1. Consider the following Octave program.

```
format long;
epsilon = 1e-14;

A = vandermonde (10); % vandermonde(n) gives the n x n Vandermonde matri:
b = sum (A,2);

[L,U,P] = lu(A);
y = L\(P*b);
x = U\y;
r = b-A*x;
[x r]

while norm(r)>epsilon
  v = L\(P*r);
  w = U\v;
  x = x + w;
  r = b-A*x;
  [x r]
end
```

Solve the linear system $Ax = b$ by LU-decomposition, compute the residual $r = b - Ax$

iterative improvement, See Lab 4

(a) What does this code do? Explain.

Notes: ① This code is a "proof of concept", instead of the built-in backslash "\", one should use explicit backward or forward substitution.

② To make an appreciable difference, the computation of the residual should be done at higher precision than the LU-decomposition.

(b) If you run the code in Octave, you'll find the following:

```
ans =

   0.999661641799626   0.000000000000000
   1.000619472137607   0.000000016163540
   0.999521918673179  -0.000000093248673
   1.000205001753909  -0.000000229105353
   0.999945946146878  -0.000000005587935
   1.000009127129884  -0.000000104308128
   0.999999009142851  -0.000000208616257
   1.000000066933674  -0.000000119209290
   0.999999997438762   0.000000000000000
   1.000000000042420   0.000000000000000

ans =

   0.999992500612691   0.000000000000000
   1.000014060201136   0.000000000000000
   0.999988837483866   0.000000000000000
   1.000004942746758   0.000000000000000
   0.999998650834743   0.000000000000000
   1.000000235987559   0.000000000000000
   0.999999973495598   0.000000000000000
   1.000000001846354   0.000000000000000
   0.999999999927497   0.000000000000000
   1.000000000001225   0.000000000000000
```

What can you say about the condition of the problem? Is the residual a good error indicator?

The Vandermonde test problem is known to be ill-conditioned. This is manifest in the above in that a small residual does not always guarantee an equally small error in x.

(c) Will this code always terminate? Explain.

(10+10+10)

No, for two reasons.

① Iterative improvement may not converge at all. The iteration rule can be symbolically written as

$$x_{k+1} = \tilde{U}^{-1}\tilde{L}^{-1}b + \underbrace{(I - \tilde{U}^{-1}\tilde{L}^{-1}A)}_{=:B} x_k$$

where $\tilde{L}^{-1}$ and $\tilde{U}^{-1}$ are the approximate inversions of the LU factors. If the problem is poorly conditioned, the spectral radius can be larger than 1, and the residual can possibly grow without bounds.

② Even if there is convergence (in principle), hard-coding an absolute error will not work for every case. One should monitor the decrease of some norm of r and terminate if it does not decrease appreciably.

Final note: The demonstration in part (b) seems to "work" due to a bug in the linear algebra package LAPACK on PPC-Linux. On i86 systems, the initial result is already much better, and no subsequent further improvement occurs.

2. Use Newton's method for solving the quadratic equation $x^2 = q$ for a given positive real number q.

(a) Show that, in this case, the fixed point iteration reads

$$x_{k+1} = \frac{x_k}{2} + \frac{q}{2x_k}.$$

(b) Let x denote the exact solution. Show that

$$x_{k+1} - x = \frac{(x_k - x)^2}{2x_k}.$$

(c) What can you say about the order of convergence?

(10+5+5)

(a) Take $f(x) = x^2 - q$

$$\Rightarrow x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{x_k^2 - q}{2x_k} = \frac{x_k}{2} + \frac{q}{2x_k}$$

(b) $$x_{k+1} - x = \frac{x_k}{2} + \frac{q}{2x_k} - x$$

$$= \frac{1}{2x_k}\left(x_k^2 + x^2 - 2xx_k\right)$$

$$= \frac{(x_k - x)^2}{2x_k}$$

(c) If $|x| > M$ and $|x_0 - x| \leq \frac{M}{2}$, then

$$|x_{k+1} - x| \leq \frac{|x_k - x|^2}{M} \quad \text{for } \varepsilon(M) \text{ small enough}$$

$\Rightarrow$ Quadratic convergence for any $M > 0$.

If $x = 0$, convergence is only linear.

3. Compute the location of the quadrature points for Gauss quadrature on the interval $[-1, 1]$ with two quadrature points.

Hint: you may use the fact that Gauss quadrature integrates polynomials up to a certain degree exactly. (10)

Gauss quadrature with two quadrature points reads

$$\int_{-1}^{1} f(x)\,dx \approx f(\alpha) + f(-\alpha)$$

because of symmetry considerations on the fact that the weights need to add up to 2.

We must integrate polynomials up to degree 3 exactly in particular

$$\int_{-1}^{1} x^2\,dx = \frac{2}{3} \overset{!}{=} 2\alpha^2 = \alpha^2 + (-\alpha)^2$$

$$\Rightarrow \boxed{\alpha = \sqrt{\frac{1}{3}}}$$

Remark: $\int_{-1}^{1} x\,dx = 0 = \int_{-1}^{1} x^3\,dx$; these are trivially satisfied by the symmetry of the Gauss formula, hence do not give new information

4. **Note:** The remaining questions are all connected, but can be worked on independently.

In the following, we consider the cubic interpolating spline on N equidistant grid points $x_j = jh$ for $j = 0, \ldots, N-1$.

Let y_j denote the given value of the spline on the grid nodes, and y_j'' the (unknown) value of the second derivative.

Show that the following system of linear equations is satisfied:

$$y_{j-1}'' + 4y_j'' + y_{j+1}'' = \frac{6}{h^2}(y_{j-1} - 2y_j + y_{j+1}). \qquad (*)$$

Hint: Use the notation from class: write

$$s_j(x) = a_j(x - x_j)^3 + b_j(x - x_j)^2 + c_j(x - x_j) + d_j$$

to denote the spline function on the jth interval. Show that $2b_j = y_j''$ and $d_j = y_j$ and eliminate a_j and c_j by using the matching conditions at the interpolation nodes. (10)

$$s_j' = 3a_j(x - x_j)^2 + 2b_j(x - x_j) + c_j$$

$$s_j'' = 6a_j(x - x_j) + 2b_j$$

$$y_j = s_j(x_j) = d_j\,; \quad y_j'' = s_j''(x_j) = 2b_j$$

Matching conditions:

$$s_{j-1}(x_{j-1}) = s_j(x_{j-1}) \Rightarrow y_{j-1} = -a_j h^3 + b_j h^2 - c_j h + y_j \qquad (*)$$

$$s_{j-1}'(x_{j-1}) = s_j'(x_{j-1}) \Rightarrow c_{j-1} = 3a_j h^2 - 2b_j h + c_j$$

$$s_{j-1}''(x_{j-1}) = s_j''(x_{j-1}) \Rightarrow y_{j-1}'' = -6a_j h + y_j'' \Rightarrow 6a_j h^3 = (y_j'' - y_{j-1}'')h^3$$

From (*): $6(y_j - y_{j-1}) = 6a_j h^3 - 3y_j'' h^2 + 6c_j h = -(2y_j'' + y_{j-1}'')h^2 + 6c_j h$

Into (**): $0 = 3(y_j'' - y_{j-1}'')h^2 - 6y_j'' h^2 + 6c_j h - 6c_{j-1} h$

$$= -3(y_j'' - y_{j-1}'')h^2 + 6(y_j - y_{j-1}) + (2y_j'' + y_{j-1}'')h^2 - 6(y_{j-1} - y_{j-2}) - (2y_{j-1}'' + y_{j-2}'')h^2$$

$$\Rightarrow \boxed{6(y_j - 2y_{j-1} + y_{j-2}) = (y_{j-2}'' + 4y_{j-1}'' + y_j'')h^2}$$

5. Assume that the cubic spline from question 4 is defined on a periodic grid. In other words, interpolation node x_0 is identified with x_N.

   Explain why system (*) has the same number of equations as there are unkowns, i.e., why we do not need to impose additional conditions as in the case of non-periodic splines. (10)

On each interval, there are 4 unknowns → 4N total

At each node, there are

2 interpolation conditions (for S_{j-1} and S_j)

1 continuity condition for S'

1 continuity condition for S''

} → also 4N total

Note that the periodic grid has no boundary nodes, so the continuity conditions can be imposed at every node.

6. Let u_j with $j = 0, \ldots, N-1$ be a given tuple of numbers on an N-periodic equidistant grid with grid spacing $h = 2\pi/N$. Let $\hat{u}_k$ with $k = -N/2, \ldots, N/2-1$ denote its discrete Fourier transform. Further, let τ_ℓ denote translation by ℓ grid points, i.e.

$$(\tau_\ell u)_j = u_{j+\ell}.$$

Show that

$$(\widehat{\tau_\ell u})_k = e^{ihk\ell}\hat{u}_k.$$

(10)

$$\left(\widehat{\tau_\ell u}\right)_k = \frac{1}{N}\sum_{j=0}^{N-1} e^{-ijhk} u_{j+\ell} \qquad m = j+\ell$$

$$= \frac{1}{N}\sum_{m=\ell}^{N-1+\ell} e^{-ihk(m-\ell)} u_m$$

$$= e^{ihk\ell}\,\frac{1}{N}\sum_{m=\ell}^{N-1+\ell} e^{-ihkm} u_m$$

Since e^{-ihkm} and u_m are N-periodic in m, we can let the sum run from $m=0$ to $m=N-1$, which completes the proof.

7. The linear system (*) for computing the periodic spline in question 4 can be solved by taking the discrete Fourier transform on both sides of the equality.

   (a) Show that in the special case that the right hand side is non-zero on a single interpolation node only,

   $$y''_{j-1} + 4y''_j + y''_{j+1} = \delta_{0j}$$

   the solution of the linear system is given by

   $$y''_j = \frac{1}{N} \sum_{k=-N/2}^{N/2-1} \frac{e^{ihkj}}{4 + 2\cos(kh)}.$$

   Note: You are required to use the discrete Fourier transform and inverse discrete Fourier transform. The given solution is only for your convenience—direct substitution into the linear system will not earn credit.

   (b) Show that the result from part (a) implies that at the gridpoint $j = N/2$,

   $$|y''_{N/2}| \le \text{const} \cdot \frac{1}{N}.$$

   Hint: First show that

   $$y''_{N/2} = \frac{1}{N} \sum_{k=1}^{N/2} \frac{(-1)^k}{2 + \cos(kh)}.$$

(10+10)

(a) Use question 6 and write $z_j = y''_j$ for simplicity:

$$e^{-ikh}\hat{z}_k + 4\hat{z}_k + e^{ikh}\hat{z}_k = \frac{1}{N}\sum_{j=0}^{N-1} e^{-ijhk}\delta_{0j} = \frac{1}{N}$$

$$\Rightarrow \hat{z}_k = \frac{1}{N}\frac{1}{4 + 2\cos kh}$$

Use inverse DFT:

$$z_j = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{z}_k e^{ikhj} = \frac{1}{N}\sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \frac{e^{ikhj}}{4 + 2\cos kh}$$

(b) For $j = \frac{N}{2}$, $e^{ikhj} = e^{i\frac{2\pi}{N}k\frac{N}{2}} = e^{i\pi k} = (-1)^k$

Moreover, $\cos(kh) = \cos(-kh)$ for $k = 1, \dots, \frac{N}{2}-1$

and $\cos(0h) = \cos\left(\pm\frac{N}{2}h\right) = 1$,

so that

$$z_j = \frac{1}{N}\sum_{k=1}^{\frac{N}{2}} \frac{(-1)^k}{2 + \cos(kh)}$$

$$\Rightarrow |z_j| \le \frac{1}{N}\left|\sum_{\substack{k=1 \\ k \text{ odd}}}^{N/2} \underbrace{\frac{1}{2+\cos(kh)} - \frac{1}{2+\cos((k+1)h)}}\right|$$

$$= \frac{\cos((k+1)h) - \cos(kh)}{\left(2+\cos(kh)\right)\left(2+\cos((k+1)h)\right)}$$

$$= \frac{\cos kh\cos h - \sin kh \sin h - \cos kh}{\left(2+\cos(kh)\right)\left(2+\cos((k+1)h)\right)}$$

$$\le \frac{1}{N}\sum_{\substack{k=1 \\ k \text{ odd}}}^{N/2} \frac{|1-\cos h||\cos(kh)| + |\sin h||\sin kh|}{|2-1|\cdot|2-1|}$$

$$\le \frac{1}{N}\sum_{\substack{k=1 \\ k \text{ odd}}}^{N/2} \underbrace{\underbrace{\left(|1-\cos h| + |\sin h|\right)}_{\le 2} = O(h)}_{= O(1)}$$

□

8. Write out the linear system (*) of question 4 in matrix form *for the periodic case.*

You now know three methods for numerically solving this system:

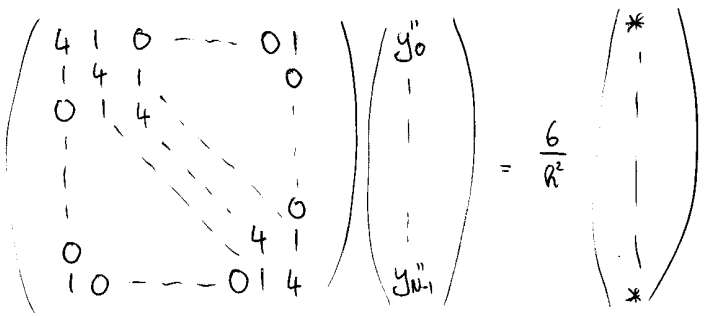
(a) Gaussian elimination (or LU-decomposition)

(b) Iterative methods

(c) The procedure outlined in question 7, implemented in terms of the FFT and IFFT

Comment on the computational complexity and efficiency of each for this particular matrix.

(10)



(a) The system is almost tridiagonal. When doing Gaussian elimination, fill-in only occurs in last row and last column, so the system can be solved in $O(N)$ operations.

(b) The system is strictly diagonally dominant, hence can be solved by Jacobi or Gauss-Seidel. However, only 2-4 iterations reach the operation count of a direct method, hence not competitive.

(c) The DFT method requires one FFT and one IFFT, each with $O(N \log N)$ operations. Not as good as a direct solve, but for moderate N competitive and easy to implement in Octave.