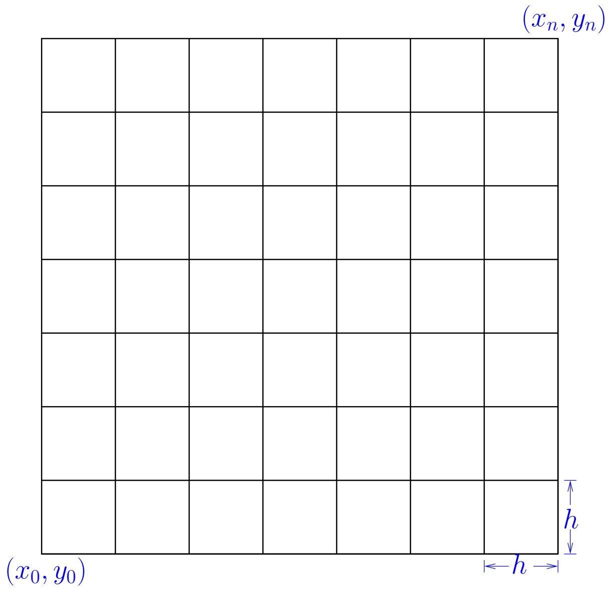
1 The Poisson problem in $\mathbb{R}^2$

The 2D-Poisson problem is given by

$$\begin{aligned} -\nabla^2 u &= f \qquad \text{in } \Omega = (0,1) \times (0,1), \\ u &= 0 \qquad \text{on } \partial\Omega, \end{aligned} \tag{1}$$

where $\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$.

Finite difference grid:



Discrete equations:

By using the notation

$$u_{i,j} \simeq u(x_i, y_j) = u(ih, jh),$$

and discretizing (1) using the 5-point stencil, the discrete equations read:

$$-\frac{(u_{i+1,j} - 2u_{i,j} + u_{i-1,j})}{h^2} - \frac{(u_{i,j+1} - 2u_{i,j} + u_{i,j-1})}{h^2} = f_{i,j} \qquad 1 \leq i,j \leq n-1.$$

1.1 Diagonalization

Let

$$\underline{\mathbf{U}} = \begin{bmatrix} u_{1,1} & \cdots & \cdots & u_{1,n-1} \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ u_{n-1,1} & \cdots & \cdots & u_{n-1,n-1} \end{bmatrix}$$

and

$$
\underline{\mathbf{T}} = \begin{bmatrix} 2 & -1 & & & 0 \\ -1 & 2 & -1 & & \\ & & \ddots & & \\ & & -1 & 2 & -1 \\ 0 & & & -1 & 2 \end{bmatrix}
$$

Then,

$$
\begin{aligned}
(\underline{\mathbf{T}}\,\underline{\mathbf{U}})_{ij} &= 2u_{i,j} - u_{i+1,j}, & i &= 1, \\
(\underline{\mathbf{T}}\,\underline{\mathbf{U}})_{ij} &= -u_{i-1,j} + 2u_{i,j} - u_{i+1,j}, & 2 \leq &i \leq n-2, \\
(\underline{\mathbf{T}}\,\underline{\mathbf{U}})_{ij} &= -u_{i-1,j} + 2u_{i,j}, & i &= n-1.
\end{aligned}
$$

and thus,

$$
\boxed{\frac{1}{h^2}(\underline{\mathbf{T}}\,\underline{\mathbf{U}})_{ij} \simeq -\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j}.}
$$

Similarly,

$$
\boxed{\frac{1}{h^2}(\underline{\mathbf{U}}\,\underline{\mathbf{T}})_{ij} \simeq -\left(\frac{\partial^2 u}{\partial y^2}\right)_{i,j}.}
$$

Exercise: Show this!

Our finite difference system can then be expressed as

$$
\frac{1}{h^2}(\underline{\mathbf{T}}\,\underline{\mathbf{U}} + \underline{\mathbf{U}}\,\underline{\mathbf{T}})_{i,j} = f_{i,j} \qquad \text{for} \qquad \begin{array}{c} 1 \leq i \leq n-1, \\ 1 \leq j \leq n-1, \end{array}
$$

or

$$
\underline{\mathbf{T}}\,\underline{\mathbf{U}} + \underline{\mathbf{U}}\,\underline{\mathbf{T}} = \underline{\mathbf{G}} \tag{2}
$$

where

$$
\underline{\mathbf{G}} = h^2 \begin{bmatrix} f_{1,1} & \cdots & \cdots & f_{1,n-1} \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ f_{n-1,1} & \cdots & \cdots & f_{n-1,n-1} \end{bmatrix}
$$

The matrix, $\underline{\mathbf{T}}$, may be diagonalized by

$$\underline{\mathbf{T}} = \underline{\mathbf{Q}}\underline{\mathbf{\Lambda}}\underline{\mathbf{Q}}^T,$$

where $\underline{\mathbf{\Lambda}}$ is a diagonal matrix and $\underline{\mathbf{Q}}^T\underline{\mathbf{Q}} = \underline{\mathbf{I}}$, the identity matrix. Inserted into (2), we get

$$\underline{\mathbf{Q}}\underline{\mathbf{\Lambda}}\underline{\mathbf{Q}}^T\underline{\mathbf{U}} + \underline{\mathbf{U}}\underline{\mathbf{Q}}\underline{\mathbf{\Lambda}}\underline{\mathbf{Q}}^T = \underline{\mathbf{G}}.$$

Multiplying from the right with $\underline{\mathbf{Q}}$ and multiplying from the left with $\underline{\mathbf{Q}}^T$ gives:

$$\underline{\mathbf{\Lambda}}\underbrace{\underline{\mathbf{Q}}^T\underline{\mathbf{U}}\underline{\mathbf{Q}}}_{\equiv\underline{\widetilde{\mathbf{U}}}} + \underbrace{\underline{\mathbf{Q}}^T\underline{\mathbf{U}}\underline{\mathbf{Q}}}_{\equiv\underline{\widetilde{\mathbf{U}}}}\underline{\mathbf{\Lambda}} = \underbrace{\underline{\mathbf{Q}}^T\underline{\mathbf{G}}\underline{\mathbf{Q}}}_{\equiv\underline{\widetilde{\mathbf{G}}}}.$$

Hence, (2) may be solved in three steps:

Step 1)

$$\boxed{\underline{\widetilde{\mathbf{G}}} = \underline{\mathbf{Q}}^T\underline{\mathbf{G}}\underline{\mathbf{Q}}} \quad - \quad \text{matrix-matrix products.}$$

Step 2)

$$\underline{\mathbf{\Lambda}}\,\underline{\widetilde{\mathbf{U}}} + \underline{\widetilde{\mathbf{U}}}\,\underline{\mathbf{\Lambda}} = \underline{\widetilde{\mathbf{G}}}$$

or

$$\begin{aligned} \lambda_i\widetilde{u}_{i,j} + \widetilde{u}_{i,j}\lambda_j &= \widetilde{g}_{i,j}, \qquad 1 \le i,j \le n-1 \\ (\lambda_i + \lambda_j)\widetilde{u}_{i,j} &= \widetilde{g}_{i,j}, \qquad 1 \le i,j \le n-1 \\ \boxed{\widetilde{u}_{i,j} = \frac{\widetilde{g}_{i,j}}{\lambda_i + \lambda_j}} & \qquad 1 \le i,j \le n-1. \end{aligned}$$

Step 3)

$$\boxed{\underline{\mathbf{U}} = \underline{\mathbf{Q}}\underline{\widetilde{\mathbf{U}}}\underline{\mathbf{Q}}^T} \quad - \quad \text{matrix-matrix products.}$$

Note:

$$\underline{\mathbf{U}}, \underline{\widetilde{\mathbf{U}}}, \underline{\widetilde{\mathbf{G}}}, \underline{\mathbf{Q}}, \underline{\mathbf{Q}}^T \in \mathbb{R}^{(n-1)\times(n-1)}.$$

1.1.1 Computational cost

$N = N_{d.o.f.} = (n-1)^2 \sim \mathcal{O}(n^2) \qquad (n \gg 1)$.

Step 1)

$$\underline{\widetilde{\mathbf{G}}} = \overbrace{\underline{\mathbf{Q}}^T \underbrace{\underline{\mathbf{G}}\underline{\mathbf{Q}}}_{\mathcal{O}(n^3)}}^{\mathcal{O}(n^3)} \qquad \longrightarrow \qquad \mathcal{O}(n^3) \text{ operations.}$$

Step 2)

$$\widetilde{u}_{i,j} = \frac{\widetilde{g}_{i,j}}{\lambda_i + \lambda_j} \qquad \longrightarrow \qquad \mathcal{O}(n^2) \text{ operations.}$$

Step 3)

$$\underline{\mathbf{U}} = \overbrace{\underline{\mathbf{Q}} \underbrace{\underline{\widetilde{\mathbf{U}}}\underline{\mathbf{Q}}^T}_{\mathcal{O}(n^3)}}^{\mathcal{O}(n^3)} \qquad \longrightarrow \qquad \mathcal{O}(n^3) \text{ operations.}$$

In summary, we can compute the discrete solution, $\underline{\mathbf{U}}$, in $\underline{\mathcal{O}(n^3) = \mathcal{O}(N^{3/2}) \text{ operations}}$.

Note:

This method is an example of a direct method.

1.1.2 Comparison with other direct methods

Computational cost		
Method	Operations	Memory requirement
Diagonalization	$\mathcal{O}(N^{3/2}) = \mathcal{O}(n^3)$	$\mathcal{O}(N) = \mathcal{O}(n^2)$
Full LU	$\mathcal{O}(N^3) = \mathcal{O}(n^6)$	$\mathcal{O}(N^2) = \mathcal{O}(n^4)$
Banded LU	$\mathcal{O}(Nb^2) = \mathcal{O}(n^4)$	$\mathcal{O}(Nb) = \mathcal{O}(n^3)$

Table 1: Computational cost and memory requirement for direct methods. For the bandwidth, we have $b \sim \mathcal{O}(n)$.

The diagonalization method is much more attractive in $\mathbb{R}^2$ than in $\mathbb{R}^1$!

1.1.3 The matrices, $\underline{\mathbf{Q}}$ and $\underline{\mathbf{\Lambda}}$.

Consider the eigenvalue problem

$$\begin{aligned} -u_{xx} &= \lambda u \qquad \text{in } \Omega = (0,1), \\ u(0) = u(1) &= 0, \end{aligned}$$

which has solutions

$$\begin{aligned} \bar{u}_j(x) &= \sin(j\pi x), \\ \bar{\lambda}_j &= j^2\pi^2, \end{aligned} \qquad j = 1, 2, \ldots, \infty.$$

Consider now the discrete eigenvalue problem:

$$\underline{\mathbf{T}}\underline{\widetilde{q}}_j = \lambda_j \underline{\widetilde{q}}_j.$$

Try

$$\begin{aligned} (\underline{\widetilde{q}}_j)_i &= \bar{u}_j(x_i) \\ &= \sin(j\pi x_i) \\ &= \sin(j\pi(ih)), \qquad \left(h = \frac{1}{n}\right) \\ &= \sin\left(\frac{ij\pi}{n}\right) \end{aligned}$$

Operating on $\underline{\widetilde{q}}_j$ with $\underline{\mathbf{T}}$ gives

$$(\underline{\mathbf{T}}\underline{\widetilde{q}}_j)_i = \underbrace{2\left(1 - \cos\left(\frac{j\pi}{n}\right)\right)}_{\lambda_j}\underbrace{\sin\left(\frac{ij\pi}{n}\right)}_{(\underline{\widetilde{q}}_j)_i}.$$

Set $\underline{q}_j = \alpha \underline{\widetilde{q}}_j$, and choose α such that $\underline{q}_j$ is normalized:

$$\begin{aligned} &\underline{q}_j^T \underline{q}_j = 1, \\ &\qquad\Downarrow \\ &(\underline{q}_j)_i = \sqrt{\frac{2}{n}}\sin\left(\frac{ij\pi}{n}\right), \qquad 1 \le i, j \le n-1, \\ &\lambda_j = 2\left(1 - \cos\left(\frac{j\pi}{n}\right)\right). \end{aligned}$$

For $\underline{j \ll n}$, we observe that

$$\lambda_j \simeq 2\left(1 - \left(1 - \frac{1}{2}\frac{j^2\pi^2}{n^2} + \ldots\right)\right) \simeq \frac{j^2\pi^2}{n^2}.$$

Since $h = \frac{1}{n}$, we have

$$\lambda_j \simeq h^2 j^2 \pi^2 = h^2 \bar{\lambda}_j, \qquad \text{for } j \ll n.$$

This is the same as saying that the first (lowest) eigenvalues for the continuous case are well approximated by our finite difference formulation.

Note that in this case

$$Q_{ij} = (\underline{q}_j)_i = \sqrt{\frac{2}{n}} \sin\left(\frac{ij\pi}{n}\right), \qquad 1 \leq i, j \leq n-1,$$

and that indeed

$$\underline{\mathbf{Q}}^T = \underline{\mathbf{Q}}.$$

From the comparison of computational cost shown earlier, the diagonalization approach to solving the discrete Poisson problem appears promising.

<u>Question 1</u>:

Can the matrix-matrix multiplications be done fast?

<u>Question 2</u>:

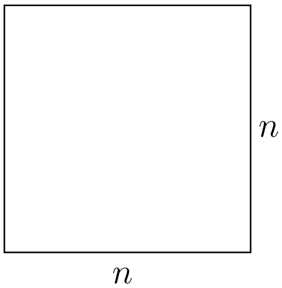
Can the matrix-matrix multiplications be parallelized?

<u>Question 3</u>:

Can we do better?

1.1.4 Numerical results

- Diagonalization
- "Standard" $m \times m$
- PC, Pentium III, 512 MB RAM



$$\mathcal{N}_{\text{d.o.f.}} = (n-1)^2$$

$$\tau^n = \text{ total simulation time (in seconds)}$$
$$\tau_1^n = \frac{\tau}{n^2}$$
$$r_n = \frac{\tau_1^n}{\tau_1^{\frac{n}{2}}}$$

n	τ^n	τ_1^n	r^n
32	$1.80 \cdot 10^{-2}$	$1.76 \cdot 10^{-5}$	
64	$1.50 \cdot 10^{-1}$	$3.66 \cdot 10^{-5}$	2.1
128	1.20	$7.34 \cdot 10^{-5}$	2.0
256	9.84	$1.50 \cdot 10^{-4}$	2.0
512	103.9	$3.96 \cdot 10^{-4}$	2.6
1024	873.2	$8.33 \cdot 10^{-4}$	2.1
	$\tau^n \sim \mathcal{O}(n^3)$	$\tau_1^n \sim \mathcal{O}(n)$	

Table 2: Simulation results for the numerical approximation of the two-dimensional Poisson equation by the use of diagonalization techniques.

1.2 Fast diagonalization methods

The most expensive operation in the diagonalization method introduced in the previous section is of the type,

$$\underline{v}^* = \underline{\mathbf{Q}}\underline{v} = \underline{\mathbf{Q}}^T\underline{v},$$

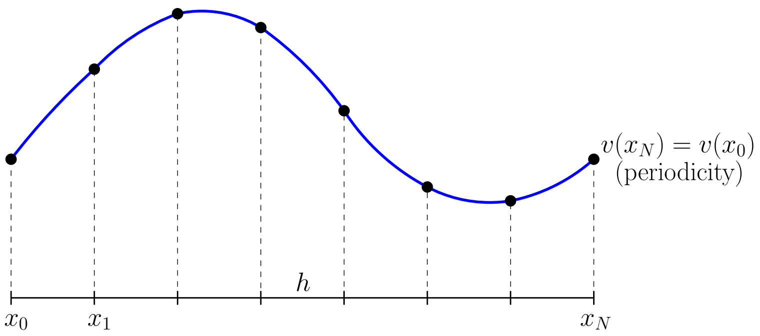
where

$$Q_{ij} = \sqrt{\frac{2}{n}}\sin\left(\frac{ij\pi}{n}\right) \qquad 1 \leq i, j \leq n-1.$$

We will now consider ways to obtain $\underline{v}^*$ in $\mathcal{O}(n\log n)$ operations instead of $\mathcal{O}(n^2)$.

1.2.1 Discrete Fourier Transform (DFT)

Consider a periodic function $v(x)$ with period 2π. Consider sampling this function at the equidistant points $x_j, \quad j = 0, 1, \ldots, N$, with $x_j = jh, \quad h = \frac{2\pi}{N}$. Let $v_j = v(x_j) = v(jh), \quad j = 0, 1, \ldots, N$.



Consider the vectors $\underline{\varphi}_k$, where

$$(\underline{\varphi}_k)_j = e^{ikx_j}, \qquad j, k = 0, 1, \ldots, N-1.$$

Note that the vector elements in $\underline{\varphi}_k$ represent the values of the function $\varphi_k(x) = e^{ikx}$ sampled at the discrete points $x_j, \quad j = 0, 1, \ldots, N-1$. Note also that the function $\varphi_k(x) = e^{ikx}$ is an eigenfunction of the Laplace operator with periodic boundary conditions.

The vectors $\{\underline{\varphi}_k\}_{k=0}^{N-1}$ form a basis for the N-dimensional vector space $\mathbb{C}^N$. In particular, we have that

$$\underline{\varphi}_k^H \underline{\varphi}_k = \begin{cases} N, & k = l, \\ 0, & k \neq l, \end{cases} \qquad k, l = 0, 1, \ldots, N-1.$$

The vector

$$\underline{v} = \begin{bmatrix} v_0 \\ \vdots \\ v_{N-1} \end{bmatrix} \in \mathbb{R}^N$$

can be expressed in this basis as

$$\underline{v} = \sum_{k=0}^{N-1} \hat{v}_k \underline{\varphi}_k$$

or

$$v_j = \sum_{k=0}^{N-1} \hat{v}_k (\underline{\varphi}_k)_j = \sum_{k=0}^{N-1} \hat{v}_k e^{ikx_j},$$

where $\hat{v}_k$, are the discrete Fourier coefficients given by

$$\hat{v}_k = \frac{1}{N} \sum_{j=0}^{N-1} v_j e^{-ikx_j}, \qquad \begin{aligned} x_j &= jh \\ h &= \tfrac{2\pi}{N} \end{aligned} \qquad k = 0, 1, \ldots, N-1$$

<u>**Summary (DFT)**</u>:

$$v_j = \sum_{k=0}^{N-1} \hat{v}_k e^{ijkh}, \qquad j = 0, 1, \ldots, N-1,$$

where

$$\hat{v}_k = \frac{1}{N} \sum_{j=0}^{N-1} v_j e^{-ijkh}, \qquad k = 0, 1, \ldots, N-1,$$

and

$$h = \frac{2\pi}{N}.$$

1.2.2 Discrete Sine Transform (DST)

The Discrete Sine Transform is applicable to a function $v(x)$ which is <u>periodic</u> with period 2π and <u>odd</u>. Discretize the function on an equidistant grid on $[0, \pi]$ with $\overline{h = \frac{\pi}{n}}$. Set

$$v_j = v(x_j) = v(jh) = v\left(\frac{j\pi}{n}\right), \qquad j = 0, 1, \ldots, n.$$

Since v is odd,

$$v(x_0) = v(x_n) = 0.$$

The discretized function is therefore represented by the $(n-1)$ real values $v_1, \ldots, v_{n-1}$, i.e., by the vector

$$\underline{v} = \begin{bmatrix} v_1 \\ \vdots \\ v_{n-1} \end{bmatrix} \in \mathbb{R}^{n-1}.$$

An orthogonal basis for $\mathbb{R}^{n-1}$ is given by the vectors $\underline{\psi}_k, \quad k = 1, \ldots, n-1$, where

$$(\underline{\psi}_k)_j = \sin\left(\frac{kj\pi}{n}\right), \qquad j = 1, \ldots, n-1,$$

and with

$$\underline{\psi}_k^T \underline{\psi}_l = \begin{cases} \frac{n}{2}, & k = l, \\ 0, & k \neq 0. \end{cases}$$

In terms of this basis, we can write $\underline{v}$ as

$$v_j = \sum_{k=1}^{n-1} \widetilde{v}_k \sin\left(\frac{kj\pi}{n}\right), \qquad j = 1, \ldots, n-1,$$

where

$$\widetilde{v}_k = \frac{2}{n} \sum_{j=1}^{n-1} v_j \sin\left(\frac{jk\pi}{n}\right), \qquad k = 1, \ldots, n-1.$$

We can also write this as

$$\underline{\widetilde{v}} = \underline{\mathbf{S}}(\underline{v}) \qquad \text{(DST)}.$$

and

$$\underline{v} = \underline{\mathbf{S}}^{-1}(\underline{\widetilde{v}}) \qquad \text{(inverse DST)}.$$

Note that $\underline{\mathbf{S}}(\cdot)$ and $\underline{\mathbf{S}}^{-1}(\cdot)$ are related as

$$\boxed{\underline{\mathbf{S}} = \frac{2}{n}\underline{\mathbf{S}}^{-1}}$$

Also note that

$$\underline{\mathbf{Q}} = \sqrt{\frac{2}{n}}\underline{\mathbf{S}}^{-1},$$

and

$$\underline{\mathbf{Q}} = \sqrt{\frac{2}{n}}\underline{\mathbf{S}}.$$

Now, consider the matrix $\underline{\mathbf{F}}^{(N)}$ where

$$\begin{aligned} F_{k,j}^{(N)} &= e^{-ijkh} \\ &= \cos\left(\frac{jk2\pi}{N}\right) - i\sin\left(\frac{jk2\pi}{N}\right), \qquad 0 \le j,k \le N-1. \end{aligned}$$

Note that

$$F_{k,j}^{(2n)} = \cos\left(\frac{jk\pi}{n}\right) - i\sin\left(\frac{jk\pi}{n}\right), \qquad 0 \le j,k \le 2n-1.$$

Now, consider

$$\underline{v} = \begin{bmatrix} v_1 \\ \vdots \\ v_{n-1} \end{bmatrix} \in \mathbb{R}^{n-1}.$$

Construct the extended vector as an "odd" extension

$$w = \begin{bmatrix} 0 \\ v_1 \\ \vdots \\ v_{n-1} \\ 0 \\ -v_{n-1} \\ \vdots \\ -v_1 \end{bmatrix} \in \mathbb{R}^{2n}.$$

First, note that

$$(\underline{\mathbf{F}}^{(2n)}\underline{w})_k = \sum_{j=0}^{2n-1} e^{\frac{-ijk\pi}{n}} w_j = 2n\hat{w}_k,$$

where $\hat{w}$, $\quad k = 0, 1, \ldots, 2n-1$, are the discrete Fourier coefficients. Second,

$$
\begin{aligned}
(\underline{\mathbf{F}}^{(2n)}\underline{w})_k &= \sum_{j=0}^{2n-1} \left[\cos\left(\frac{jk\pi}{n}\right) - i \sin\left(\frac{jk\pi}{n}\right) \right] w_j \\
&= \sum_{j=0}^{2n-1} \underset{\text{odd}}{\underset{\uparrow}{w_j}} \underset{\text{even}}{\underset{\uparrow}{\cos}} \left(\frac{jk\pi}{n}\right) - i \sum_{j=0}^{2n-1} \underset{\text{odd}}{\underset{\uparrow}{w_j}} \underset{\text{odd}}{\underset{\uparrow}{\sin}} \left(\frac{jk\pi}{n}\right) \\
&= -2i \sum_{j=0}^{n-1} w_j \sin\left(\frac{jk\pi}{n}\right) \\
&= -2i \sum_{j=1}^{n-1} w_j \sin\left(\frac{jk\pi}{n}\right) \qquad \text{(since } w_0 = 0\text{)}.
\end{aligned}
$$

Hence

$$
\frac{i}{2} (\underline{\mathbf{F}}^{(2n)}\underline{w})_k = \sum_{j=1}^{n-1} w_j \sin\left(\frac{jk\pi}{n}\right) = \frac{n}{2} \widetilde{w}_k.
$$

Since

$$
w_j = v_j, \qquad j = 1, \ldots, n-1,
$$

it follows that

$$
\widetilde{w}_k = \widetilde{v}_k, \qquad k = 1, \ldots, n-1.
$$

In summary, for $k = 1, \ldots, n-1$,

$$
\begin{aligned}
\widetilde{v}_k = \widetilde{w}_k &= \frac{2}{n} \frac{i}{2} (\underline{\mathbf{F}}^{(2n)}\underline{w})_k \\
&= \frac{i}{n} (\underline{\mathbf{F}}^{(2n)}\underline{w})_k \\
&= \frac{i}{n} 2n \hat{w}_k \\
&= 2i \hat{w}_k.
\end{aligned}
$$

By computing the discrete Fourier coefficients $\hat{w}_k$, we can find the discrete sine coefficients $\widetilde{v}_k$, $\quad k = 1, \ldots, n-1$, where

$$
\underline{\widetilde{v}} = \underline{\mathbf{S}}(\underline{v}) = \sqrt{\frac{2}{n}} \underline{\mathbf{Q}} \underline{v}.
$$

The operator $(\underline{\mathbf{F}}^{(2n)}\underline{w})$ can be computed efficiently by a FFT in $\mathcal{O}(2n \log 2n) \sim \mathcal{O}(n \log n)$ operations.

This leads to the modified algorithm (Poisson solver):

1)

$$
\begin{aligned}
&\underline{\widetilde{\mathbf{G}}} = \underline{\mathbf{Q}}^T \underline{\mathbf{G}}\underline{\mathbf{Q}} \\
\Rightarrow \quad &\underline{\widetilde{\mathbf{G}}}^T = \underline{\mathbf{Q}}^T \underline{\mathbf{G}}^T \underline{\mathbf{Q}} \\
&\quad = \underline{\mathbf{Q}}\underline{\mathbf{G}}^T \underline{\mathbf{Q}}^T \qquad (\underline{\mathbf{Q}} = \underline{\mathbf{Q}}^T) \\
&\quad = \underline{\mathbf{Q}}(\underline{\mathbf{Q}}\underline{\mathbf{G}})^T \\
&\quad = \sqrt{\frac{2}{n}}\underline{\mathbf{S}}^{-1}\left(\left(\sqrt{\frac{n}{2}}\underline{\mathbf{S}}(\underline{\mathbf{G}})\right)^T\right) \\
&\quad = \underline{\mathbf{S}}^{-1}((\underline{\mathbf{S}}(\underline{\mathbf{G}}))^T) \qquad \mathcal{O}(n^2 \log n)
\end{aligned}
$$

2)

$$
\widetilde{U}_{ji} = \frac{\widetilde{G}_{ji}}{\lambda_j + \lambda_i} \qquad \mathcal{O}(n^2)
$$

3)

$$
\begin{aligned}
\underline{\mathbf{U}} &= \underline{\mathbf{Q}}\underline{\widetilde{\mathbf{U}}}\underline{\mathbf{Q}}^T \\
&= \underline{\mathbf{Q}}(\underline{\mathbf{Q}}\underline{\widetilde{\mathbf{U}}}^T)^T \\
&= \underline{\mathbf{S}}^{-1}((\underline{\mathbf{S}}(\underline{\widetilde{\mathbf{U}}}^T))^T) \qquad \mathcal{O}(n^2 \log n)
\end{aligned}
$$

Again,

$$
\underline{\mathbf{S}} = \frac{2}{n}\underline{\mathbf{S}}^{-1}
$$

and $\underline{\widetilde{v}} = \underline{\mathbf{S}}\,\underline{v}$ is obtained as follows

i)

$$
\underline{v} \in \mathbb{R}^{n-1} \quad \rightarrow \quad \underline{w} \in \mathbb{R}^{2n}
$$

ii)

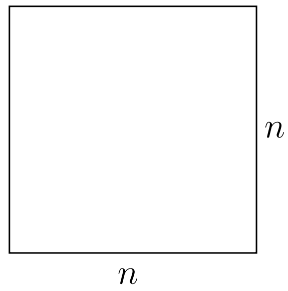
Compute $\underline{\hat{w}}$ via FFT ($\mathcal{O}(n \log n)$).

iii)

$$
\widetilde{v}_k = 2i\hat{w}_k, \qquad k = 1, \ldots, n-1.
$$

1.2.3 Numerical results

- Diagonalization
- DST (FFT)
- PC, Pentium III, 512 MB RAM



$$\mathcal{N}_{\text{d.o.f.}} = (n-1)^2$$

$$\tau^n = \text{ total simulation time (in seconds)}$$
$$\tau_1^n = \frac{\tau}{n^2}$$

n	τ^n	τ_1^n	$\tau_1^n(m \times m)$
32	$2.36 \cdot 10^{-2}$	$2.31 \cdot 10^{-5}$	$1.76 \cdot 10^{-5}$
64	$1.11 \cdot 10^{-1}$	$2.71 \cdot 10^{-5}$	$3.66 \cdot 10^{-5}$
128	$5.19 \cdot 10^{-1}$	$3.17 \cdot 10^{-5}$	$7.34 \cdot 10^{-5}$
256	2.35	$3.58 \cdot 10^{-5}$	$1.50 \cdot 10^{-4}$
512	10.5	$3.99 \cdot 10^{-5}$	$3.96 \cdot 10^{-4}$
1024	46.2	$4.41 \cdot 10^{-5}$	$8.33 \cdot 10^{-4}$
	$\tau^n \sim \mathcal{O}(n^2 \log n)$	$\tau_1^n \sim \mathcal{O}(\log n)$	$\tau_1^n \sim \mathcal{O}(n)$

Table 3: Simulation results for the numerical approximation of the two-dimensional Poisson equation by the use of fast diagonalization techniques.