# Fast Poisson Solvers and FFT

Tom Lyche

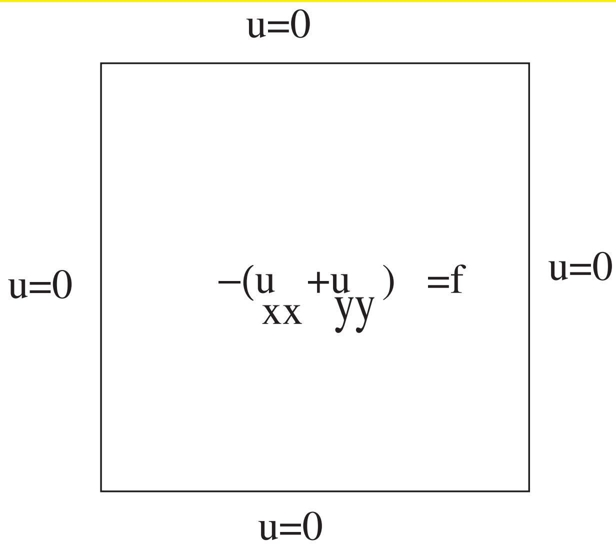
University of Oslo

Norway

# Contents of lecture

- Recall the discrete Poisson problem
- Recall methods used so far
- Exact solution by diagonalization
- Exact solution by Fast Sine Transform
  - The sine transform
  - The Fourier transform
  - The Fast Fourier Transform
  - The Fast Sine Transform
  - Algorithm

# Recall 5 point-scheme for the Poisson Problem




- $h = 1/(m+1)$ gives $n := m^2$ interior grid points
- Discrete approximation $\boldsymbol{V} = [v_{j,k}] \in \mathbb{R}^{m,m}$ with $v_{j,k} \approx u(jh, kh)$.
- Matrix equation $\boldsymbol{TV} + \boldsymbol{VT} = h^2\boldsymbol{F}$ with $\boldsymbol{T} =$ tridiag$(-1, 2, -1) \in \mathbb{R}^{m,m}$ and $\boldsymbol{F} = [f(jh, kh)] \in \mathbb{R}^{m,m}$
- Linear system $\boldsymbol{Ax} = \boldsymbol{b}$ with $\boldsymbol{A} = \boldsymbol{T} \otimes \boldsymbol{V} + \boldsymbol{V} \otimes \boldsymbol{T}$, $\boldsymbol{x} =$ vec$(\boldsymbol{V}) \in \mathbb{R}^n$ and $\boldsymbol{b} = h^2$vec$(\boldsymbol{F}) \in \mathbb{R}^n$

# Compare #flops for different methods

System $\boldsymbol{Ax} = \boldsymbol{b}$ of order $n = m^2$ and bandwidth $m = \sqrt{n}$.

- full Cholesky: $O(n^3)$
- band Cholesky, block tridiagonal: $O(n^2)$

If $m = 10^3$ then the computing time is

- full Cholesky: years
- band Cholesky and block tridiagonal , : hours
- Derive a method which takes : seconds

# New exact fast method

1. Diagonalization of $T$
2. Fast Sine transform

- restricted to rectangular domains
- can be extended to 9 point scheme and biharmonic problem
- can be extended to 3D

# Eigenpairs of $\boldsymbol{T} = \operatorname{diag}(-1, 2, -1) \in \mathbb{R}^{m,m}$

Let $h = 1/(m+1)$. We know that

$$\boldsymbol{T}\boldsymbol{s}_j = \lambda_j \boldsymbol{s}_j \text{ for } j = 1, \ldots, m, \text{ where}$$

$$\boldsymbol{s}_j = [\sin(j\pi h), \sin(2j\pi h), \ldots, \sin(mj\pi h)]^T,$$

$$\lambda_j = 4\sin^2\left(\frac{j\pi h}{2}\right),$$

$$\boldsymbol{s}_j^T \boldsymbol{s}_k = \frac{1}{2h}\delta_{j,k}, \quad j, k = 1, \ldots, m.$$

# The sine matrix $S$

- $\boldsymbol{S} := [\boldsymbol{s}_1, \ldots, \boldsymbol{s}_m]$, $\boldsymbol{D} = \text{diag}(\lambda_1, \ldots, \lambda_n)$,
- $\boldsymbol{T}\boldsymbol{S} = \boldsymbol{S}\boldsymbol{D}$,
- $\boldsymbol{S}^T\boldsymbol{S} = \boldsymbol{S}^2 = \frac{1}{2h}\boldsymbol{I}$,
- $\boldsymbol{S}$ is almost, but not quite orthogonal.

# Find $V$ using diagonalization

We define a matrix $\boldsymbol{X}$ by $\boldsymbol{V} = \boldsymbol{S}\boldsymbol{X}\boldsymbol{S}$, where $\boldsymbol{V}$ is the solution of $\boldsymbol{T}\boldsymbol{V} + \boldsymbol{V}\boldsymbol{T} = h^2\boldsymbol{F}$

$$\begin{aligned}
&\boldsymbol{T}\boldsymbol{V} + \boldsymbol{V}\boldsymbol{T} = h^2\boldsymbol{F} \\
\overset{\boldsymbol{V}=\boldsymbol{S}\boldsymbol{X}\boldsymbol{S}}{\iff}\; &\boldsymbol{T}\boldsymbol{S}\boldsymbol{X}\boldsymbol{S} + \boldsymbol{S}\boldsymbol{X}\boldsymbol{S}\boldsymbol{T} = h^2\boldsymbol{F} \\
\overset{\boldsymbol{S}(\ )\boldsymbol{S}}{\iff}\; &\boldsymbol{S}\boldsymbol{T}\boldsymbol{S}\boldsymbol{X}\boldsymbol{S}^2 + \boldsymbol{S}^2\boldsymbol{X}\boldsymbol{S}\boldsymbol{T}\boldsymbol{S} = h^2\boldsymbol{S}\boldsymbol{F}\boldsymbol{S} \\
\overset{\boldsymbol{T}\boldsymbol{S}=\boldsymbol{S}\boldsymbol{D}}{\iff}\; &\boldsymbol{S}^2\boldsymbol{D}\boldsymbol{X}\boldsymbol{S}^2 + \boldsymbol{S}^2\boldsymbol{X}\boldsymbol{S}^2\boldsymbol{D} = h^2\boldsymbol{S}\boldsymbol{F}\boldsymbol{S} \\
\overset{\boldsymbol{S}^2=\boldsymbol{I}/(2h)}{\iff}\; &\boldsymbol{D}\boldsymbol{X} + \boldsymbol{X}\boldsymbol{D} = 4h^4\boldsymbol{S}\boldsymbol{F}\boldsymbol{S}.
\end{aligned}$$

# $X$ is easy to find

- An equation of the form $DX + XD = B$ for some $B$ is easy to solve.
- Since $D = \mathrm{diag}(\lambda_j)$ we obtain for each entry
- $\lambda_j x_{jk} + x_{jk}\lambda_k = b_{jk}$
- so $x_{jk} = b_{jk}/(\lambda_j + \lambda_k)$ for all $j, k$.

# Algorithm

**Algorithm 1** (A Simple Fast Poisson Solver)**.**

$$\begin{aligned}
&1. \quad h = 1/(m+1); \boldsymbol{F} = \big(f(jh,kh)\big)_{j,k=1}^{m};\\
&\qquad \boldsymbol{S} = \big(\sin(jk\pi h)\big)_{j,k=1}^{m}; \boldsymbol{\sigma} = \big(\sin^2((j\pi h)/2)\big)_{j=1}^{m}\\
&2. \quad \boldsymbol{G} = (g_{j,k}) = \boldsymbol{SFS};\\
&3. \quad \boldsymbol{X} = (x_{j,k})_{j,k=1}^{m}, \text{ where } x_{j,k} = h^4 g_{j,k}/(\sigma_j + \sigma_k);\\
&4. \quad \boldsymbol{V} = \boldsymbol{SXS};
\end{aligned} \tag{1}$$

- Output is the exact solution of the discrete Poisson equation on a square computed in $O(n^{3/2})$ operations.
- Only a couple of $m \times m$ matrices are required for storage.
- Next: Use FFT to reduce the complexity to $O(n \log_2 n)$

# Discrete Sine Transform, DST

- Given $\boldsymbol{v} = [v_1, \ldots, v_m]^T \in \mathbb{R}^m$ we say that the vector $\boldsymbol{w} = [w_1, \ldots, w_m]^T$ given by

$$w_j = \sum_{k=1}^{m} \sin\left(\frac{jk\pi}{m+1}\right) v_k, \quad j = 1, \ldots, m$$

  is the **Discrete Sine Transform** (DST) of $\boldsymbol{v}$.
- In matrix form we can write the DST as the matrix times vector $\boldsymbol{w} = \boldsymbol{S}\boldsymbol{v}$, where $\boldsymbol{S}$ is the sine matrix.
- We can identify the matrix $\boldsymbol{B} = \boldsymbol{S}\boldsymbol{A}$ as the DST of $\boldsymbol{A} \in \mathbb{R}^{m,n}$, i.e. as the DST of the columns of $\boldsymbol{A}$.

# The product $B = AS$

- The product $\boldsymbol{B} = \boldsymbol{AS}$ can also be interpreted as a DST.
- Indeed, since $\boldsymbol{S}$ is symmetric we have $\boldsymbol{B} = (\boldsymbol{SA}^T)^T$ which means that $\boldsymbol{B}$ is the transpose of the DST of the rows of $\boldsymbol{A}$.
- It follows that we can compute the unknowns $\boldsymbol{V}$ in Algorithm 1 by carrying out Discrete Sine Transforms on 4 $m$-by-$m$ matrices in addition to the computation of $\boldsymbol{X}$.

# The Euler formula

$$e^{i\phi} = \cos\phi + i\sin\phi, \quad \phi \in \mathbb{R},\ i = \sqrt{-1}$$
$$e^{-i\phi} = \cos\phi - i\sin\phi$$

$$\cos\phi = \frac{e^{i\phi} + e^{-i\phi}}{2}, \quad \sin\phi = \frac{e^{i\phi} - e^{-i\phi}}{2i}$$

Consider

$$\omega_N := e^{-2\pi i/N} = \cos\left(\frac{2\pi}{N}\right) - i\sin\left(\frac{2\pi}{N}\right), \quad \omega_N^N = 1$$

Note that

$$\omega_{2m+2}^{jk} = e^{-2jk\pi i/(2m+2)} = e^{-jk\pi h i} = \cos(jk\pi h) - i\sin(jk\pi h).$$

# Discrete Fourier Transform (DFT)

If

$$z_j = \sum_{k=1}^{N} \omega_N^{(j-1)(k-1)} y_k, \quad j = 1, \ldots, N$$

then $\boldsymbol{z} = [z_1, \ldots, z_N]^T$ is the DFT of $\boldsymbol{y} = [y_1, \ldots, y_N]^T$.

$$\boldsymbol{z} = \boldsymbol{F}_N \boldsymbol{y}, \text{ where } \boldsymbol{F}_N := \left(\omega_N^{(j-1)(k-1)}\right)_{j,k=1}^{N}, \in \mathbb{R}^{N,N}$$

$\boldsymbol{F}_N$ is called the Fourier Matrix

# Example

$$\omega_4 = \exp^{-2\pi i/4} = \cos(\tfrac{\pi}{2}) - i\sin(\tfrac{\pi}{2}) = -i$$

$$\boldsymbol{F}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega_4 & \omega_4^2 & \omega_4^3 \\ 1 & \omega_4^2 & \omega_4^4 & \omega_4^6 \\ 1 & \omega_4^3 & \omega_4^6 & \omega_4^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}$$

# Connection DST and DFT

DST of order $m$ can be computed from DFT of order $N = 2m + 2$ as follows:

**Lemma 1.** *Given a positive integer $m$ and a vector $\boldsymbol{x} \in \mathbb{R}^m$. Component $k$ of $\boldsymbol{S}_m\boldsymbol{x}$ is equal to $i/2$ times component $k+1$ of $\boldsymbol{F}_{2m+2}\boldsymbol{z}$ where*

$$\boldsymbol{z} = (0, x_1, \ldots, x_m, 0, -x_m, -x_{m-1}, \ldots, -x_1)^T \in \mathbb{R}^{2m+2}.$$

*In symbols*

$$(\boldsymbol{S}_m\boldsymbol{x})_k = \frac{i}{2}\,(\boldsymbol{F}_{2m+2}\boldsymbol{z})_{k+1}\,, \quad k = 1, \ldots, m.$$

# Fast Fourier Transform (FFT)

Suppose $N$ is even. Express $\boldsymbol{F}_N$ in terms of $\boldsymbol{F}_{N/2}$. Reorder the columns in $\boldsymbol{F}_N$ so that the odd columns appear before the even ones.

$\boldsymbol{P}_N := (e_1, e_3, \ldots, e_{N-1}, e_2, e_4, \ldots, e_N)$

$$\boldsymbol{F}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \quad \boldsymbol{F}_4\boldsymbol{P}_4 = \left[\begin{array}{cc|cc} 1 & 1 & 1 & 1 \\ 1 & -1 & -i & i \\ \hline 1 & 1 & -1 & -1 \\ 1 & -1 & i & -i \end{array}\right].$$

$$\boldsymbol{F}_2 = \begin{bmatrix} 1 & 1 \\ 1 & \omega_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \boldsymbol{D}_2 = \text{diag}(1, \omega_4) = \begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix}.$$

$$\boldsymbol{F}_4\boldsymbol{P}_4 = \left[\begin{array}{c|c} \boldsymbol{F}_2 & \boldsymbol{D}_2\boldsymbol{F}_2 \\ \hline \boldsymbol{F}_2 & -\boldsymbol{D}_2\boldsymbol{F}_2 \end{array}\right]$$

# $\boldsymbol{F}_{2m}$, $\boldsymbol{P}_{2m}$, $\boldsymbol{F}_m$ and $\boldsymbol{D}_m$

**Theorem 2.** *If $N = 2m$ is even then*

$$\boldsymbol{F}_{2m}\boldsymbol{P}_{2m} = \left[\begin{array}{c|c} \boldsymbol{F}_m & \boldsymbol{D}_m\boldsymbol{F}_m \\ \hline \boldsymbol{F}_m & -\boldsymbol{D}_m\boldsymbol{F}_m \end{array}\right], \tag{2}$$

*where*

$$\boldsymbol{D}_m = \textit{diag}(1, \omega_N, \omega_N^2, \ldots, \omega_N^{m-1}). \tag{3}$$

# Proof

Fix integers $j, k$ with $0 \le j, k \le m-1$ and set $p = j+1$ and $q = k+1$. Since $\omega_m^m = 1$, $\omega_N^2 = \omega_m$, and $\omega_N^m = -1$ we find by considering elements in the four sub-blocks in turn

$$\begin{array}{lclclcl}
(\boldsymbol{F}_{2m}\boldsymbol{P}_{2m})_{p,q} & = & \omega_N^{j(2k)} & = & \omega_m^{jk} & = & (\boldsymbol{F}_m)_{p,q}, \\
(\boldsymbol{F}_{2m}\boldsymbol{P}_{2m})_{p+m,q} & = & \omega_N^{(j+m)(2k)} & = & \omega_m^{(j+m)k} & = & (\boldsymbol{F}_m)_{p,q}, \\
(\boldsymbol{F}_{2m}\boldsymbol{P}_{2m})_{p,q+m} & = & \omega_N^{j(2k+1)} & = & \omega_N^j\omega_m^{jk} & = & (\boldsymbol{D}_m\boldsymbol{F}_m)_{p,q}, \\
(\boldsymbol{F}_{2m}\boldsymbol{P}_{2m})_{p+m,q+m} & = & \omega_N^{(j+m)(2k+1)} & = & -\omega_N^{j(2k+1)} & = & (-\boldsymbol{D}_m\boldsymbol{F}_m)_{p,q}
\end{array}$$

It follows that the four $m$-by-$m$ blocks of $\boldsymbol{F}_{2m}\boldsymbol{P}_{2m}$ have the required structure.

# The basic step

- Using Theorem 2 we can carry out the DFT as a block multiplication.
- Let $\boldsymbol{y} \in \mathbb{R}^{2m}$ and set $\boldsymbol{w} = \boldsymbol{P}_{2m}^T \boldsymbol{y} = (\boldsymbol{w}_1^T, \boldsymbol{w}_2^T)^T$, where $\boldsymbol{w}_1, \boldsymbol{w}_2 \in \mathbb{R}^m$.
- Then $\boldsymbol{F}_{2m}\boldsymbol{y} = \boldsymbol{F}_{2m}\boldsymbol{P}_{2m}\boldsymbol{P}_{2m}^T\boldsymbol{y} = \boldsymbol{F}_{2m}\boldsymbol{P}_{2m}\boldsymbol{w}$ and
- $\boldsymbol{F}_{2m}\boldsymbol{P}_{2m}\boldsymbol{w} = \left[\begin{array}{c|c} \boldsymbol{F}_m & \boldsymbol{D}_m\boldsymbol{F}_m \\ \hline \boldsymbol{F}_m & -\boldsymbol{D}_m\boldsymbol{F}_m \end{array}\right] \begin{bmatrix} \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{q}_1 + \boldsymbol{q}_2 \\ \boldsymbol{q}_1 - \boldsymbol{q}_2 \end{bmatrix}$, where $\boldsymbol{q}_1 = \boldsymbol{F}_m\boldsymbol{w}_1$ and $\boldsymbol{q}_2 = \boldsymbol{D}_m(\boldsymbol{F}_m\boldsymbol{w}_2)$.
- In order to compute $\boldsymbol{F}_{2m}\boldsymbol{y}$ we need to compute $\boldsymbol{F}_m\boldsymbol{w}_1$ and $\boldsymbol{F}_m\boldsymbol{w}_2$.
- Note that $\boldsymbol{w}_1^T = [y_1, y_3, \ldots, y_{N-1}]$, while $\boldsymbol{w}_2^T = [y_2, y_4, \ldots, y_N]$.
- This follows since $\boldsymbol{w}^T = [\boldsymbol{w}_1^T, \boldsymbol{w}_2^T] = \boldsymbol{y}^T\boldsymbol{P}_{2m}$ and post multiplying a vector by $\boldsymbol{P}_{2m}$ moves odd indexed components to the left of all the even indexed components.

# Recursive FFT

Recursive Matlab function when $N = 2^k$

**Algorithm 3.** (**Recursive FFT** )

```
function z=fftrec(y)
n=length(y);
if n==1   z=y;
else
   q1=fftrec(y(1:2:n-1));
   q2=exp(-2*pi*i/n).^(0:n/2-1).*fftrec(y(2:2:n));
   z=[q1+q2 q1-q2];
end
```

Such a recursive version of FFT is useful for testing purposes, but is much too slow for large problems. A challenge for FFT code writers is to develop nonrecursive versions and also to handle efficiently the case where $N$ is not a power of two.

# FFT Complexity

- Let $x_k$ be the complexity (the number of flops) when $N = 2^k$.
- Since we need two FFT's of order $N/2 = 2^{k-1}$ and a multiplication with the diagonal matrix $\boldsymbol{D}_{N/2}$ it is reasonable to assume that $x_k = 2x_{k-1} + \gamma 2^k$ for some constant $\gamma$ independent of $k$.
- Since $x_0 = 0$ we obtain by induction on $k$ that $x_k = \gamma k 2^k = \gamma N \log_2 N$.
- This also holds when $N$ is not a power of 2.
- Reasonable implementations of FFT typically have $\gamma \approx 5$

# Improvement using FFT

- The efficiency improvement using the FFT to compute the DFT is impressive for large $N$.
- The direct multiplication $\boldsymbol{F}_N y$ requires $O(8n^2)$ flops since complex arithmetic is involved.
- Assuming that the FFT uses $5N\log_2 N$ flops we find for $N = 2^{20} \approx 10^6$ the ratio

$$\frac{8N^2}{5N\log_2 N} \approx 84000.$$

- Thus if the FFT takes one second of computing time and if the computing time is proportional to the number of flops then the direct multiplication would take something like 84000 seconds or 23 hours.

# Poisson solver based on FFT

- requires $O(n \log_2 n)$ flops, where $n = m^2$ is the size of the linear system $\boldsymbol{Ax} = \boldsymbol{b}$. Why?
- $4m$ sine transforms: $\boldsymbol{G}_1 = \boldsymbol{SF}$, $\boldsymbol{G} = \boldsymbol{G}_1\boldsymbol{S}$, $\boldsymbol{V}_1 = \boldsymbol{SX}$, $\boldsymbol{V} = \boldsymbol{V}_1\boldsymbol{S}$.
- A total of $4m$ FFT's of order $2m+2$ are needed.
- Since one FFT requires $O(\gamma(2m+2)\log_2(2m+2))$ flops the $4m$ FFT's amounts to $8\gamma m(m+1)\log_2(2m+2) \approx 8\gamma m^2 \log_2 m = 4\gamma n \log_2 n$,
- This should be compared to the $O(8n^{3/2})$ flops needed for 4 straightforward matrix multiplications with $S$.
- What is faster will depend on the programming of the FFT and the size of the problem.

# Compare exact methods

System $\boldsymbol{Ax} = \boldsymbol{b}$ of order $n = m^2$.

Assume $m = 1000$ so that $n = 10^6$.

| Method | $\#$ flops | Storage | $T = 10^{-9}$flops |
|---|---|---|---|
| Full Cholesky | $\frac{1}{3}n^3 = \frac{1}{3}10^{18}$ | $n^2 = 10^{12}$ | 10 years |
| Band Cholesky | $2n^2 = 2 \times 10^{12}$ | $n^{3/2} = 10^9$ | 1/2 hour |
| Block Tridiagonal | $2n^2 = 2 \times 10^{12}$ | $n^{3/2} = 10^9$ | 1/2 hour |
| Diagonalization | $8n^{3/2} = 8 \times 10^9$ | $n = 10^6$ | 8 sec |
| FFT | $20n \log_2 n = 4 \times 10^8$ | $n = 10^6$ | 1/2 sec |