Question 1 **5 marks**

Consider the problem of interpolating the data

k	0	1	2	3	4
x_k	0	1	2	3	4
y_k	0	1	5	14	35

using a polynomial interpolant $\Pi_4(x)$

(**a**) Construct the Vandermonde matrix V associated with the data $\{x_k\}_{k=0}^4$ from the table above.

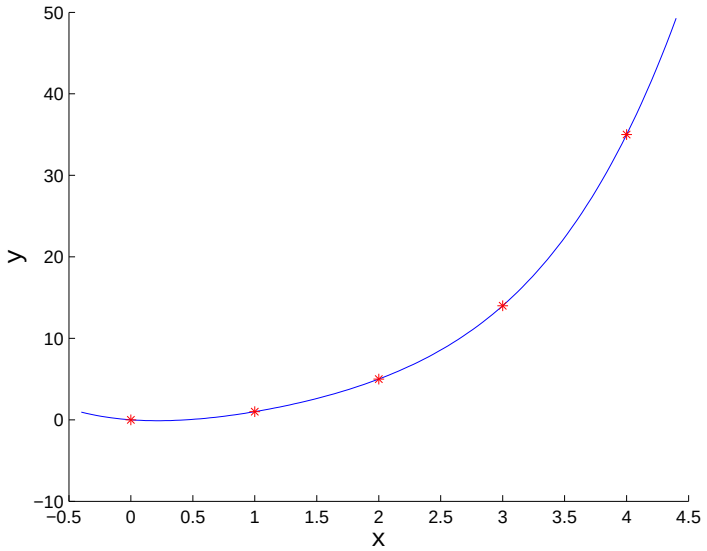
$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 & 16 \\ 1 & 3 & 9 & 27 & 81 \\ 1 & 4 & 16 & 64 & 256 \end{pmatrix}$$

(**b**) Write out the interpolating conditions that $\Pi_4(x)$ satisfies.

$$\begin{aligned} \Pi_4(0) &= a_0 = 0 \\ \Pi_4(1) &= a_0 + a_1 + a_2 + a_3 + a_4 = 1 \\ \Pi_4(2) &= a_0 + 2a_1 + 4a_2 + 8a_3 + 16a_4 = 5 \\ \Pi_4(3) &= a_0 + 3a_1 + 9a_2 + 27a_3 + 81a_4 = 14 \\ \Pi_4(4) &= a_0 + 4a_1 + 16a_2 + 64a_3 + 256a_4 = 35 \end{aligned}$$

(**c**) Solve the resulting linear system to determine the polynomial interpolant $\Pi_4(x)$. Produce a figure that shows the interpolant as well as the points in the table.

$$\Pi_4(x) = -\frac{13}{12}\,x + \frac{67}{24}\,x^2 - \frac{11}{12}\,x^3 + \frac{5}{24}x^4$$



Question 2 **15 marks**

Polynomial interpolation using the Vandermonde matrix, as in Question 1, requires solving a dense linear systems and thus has complexity $O(n^3)$, where $(n+1)$ is the number of interpolation nodes. In this exercise, you will find the interpolant faster. Consider the basis functions

$$\begin{array}{lll} \phi_0(x) = 1 & \phi_2(x) = (x-x_0)(x-x_1) & \dots \\ \phi_1(x) = (x-x_0) & \phi_3(x) = (x-x_0)(x-x_1)(x-x_2) & \phi_n(x) = \prod_{k=0}^{n-1}(x-x_k) \end{array}$$

and the interpolant $\Pi_n(x) = \sum_{k=0}^{n} a_k \phi_k(x)$.

(**a**) Write out a couple of the conditions for the interpolant, starting from the leftmost interpolation node.

$$\begin{aligned} \Pi_n(x_0) &= a_0 = y_0 \\ \Pi_n(x_1) &= a_0 + a_1(x_1 - x_0) = y_1 \\ \Pi_n(x_2) &= a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) = y_2 \\ &\dots = \dots \end{aligned}$$

(**b**) Rewrite the system of equations as a matrix-vector equation for $\vec{a}$. What property does the matrix have? How many flops will it take to solve the system?

$$\begin{pmatrix} 1 & & & & \\ 1 & (x_1-x_0) & & & \\ 1 & (x_2-x_0) & (x_2-x_0)(x_2-x_1) & & \\ 1 & (x_3-x_0) & (x_3-x_0)(x_3-x_1) & (x_3-x_0)(x_3-x_1)(x_3-x_2) & \\ \dots & & & & \ddots \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \end{pmatrix}$$

The matrix is lower triangular so the system can be solved by forward substitution in $\mathrm{O}(n^2)$ flops.

(**c**) Write a pseudo-code for a function that takes an array of interpolation nodes and an array of function values as input and computes the interpolating polynomial using the approach of part (**b**). Implement it in Matlab as a function called `poly_int.m`.

It is important to fill the matrix row-by-row and re-use previously computed elements. If you compute the elements of the matrix, computing the products from scratch for every entry, the flop count will be $O(n^3)$! Here is an example:

Fuction *poly_int*.
In: array of $(n+1)$ nodes x, array of $(n+1)$ y-values y.
Out: array of coefficients a.

1. Initialize V such that $V_{i1} = 1$ for $1 \le i \le n+1$ and $V_{ij} = 0$ for $1 \le i \le n+1,\ 2 \le j \le n+1$.
2. For $i = 2 : n+1$ % loop over rows
   a. For j=2:i % loop over row elements on/left of diagonal
      i. $V_{ij} = V_{i\,j-1} \times (x_{i-1} - x_{j-2})$

   End;

   End;
3. Solve $Va = y$ by forward substitution.

Note, that in this pseudo-code, the indices of the nodes start from 0, while those of the matrix elements start from 1. In the Matlab code below, all indices start from 1. Also, note that the flop count is

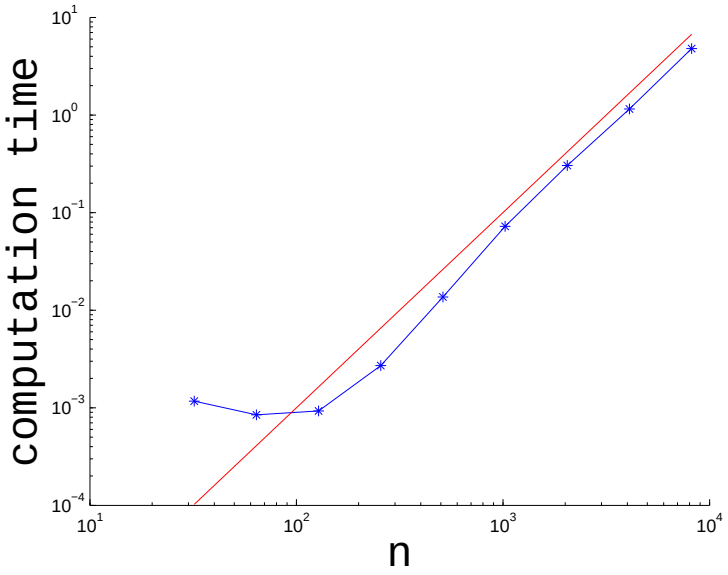
$$\#\text{flops} = \sum_{i=2}^{n+1} \sum_{j=2}^{i} 2 = \sum_{i=2}^{n+1} 2(i-1) = \sum_{i=1}^{n} 2i = n(n+1) = O(n^2)$$

```
function [ a ] = poly_int( xs,ys )
%poly_int: compute a polynomial interpolant to discrete data.
%   Input:  row vectors xs (nodes) and ys (data) of equal length.
%   Output: array of coefficients of the basis functions
%   1, (x-xs(1)), (x-xs(1))*(x-xs(2)), ...
n=size(xs,2)-1; % maximal degree of interpolant
V=zeros(n+1,n+1);
V(:,1)=ones(n+1,1); % first column is all ones
for i=2:n+1 % loop over rows
  for j=2:i % loop over row elements
    V(i,j)=V(i,j-1)*(xs(i)-xs(j-1));
  end;
end;
opts.LT=true; % solve lower triangular system
a=linsolve(V,ys.',opts);
end
```

(**d**) Write a script to call your function for an increasing number of interpolation nodes. Choose nodes $\{x_k\}_{k=0}^{n} = -1 + k\Delta$ for $\Delta = 2/n$ and the function values $y_k = \exp(x_k)$. Plot the time it takes for each function call to complete against n. What is the order of complexity? If you see the computation time increase as $\mathrm{O}(n^3)$ you need to program the function more efficiently!

```
times=[];
ns=[];
f=@(x)exp(x);
for k=5:13
    n=2^k;
    ns=[ns,n];
    del=2/n;
    xs=[-1:del:1];
    ys=f(xs);
    tic;poly_int(xs,ys);times=[times,toc];
end;
figure;hold;
plot(ns,times,'-*');
plot(ns,ns.^2/1e7,'-r');
set(gca,'XScale','log');
set(gca,'YScale','log');
```

This script produces a figure like this, with in red quadratic scaling for comparison:


computation time
n
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{1}
10^{2}
10^{3}
10^{4}