



1 a) We have that

$$\|x\|_1^2 = \left(\sum_{i=1}^n |x_i| \right)^2 = \sum_{i=1}^n \sum_{j=1}^n |x_i| |x_j| = \sum_{i=1}^n |x_i|^2 + \sum_{i=1}^n \sum_{j \neq i} |x_i| |x_j| \geq \|x\|_2^2,$$

since $|x_i| |x_j|$ is non-negative. We also have

$$\begin{aligned} \|x\|_1^2 &= \left(\sum_{i=1}^n |x_i| \right)^2 \\ &= \sum_{i=1}^n \sum_{j=1}^n |x_i| |x_j| \\ &\leq \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n (|x_i|^2 + |x_j|^2) \quad (ab \leq \frac{1}{2}(a^2 + b^2) \text{ for any } a, b \in \mathbb{R}) \\ &= n \sum_{i=1}^n |x_i|^2 \\ &= n \|x\|_2^2. \end{aligned}$$

Hence, $\|x\|_2 \leq \|x\|_1 \leq \sqrt{n} \|x\|_2$. Alternatively, for the second step we could have used the Cauchy-Schwarz inequality

$$|y^H x| \leq \|x\|_2 \|y\|_2$$

and chosen

$$y_i = \begin{cases} 1 & x_i = 0, \\ x_i / |x_i| & x_i \neq 0. \end{cases}$$

Since $|y_i| = 1$, this would have given $\|x\|_1 = |y^H x| \leq \|x\|_2 \|y\|_2 = \sqrt{n} \|x\|_2$, giving the same result as above.

b) For the first inequality,

$$\|x\|_2^2 = \sum_{i=1}^n |x_i|^2 \geq \max_{1 \leq i \leq n} |x_i|^2 = \|x\|_\infty^2.$$

For the second inequality we have

$$\|x\|_2^2 = \sum_{i=1}^n |x_i|^2 \leq \sum_{i=1}^n \max_{1 \leq j \leq n} |x_j|^2 = \sum_{i=1}^n \|x\|_\infty^2 = n \|x\|_\infty^2.$$

Hence, $\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n} \|x\|_\infty$.

c)

$$\|x\|_1 = \sum_{i=1}^n |x_i| \geq \max_{1 \leq i \leq n} |x_i| = \|x\|_\infty$$

and

$$\|x\|_1 = \sum_{i=1}^n |x_i| \leq \sum_{i=1}^n \max_{1 \leq j \leq n} |x_j| = n \|x\|_\infty$$

give us that $\|x\|_\infty \leq \|x\|_1 \leq n \|x\|_\infty$.

d) Using, for example, a) we obtain, for any $A \in \mathbb{C}^{n \times n}$ and $0 \neq x \in \mathbb{C}^n$:

$$\frac{\|Ax\|_2}{\|x\|_2} \leq \frac{\|Ax\|_1}{\|x\|_1 / \sqrt{n}},$$

and as a result $\|A\|_2 \leq \sqrt{n} \|A\|_1$. On the other hand,

$$\frac{\|Ax\|_1}{\|x\|_1} \leq \frac{\sqrt{n} \|Ax\|_2}{\|x\|_2},$$

which yields $\|A\|_1 \leq \sqrt{n} \|A\|_2$ after taking supremum over all $0 \neq x \in \mathbb{C}^n$. Combining the two inequalities we conclude that

$$\frac{1}{\sqrt{n}} \|A\|_2 \leq \|A\|_1 \leq \sqrt{n} \|A\|_2.$$

Other estimates are obtained in a similar fashion.

2 We define the matrix norm induced by the vector norm $\|\cdot\|$ by

$$\|A\| := \max_{x \in \mathbb{C}^n \setminus \{0\}} \frac{\|Ax\|}{\|x\|}.$$

Suppose A has p distinct eigenvalues $\lambda_1, \dots, \lambda_p$ corresponding to eigenvectors v_1, \dots, v_p respectively. Then

$$\|A\| \geq \max_i \frac{\|Av_i\|}{\|v_i\|} = \max_i \frac{|\lambda_i| \|v_i\|}{\|v_i\|} = \max_i |\lambda_i| = \rho(A).$$

3 a) We begin by seeing that for a general $x = (x_1, x_2)^T$,

$$Ax = \begin{pmatrix} x_1 - x_2 \\ x_1 + x_2 \end{pmatrix},$$

such that

$$\|A\|_{1, \infty, \mathbb{R}} = \max_{x \in \mathbb{R}^2 \setminus \{0\}} \frac{\|Ax\|_1}{\|x\|_\infty} = \max_{x \in \mathbb{R}^2 \setminus \{0\}} \frac{|x_1 - x_2| + |x_1 + x_2|}{\max\{|x_1|, |x_2|\}}. \quad (1)$$

We consider different cases for the numerator in (1): Either $x_1 - x_2 \geq 0$ or $x_1 - x_2 < 0$, and either $x_1 + x_2 \geq 0$ or $x_1 + x_2 < 0$. This gives us four different cases. We will treat the first two in detail, leaving the remaining two largely self-explanatory.

Case 1: $x_1 - x_2 \geq 0$ and $x_1 + x_2 \geq 0$.

Adding the two inequalities, we see that $x_1 \geq 0$. Furthermore, we have that $x_1 \geq x_2$ and $x_1 \geq -x_2$, such that $x_1 = |x_1| \geq |x_2|$. We then find that

$$|x_1 - x_2| + |x_1 + x_2| = x_1 - x_2 + x_1 + x_2 = 2x_1 = 2|x_1| = 2 \max\{|x_1|, |x_2|\}.$$

Case 2: $x_1 - x_2 \geq 0$ and $x_1 + x_2 < 0$.

From the first inequality we have $x_2 \leq x_1$, and using this in the second inequality we find $x_2 < 0$. Furthermore, we have $-x_2 \geq -x_1$ and $-x_2 > x_1$ such that $-x_2 = |x_2| \geq |x_1|$. We then find that

$$|x_1 - x_2| + |x_1 + x_2| = x_1 - x_2 - x_1 - x_2 = -2x_2 = 2|x_2| = 2 \max\{|x_1|, |x_2|\}.$$

Case 3: $x_1 - x_2 < 0$ and $x_1 + x_2 \geq 0$.

We have $x_2 > 0$ and $|x_2| \geq |x_1|$ such that

$$|x_1 - x_2| + |x_1 + x_2| = 2x_2 = 2|x_2| = 2 \max\{|x_1|, |x_2|\}.$$

Case 4: $x_1 - x_2 < 0$ and $x_1 + x_2 < 0$.

We have $x_1 < 0$ and $|x_1| \geq |x_2|$ such that

$$|x_1 - x_2| + |x_1 + x_2| = -2x_1 = 2|x_1| = 2 \max\{|x_1|, |x_2|\}.$$

In all cases, the denominator becomes the same, and inserting into (1) we get

$$\|A\|_{1\infty, \mathbb{R}} = 2.$$

b) Taking $x = (1 + i, 1 - i)^T$, we have

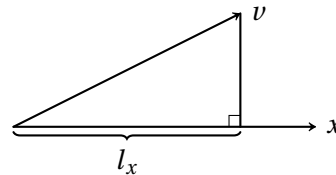
$$Ax = \begin{pmatrix} 2i \\ 2 \end{pmatrix},$$

meaning $\|Ax\|_1 = 4$. Also, $\|x\|_\infty = \sqrt{2}$, such that

$$\|A\|_{1\infty} = \max_{z \in \mathbb{C}^2 \setminus \{0\}} \frac{\|Az\|_1}{\|z\|_\infty} \geq \frac{\|Ax\|_1}{\|x\|_\infty} = 2\sqrt{2} > 2 = \|A\|_{1\infty, \mathbb{R}}.$$

4 **a)** We have

$$\begin{aligned} \|E\|_2 &= \max_{x \in \mathbb{C}^n \setminus \{0\}} \frac{\|u v^H x\|_2}{\|x\|_2} \\ &= \max_{x \in \mathbb{C}^n \setminus \{0\}} \frac{|v^H x| \|u\|_2}{\|x\|_2} \\ &= \|u\|_2 \max_{x \in \mathbb{C}^n \setminus \{0\}} \frac{|v^H x|}{\|x\|_2} \\ &= \|u\|_2 \frac{|v^H v|}{\|v\|_2} \\ &= \|u\|_2 \|v\|_2. \end{aligned}$$



(Maximum is attained when $x = \alpha v$, for $\alpha \neq 0$ scalar.)

Note that $l_x = |v^H x| / \|x\|_2$ represents the length of the orthogonal projection of v on x , so l_x is maximal for vectors x having the *same direction* as v .

b) For the Frobenius norm we get

$$\|E\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^n |u_i \bar{v}_j|^2} = \sqrt{\sum_{i=1}^n |u_i|^2 \sum_{j=1}^n |v_j|^2} = \sqrt{\sum_{i=1}^n |u_i|^2} \sqrt{\sum_{j=1}^n |v_j|^2} = \|u\|_2 \|v\|_2.$$

5 For a unitary matrix Q , $Q^H Q = I$, where I is the identity matrix. We recall that the 2-norm of a vector is given by $\|v\|_2 = \sqrt{v^H v}$. This gives

$$\|Qx\|_2 = \sqrt{(Qx)^H Qx} = \sqrt{x^H Q^H Qx} = \sqrt{x^H x} = \|x\|_2.$$

Also, let φ_i be an eigenvector of Q with corresponding eigenvalue λ_i . Then

$$\|Q\varphi_i\|_2 = \|\varphi_i\|_2. \quad (2)$$

But

$$\|Q\varphi_i\|_2 = \|\lambda_i \varphi_i\|_2 = |\lambda_i| \|\varphi_i\|_2. \quad (3)$$

Comparing (2) and (3) we get that

$$|\lambda_i| = 1.$$

In particular, $\rho(Q) = 1$.

6 The purpose of this exercise is to show how the storage format and structure of a matrix can influence the performance of the LU factorization of the matrix.

a) We here consider the one-dimensional problem

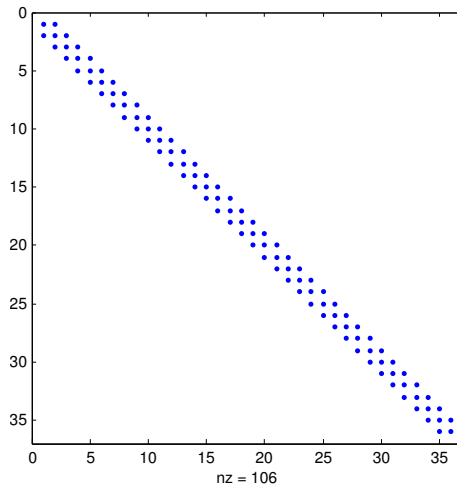
$$\begin{aligned} -\frac{d^2 u}{dx^2} &= f(x), & x \in [0, 1], \\ u &= 0, & x \in \{0, 1\}. \end{aligned}$$

We discretize the problem using finite differences, applied on a partition of the domain $[0, 1]$ into $n + 1$ subintervals. We get the difference equations

$$\begin{aligned} -\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} &= f_i, & 1 \leq i \leq n, \\ u_0 &= u_{n+1} = 0, \end{aligned}$$

where $h = 1/(n + 1)$ and u_i is the numerical approximation of $u(ih)$. Since the difference equations relate three neighbouring points, the resulting matrix must be tridiagonal (an $n \times n$ banded matrix with bandwidth 1). See Figure 1.

With Gaussian elimination we eliminate the entries below the diagonal. For a full matrix this would require $O(n^3)$ flops, but for a one-dimensional Poisson problem we only have to eliminate the entries in the first sub-diagonal, since the rest are zero. This would require only $O(n)$ flops. Generally for a banded matrix of bandwidth k , we must eliminate all nonzero entries below the diagonal. This would require $O(nk^2)$ flops.

Figure 1: Nonzero entries in the 1D Poisson matrix for $n = 36$

n	Time (s)	Time/ n^3
900	0.0104	$1.43 \cdot 10^{-11}$
1600	0.0443	$1.08 \cdot 10^{-11}$
2500	0.1551	$9.929 \cdot 10^{-12}$
3600	0.4090	$8.766 \cdot 10^{-12}$

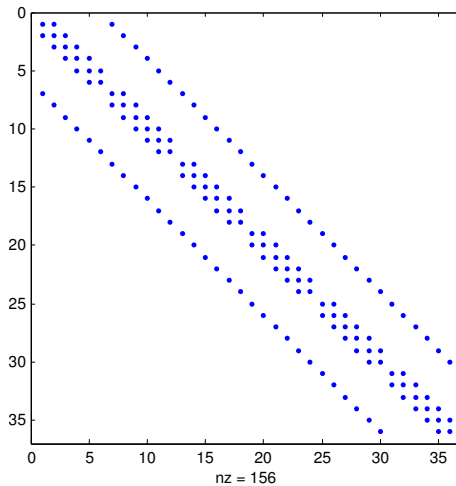
Table 1: One-dimensional Poisson problem with full matrix

- i)** For $n \times n$ matrices, Gaussian elimination in MATLAB is done by LU factorization. When we save the matrix as a full matrix, MATLAB will eliminate all entries under the diagonal, including entries that are already zeros. The amount of computing time will be $O(n^3)$. That means, doubling the problem dimension would increase the computing time by a factor 8. This can be observed in Table 1. Since the predicted computing time is $O(n^3)$ we observe that Time/ n^3 is approximately constant ($\approx 1 \cdot 10^{-11}$).¹ This confirms that the computing time is $O(n^3)$.
- ii)** We repeat the numerical experiment in **i)** but using the sparse format in MATLAB for storing the matrix. Here we observe $O(n)$ computing time (see Table 2).

¹Alternatively, a **logLog** plot of time versus n would give a straight line whose slope is approximately equal to 3.

n	Time (s)	Time/ n
900	$1.81 \cdot 10^{-4}$	$2.01 \cdot 10^{-7}$
1600	$3.11 \cdot 10^{-4}$	$1.94 \cdot 10^{-7}$
2500	$4.83 \cdot 10^{-4}$	$1.93 \cdot 10^{-7}$
3600	$6.93 \cdot 10^{-4}$	$1.93 \cdot 10^{-7}$

Table 2: One-dimensional Poisson problem with sparse matrix

Figure 2: Nonzero entries in the 2D Poisson matrix for $n = 6$.

N	Time (s)	Time/ N^3
900	0.0097	$1.3 \cdot 10^{-11}$
1600	0.0428	$1.04 \cdot 10^{-11}$
2500	0.1526	$9.765 \cdot 10^{-12}$
3600	0.4051	$8.682 \cdot 10^{-12}$

Table 3: Two-dimensional Poisson problem with full matrix

b) We consider the two-dimensional problem

$$-\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = f(x, y), \quad (x, y) \in [0, 1] \times [0, 1],$$

$$u = 0, \quad x = 0, x = 1, y = 0, y = 1.$$

We partition $[0, 1]$ into $n + 1$ intervals in both the x - and y -directions, and obtain the following difference equations upon discretizing the problem:

$$-\frac{u_{i,j-1} + u_{i-1,j} - 4u_{i,j} + u_{i+1,j} + u_{i,j+1}}{h^2} = f, \quad 1 \leq i, j \leq n,$$

$$u_{0,j} = u_{n+1,j} = u_{i,0} = u_{i,n+1} = 0, \quad 0 \leq i, j \leq n + 1,$$

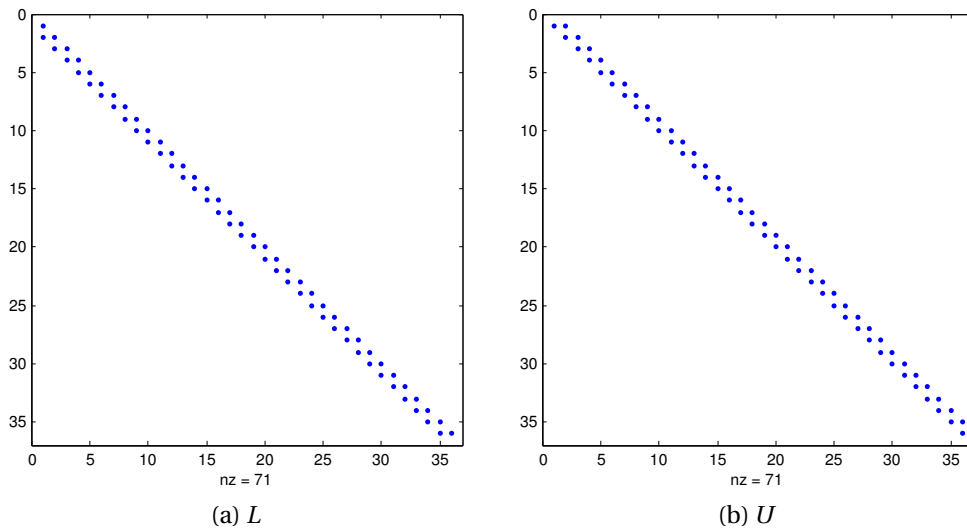
where $h = 1/(n + 1)$ and $u_{i,j}$ is the numerical approximation of $u(x_i, y_j)$ with $x_i = ih$ and $y_j = jh$. Each difference equation relates the three neighbouring points in the x -direction and the three neighbouring points in the y -direction. We number the unknown variables first in the x -direction from bottom to top. The resulting matrix has almost the same structure as in a), but in addition we have two bands a distance n away from the diagonal. See Figure 2.

The number of unknowns has now become $N = n^2$, and the bandwidth $k = 2n + 1 = 2\sqrt{N} + 1$. Storing the matrix as a full matrix will require that the computing time increases as $O(N^3)$, but if we store it as a sparse matrix we predict a computing time of order $O(Nk^2) = O(N^2)$.

i) See Table 3 for the computing time with full matrix. We observe that the result is like in a), with computing time $O(N^3)$.

N	Time (s)	Time/ N^2
900	$2.242 \cdot 10^{-3}$	$2.768 \cdot 10^{-9}$
1600	$6.824 \cdot 10^{-3}$	$2.666 \cdot 10^{-9}$
2500	$1.552 \cdot 10^{-2}$	$2.484 \cdot 10^{-9}$
3600	$2.979 \cdot 10^{-2}$	$2.298 \cdot 10^{-9}$

Table 4: Two-dimensional Poisson problem with sparse matrix

Figure 3: Nonzero entries for L and U after LU factorization of the 1D Poisson matrix.

- ii)** We repeat the experiment using MATLAB's sparse storage format (see Table 4). Indeed, we observe $O(N^2)$ computing time here. The matrix structure in the 2D case is different from the 1D case (for the same number of unknowns), but by exploiting sparsity, we get a significant speedup here too.

This exercise illustrates the problems encountered with using Gaussian elimination in the numerical solution of differential equations in several space-dimensions (e.g. 2D or 3D). The bandwidth increases with space-dimension even if the dimension of the linear system is kept constant. This makes Gaussian elimination less attractive for the solution of large linear systems even with sparse matrix format. After LU-factorization the storage requirement also increases, especially in the case of higher space-dimension. There are $O(n)$ nonzero entries before LU-factorization. After LU-factorization nearly all entries between the diagonal and the outermost sub- or super-diagonal are nonzero. The storage increases to about $O(nk)$. See Figures 3 and 4.

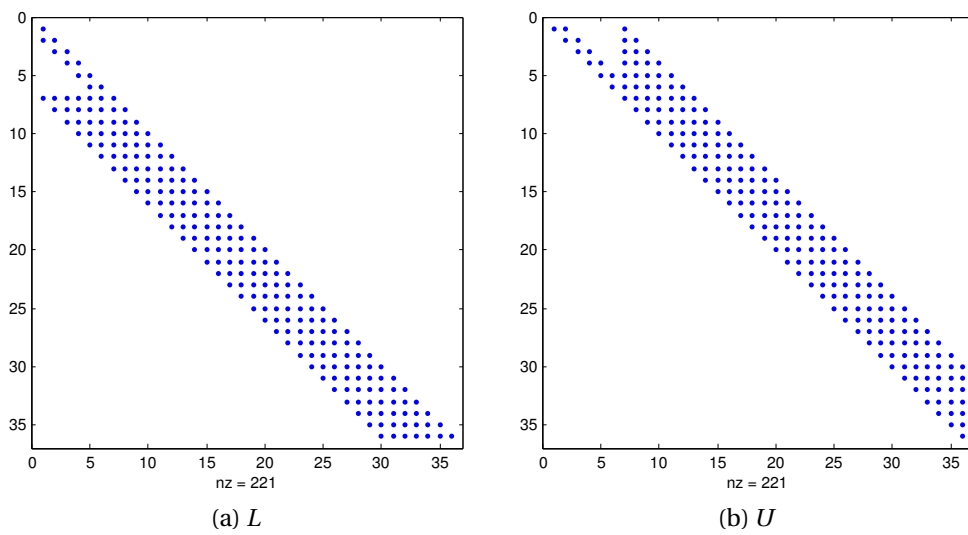


Figure 4: Nonzero entries for L and U after LU factorization of the 2D Poisson matrix.