warning messages.)

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

   (a) Write the formula for the Lagrange Polynomial $P_2(x)$ that interpolates these three points.

   (b) Find the interpolation Polynomial using the Newton method.

   (c) Verify that the two forms give identical results.

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

   (e) To see how well the approximation in (a) 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!**

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

   (g) Why do you get these results? Do they agree with the formula derived in class for the error bound?

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

3. Given the distinct points $x_0, x_1, \cdots, x_n$, define the Lagrange interpolating polynomials as

$$l_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - x_i}{x_k - x_i} \ , \qquad k = 0, 1, 2, \cdots, n \ .$$

   Prove the following identity:

$$\sum_{k=0}^{n} l_k(x) = 1 \ .$$

   Hint: the trick is to choose a *specific* function and find its interpolating polynomial.

4. Show explicitly that Newton's divided differences do not depend on the order of the variables in the following cases:

   (a) $f[x_0, x_1] = f[x_1, x_0]$

   (b) $f[x_0, x_1, x_2] = f[x_i, x_j, x_k]$, where $[x_i, x_j, x_k]$ is any combination of the nodes $x_0$, $x_1$ and $x_2$.

# THE MATLAB DIGEST

- **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);
```