The Conjugate Gradient Method

The minimization problem

We are given a symmetric positive definite matrix $A \in \mathbb{R}^{n \times n}$ and a right hand side vector $b \in \mathbb{R}^n$. We want to solve the linear system

$$\text{Find } u_* \in \mathbb{R}^n \text{ such that } Au_* = b.$$ 

Since $A$ is nonsingular there exists a unique solution. The linear system is equivalent to the minimization problem

$$\text{Find } u_* \in \mathbb{R}^n \text{ such that } F(u) = \frac{1}{2} u^T Au - b^T u \text{ is minimal.}$$

Note that $(u, v)_A := (Au, v) = v^T Au$ defines an inner product on $\mathbb{R}^n$, and a norm $\|u\|_A^2 = u^T Au$. Note that

$$\|u - u_*\|_A^2 = (u, u)_A - 2(u, u_*)_A + (u_*, u_*)_A = 2F(u) + \|u_*\|_A^2,$$

Hence minimizing $F(u)$ is equivalent to minimizing $\|u - u_*\|_A$.

In applications $F(u)$ corresponds to an energy (internal energy minus work of external forces) which is minimized by the solution. Therefore $\|u\|_A$ is also called the energy norm, and this norm is the “natural” way to measure errors for this problem.

Subspace corrections

Subspace correction with $\tilde{V} = \text{span}\{d_0\}$

We start at $u_0 \in \mathbb{R}^n$ and pick a search direction $d_0$. We define $u_{new} := u_0 + \alpha_0 d_0$ where $\alpha_0$ is chosen such that $\|u_{new} - u_*\|_A$ becomes minimal: The normal equations state that $(u_{new} - u_*, d_0)_A = 0$ or

$$(d_0, d_0)_A \alpha_0 = (u_* - u_0, d_0)_A = (A(u_* - u_0), d_0) = (r_0, d_0), \quad \text{hence } \alpha_0 = \frac{(d_0, r_0)}{(d_0, d_0)_A} \quad (1)$$

with the residual $r_0 := b - Au_0$. In this case we perform a subspace correction with the 1-dimensional subspace $\tilde{V} = \text{span}\{d_0\}$.

Subspace correction with $\tilde{V} = \text{span}\{d_0, \ldots, d_k\}$

We start at $u_0 \in \mathbb{R}^n$ and pick linearly independent vectors $d_0, \ldots, d_k \in \mathbb{R}^n$. We define

$$u_{new} := u_0 + \alpha_0 d_0 + \cdots + \alpha_k d_k$$

where $\alpha_0, \ldots, \alpha_k$ are chosen such that $\|u_{new} - u_*\|_A$ becomes minimal: The normal equations state that $(u_{new} - u_*, d_j)_A = 0$ for $j = 0, \ldots, k$. Therefore we can find $\alpha_0, \ldots, \alpha_k$ by solving the linear system

$$\begin{bmatrix} (d_0, d_0)_A & \cdots & (d_k, d_0)_A \\ \vdots & & \vdots \\ (d_0, d_k)_A & \cdots & (d_k, d_k)_A \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_k \end{bmatrix} = \begin{bmatrix} (r_0, d_0) \\ \vdots \\ (r_0, d_k) \end{bmatrix} \quad (2)$$

Here we used $(u_* - u_0, d_j)_A = (A(u_* - u_0), d_j) = (r_0, d_j)$ on the right hand side. Note that the normal equations state that $(A(u_{new} - u_*), d_j) = 0$ for $j = 0, \ldots, k$, hence the new residual $r_{new} := b - Au_{new} = A(u_* - u_{new})$ satisfies

$$r_{new} \perp d_0, \ldots, d_k$$
Steepest descent method

The current guess is \( u_j \). We choose the direction vector \( d_0 \) to be the **steepest descent direction** of the function \( F(u) \): The gradient is \( \nabla F(u) = Au - b \), so the steepest descent direction is given by the residual \( r_j = b - Au_j \). We define \( u_{j+1} := u_j + \alpha_j r_j \) where \( \alpha_j \) is chosen so that \( \|u_{j+1} - u_*\|_A \) becomes minimal:

For \( k = 0, 1, 2, \ldots \) do

\[
    r_k := b - Au_k, \quad \alpha_k := \frac{(r_k, r_k)}{(r_k, r_k)_A}, \quad u_{k+1} := u_k + \alpha_k r_k
\]

**Convergence:** The errors satisfy

\[
    u_{k+1} - u_* = u_k + \alpha_k A(u_* - u_k) - u_* = (I - \alpha_k A)(u_k - u_*)
\]

Since \( A \) is symmetric we have an orthonormal basis \( v_1, \ldots, v_n \) of eigenvectors with \( Av_j = \lambda_j v_j \). We write the old error \( u_k - u_* \) using this basis

\[
    u_k - u_* = \sum_{j=1}^n c_j v_j, \quad \|u_k - u_*\|_A^2 = \sum_{j=1}^n |c_j|^2 \lambda_j
\]

and get for the new error

\[
    u_{k+1} - u_* = (I - \alpha A)(u_k - u_*) = \sum_{j=1}^n c_j (1 - \alpha \lambda_j) v_j
\]

\[
    \|u_{k+1} - u_*\|_A^2 = \sum_{j=1}^n (1 - \alpha \lambda_j)^2 |c_j|^2 \lambda_j \leq \left( \max_{j=1,\ldots,n} |1 - \alpha \lambda_j| \right)^2 \|u_k - u_*\|_A^2
\]

The value of \( \alpha \) which minimizes \( \max_{j=1,\ldots,n} |1 - \alpha \lambda_j| \) is

\[
    \alpha_* = \frac{2}{\lambda_{\min} + \lambda_{\max}} \quad \text{where} \quad q_* = \max_{j=1,\ldots,n} |1 - \alpha_* \lambda_j| = \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} = \frac{\kappa - 1}{\kappa + 1} = 1 - \frac{2}{\kappa + 1} \quad \text{with} \quad \kappa := \frac{\lambda_{\max}}{\lambda_{\min}} = \cond_2(A)
\]

Since \( \alpha_k \) is chosen such that \( \|u_{k+1} - u_*\|_A \) is minimal we obtain for \( u_{k+1} = u_k + \alpha_k r_k \) the bounds

\[
    \|u_{k+1} - u_*\|_A \leq \left( 1 - \frac{2}{\kappa + 1} \right) \|u_k - u_*\|_A, \quad \|u_k - u_*\|_A \leq \left( 1 - \frac{2}{\kappa + 1} \right)^k \|u_0 - u_*\|_A
\]

This means that we need \( C \delta \kappa \) iterations to achieve \( \|u_k - u_*\|_A \leq \delta \).

Conjugate gradient method, version 0

We start with an initial guess \( u_0 \).

For \( k = 0, 1, 2, \ldots \) do

- Let \( r_k := b - Au_k \)
  
  If \( r_k = 0 \) stop (since \( u_k \) is the exact solution)

- Perform a subspace correction with \( V_k := \text{span}\{r_0, \ldots, r_k\} \):
  
  Solve (2) (with \( d_j := r_j \)), let \( u_{k+1} := u_0 + \alpha_0 r_0 + \cdots + \alpha_k r_k \)

The normal equations are in this case

\[
    \begin{pmatrix} (u_{k+1} - u_*, r_j)_A \end{pmatrix}_{j=0}^k = 0
\]

\[
    (r_{k+1}, r_j)
\]

\[ (3) \]
i.e., we have $r_{k+1} \perp V_k$. As long as we have $r_k \neq \vec{0}$ the vectors $r_0, \ldots, r_k$ are therefore linearly independent.

**Observation 1:** We will have $r_k = \vec{0}$, i.e., $u_k = u_*$ for $k \geq K$ with some $K \leq n$.

But the CG method is typically used as an iterative method with $k \ll n$ iterations to find an approximate solution $u_k$, rather than the exact solution.

**Observation 2:** We have

$$V_k = \text{span} \{ r_0, Ar_0, \ldots, A^kr_0 \}$$

(4)

**Proof:** It is obvious for $k = 0$. Assume it holds for $k - 1$. We have

$$r_k = b - Au_k = b - Au_0 - A (a_0r_0 + \cdots + a_{k-1}r_{k-1}) \in \text{span} \{ r_0, \ldots, A^{k-1}r_0 \}$$

hence $r_k \in \text{span} \{ r_0, Ar_0, \ldots, A^kr_0 \}$.

The subspace $V_k = \text{span} \{ r_0, Ar_0, \ldots, A^kr_0 \}$ is called a **Krylov space**, and it plays a central role in understanding CG, GMRES and related iterative methods.

### Conjugate gradient method, version 1

We can make the method more efficient by orthogonalizing the vectors $r_0, r_1, \ldots, r_k$ with respect to $(\cdot, \cdot)_A$ using the **Gram-Schmidt method**, yielding vectors $d_0, d_1, \ldots, d_k$ such that

$$V_k = \text{span}\{r_0, \ldots, r_k\} = \text{span}\{d_0, \ldots, d_k\}$$

and $(d_j, d_k)_A = 0$ for $j \neq k$. We let $u_{k+1} = u_0 + \alpha_0d_0 + \cdots + \alpha_kd_k$ where $\alpha_0, \ldots, \alpha_k$ are chosen such that $\|u_j - u_*\|_A = \|\alpha_0d_0 + \cdots + \alpha_kd_k - (u_* - u_0)\|_A$ is minimal. Since the directions $d_0, \ldots, d_k$ are $A$-orthogonal (a.k.a. “conjugate”) the normal equations (2) decouple:

$$(d_j, d_j)_A \alpha_j = (r_0, d_j) \quad \text{for } j = 0, \ldots, k$$

We can write the right hand side as

$$(u_* - u_0, d_j)_A = (u_* - (u_0 + \alpha_0d_0 + \cdots + \alpha_{j-1}d_{j-1}), d_j)_A = (r_j, d_j)$$

yielding $\alpha_j = \frac{(r_j, d_j)}{(d_j, d_j)_A}$ and $u_{k+1} = u_0 + \alpha_0d_0 + \cdots + \alpha_{k-1}d_{k-1} + \alpha_kd_k = u_k + \alpha_kd_k$.

Therefore the algorithm can be written as follows:

For $k = 0, 1, 2, \ldots$ do:

- the new steepest descent direction is given by the residual:

  $$r_k := b - Au_k, \quad \text{if } r_k = \vec{0}: \text{stop (since } u_k \text{ is exact solution)}$$

- modify this to make it conjugate to all previous search directions $d_{k-1}, d_{k-2}, \ldots, d_0$:

  $$d_k := \frac{(r_k, d_{k-1})_A}{(d_{k-1}, d_{k-1})_A} d_{k-1} - \frac{(r_k, d_{k-2})_A}{(d_{k-2}, d_{k-2})_A} d_{k-2} - \cdots - \frac{(r_k, d_0)_A}{(d_0, d_0)_A} d_0$$

  (5)

- perform optimal step in direction $d_k$ : (actually minimizes $\|(u_0 + \alpha_0d_0 + \cdots + \alpha_kd_k) - u_*\|_A$ over all $\alpha_0, \ldots, \alpha_k$)

  $$\alpha_k := \frac{(r_k, d_k)}{(d_k, d_k)_A}, \quad u_{k+1} := u_k + \alpha_kd_k$$

  (6)
There is one additional simplification: By (4) we have $Ad_{k-2} \in V_{k-1}$, and by the normal equations $r_k \perp V_{k-1}$:

$$
(r_k, d_{k-2})_A = (r_k, Ad_{k-2}) = 0 \in V_{k-1}
$$

By this argument all the red terms in (5) are zero and we have

$$d_k := r_k - \frac{(r_k, d_{k-1})_A}{(d_{k-1}, d_{k-1})_A}d_{k-1}$$

where we orthogonize only with respect to the previous direction $d_{k-1}$.

**Final version of the conjugate gradient method**

By (7) we have $(d_k, r_k) = (r_k, r_k)$ as $(r_k, d_{k-1}) = 0$ by the normal equations $r_k \perp V_{k-1}$. Hence

$$\alpha_k = \frac{(r_k, r_k)}{(d_k, d_k)_A}$$

We have from (6) that $\alpha_k Ad_k = r_k - r_{k+1}$ and hence

$$\alpha_k r_{k+1}^T Ad_k = r_{k+1}^T (r_k - r_{k+1}) = -r_{k+1}^T r_{k+1}$$

since $(r_k, r_{k+1}) = 0$ by the normal equations $r_{k+1} \perp V_k$. Using this in the numerator and (8) in the denominator we get

$$\frac{-r_{k+1}^T r_{k+1}}{r_k^T r_k} = \frac{\alpha_k r_{k+1}^T Ad_k}{\alpha_k d_k^T Ad_k} = \frac{(r_{k+1}, d_k)_A}{(d_k, d_k)_A}$$

so that we can write (7) for $k + 1$ as

$$\beta_k := \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}, \quad d_{k+1} := r_{k+1} + \beta_k d_k.$$

We then have the following algorithm:

$r_0 := b - Au_0$, $d_0 := r_0$.

For $k = 0, 1, 2, \ldots$ do

$$\alpha_k := \frac{r_k^T r_k}{d_k^T Ad_k}, \quad u_{k+1} := u_k + \alpha_k d_k, \quad r_{k+1} := r_k - \alpha_k (Ad_k), \quad \text{if } r_k = 0: \text{stop}$$

$$\beta_k := \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}, \quad d_{k+1} := r_{k+1} + \beta_k d_k$$

The cost of each step:

- 1 matrix vector product: compute $Ad_k$
- 2 dot products: compute $d_k^T (Ad_k)$, $r_{k+1}^T r_{k+1}$.

After we compute $r_{k+1}^T r_{k+1} = \|Au_{k+1} - b\|_2^2$ we can compare this with a given tolerance and terminate the iteration if the norm of the residual is sufficiently small.

**Error estimate:**

$$\|u_k - u_*\|_A \leq 2 \left(1 - \frac{2}{\kappa^{1/2} + 1}\right)^k \|u_0 - u_*\|_A$$

This means that we need $C_\delta \kappa^{1/2}$ iterations to achieve $\|u_k - u_*\|_A \leq \delta$. 
Proof of the error estimate for the Conjugate Gradient Method

We have \( u_k = u_0 + w \) where \( w \in V_{k-1} = \text{span} \{ r_0, Ar_0, \ldots, A^{k-1}r_0 \} \) is chosen such that \( \|u_k - u_*\|_A \) is minimal. Therefore

\[
u_k - u_* = u_0 - u_* + \sum_{j=0}^{k-1} \beta_j A^j A(u_* - u_0) = p(A)(u_* - u_0) \quad \text{with } p(\lambda) = 1 + \beta_0 \lambda + \cdots + \beta_k \lambda^k
\]

Using the eigenvectors \( v_1, \ldots, v_n \) of \( A \) we can write the initial error as \( u_0 - u_* = \sum_{j=1}^{n} c_j v_j \). Then

\[
\|u_k - u_*\|_A^2 = \sum_{j=1}^{k} p(\lambda_j)^2 |c_j|^2 \leq \left( \max_{j=1,\ldots,n} |p(\lambda_j)| \right)^2 \|u_0 - u_*\|_A^2
\]

We now try to choose a polynomial \( p(\lambda) = 1 + \beta_0 \lambda + \cdots + \beta_k \lambda^k \) which makes \( q := \max_{j=1,\ldots,n} |p(\lambda_j)| \) small. We want a polynomial \( p \in P_k \) with

\[
p(0) = 1, \quad q := \max_{\lambda \in [\lambda_1, \lambda_n]} |p(\lambda)| \text{ is small.}
\]

We can actually determine the polynomial \( p \) which minimizes \( q \). We start with the Chebyshev polynomial \( T_k(x) \) which has \( \max_{x \in [-1,1]} |T_k(x)| = 1 \). We then use a linear change of variables \( x \in [-1,1] \) to \( \lambda \in [\lambda_1, \lambda_n] \)

\[
\lambda = \frac{\lambda_1 + \lambda_n}{2} + x \frac{\lambda_n - \lambda_1}{2}, \quad x = \frac{2\lambda - \lambda_1 - \lambda_n}{\lambda_n - \lambda_1} =: g(\lambda)
\]

\[
\tilde{p}(\lambda) := T_k(g(\lambda)), \quad p(\lambda) := \frac{\tilde{p}(\lambda)}{\tilde{p}(0)}
\]

\[
q = \max_{\lambda \in [\lambda_1, \lambda_n]} |p(\lambda)| = \frac{1}{|\tilde{p}(0)|}, \quad \tilde{p}(0) = T_k \left( -\frac{\lambda_n + \lambda_1}{\lambda_n - \lambda_1} \right)
\]

Note that \( x := \frac{\lambda_n + \lambda_1}{\lambda_n - \lambda_1} = \kappa \) is minimal. We need a lower bound \( |T_k(-x)| = |T_k(x)| \geq \cdots \). Note that with \( x = \cos t = \frac{1}{2} (z + z^{-1}) \)

\[
T_k(x) = \cos (kt) = \frac{1}{2} (z^k + z^{-k})
\]

Note that for \( z := (\kappa^{1/2} + 1) / (\kappa^{1/2} - 1) \) we have

\[
\frac{1}{2} (z + z^{-1}) = \frac{1}{2} \left( (\kappa^{1/2} + 1)^2 + (\kappa^{1/2} - 1)^2 \right) / (\kappa^{1/2} + 1)(\kappa^{1/2} - 1) = \frac{\kappa + 1}{\kappa - 1}
\]

hence with \( \rho := z^{-1} = 1 - \frac{2}{\kappa^{1/2} + 1} \)

\[
T_k(x) = \frac{1}{2} (z^k + z^{-k}) = \frac{1}{2} (\rho^{-k} + \rho^k)
\]

yielding

\[
q = |T_k(x)|^{-1} = \frac{2}{\rho^{-k} + \rho^k} \leq 2 \rho^k = 2 \left( 1 - \frac{2}{\kappa^{1/2} + 1} \right)^k
\]

\[
\|u_k - u_*\|_A \leq 2 \left( 1 - \frac{2}{\kappa^{1/2} + 1} \right)^k \|u_0 - u_*\|_A
\]

Note that for \( k = 1 \) we have \( q = \frac{2}{\rho^{-1} + \rho} = \frac{2}{z + z^{-1}} = \frac{\kappa - 1}{\kappa + 1} = 1 - \frac{2}{\kappa + 1} \) which is the bound we obtained for the steepest descent method.
Preconditioned conjugate gradient method

Note that our algorithm uses two inner products on \( \mathbb{R}^n \): \((u, v)_A := v^T A u\) and \((u, v)_B := v^T B u\) with the matrix \( B := I \). For a given \( \tilde{b} \in \mathbb{R}^n \) we want to find \( u_\star \in \mathbb{R}^n \) such that \( F(u) := \frac{1}{2}(u, u)_A - (\tilde{b}, u)_B \) is minimal. This is equivalent to finding \( u_\star \in \mathbb{R}^n \) such that

\[
\forall v \in \mathbb{R}^n : \quad (u_\star, v)_A = (\tilde{b}, v)_B
\]  

(9)

For a given approximation \( u \) we can determine the residual \( \tilde{r} \) satisfying

\[
\forall v \in \mathbb{R}^n : \quad (\tilde{r}, v)_B = (\tilde{b}, v)_B - (u, v)_A
\]  

(10)

So far we considered \( B = I \). But we can generalize the conjugate gradient method for an arbitrary symmetric positive definite matrix \( B \). Note that \((u, u)_A\) can be bounded from above and below by \((u, u)_B\): There exist \( \lambda_{\text{min}}, \tilde{\lambda}_{\text{max}} \) such that

\[
\tilde{\lambda}_{\text{min}} (u, u)_B \leq (u, u)_A \leq \tilde{\lambda}_{\text{max}} (u, u)_B
\]

and the ratio

\[
\tilde{\kappa} := \frac{\tilde{\lambda}_{\text{max}}}{\tilde{\lambda}_{\text{min}}}
\]

measures how close \((u, u)_A\) and \((u, u)_B\) are to each other. Our error estimate gives for this generalized CG method

\[
\| u_k - u_\star \|_A \leq 2 \left( 1 - \frac{2}{\tilde{\kappa}^{1/2} + 1} \right)^k \| u_0 - u_\star \|_A
\]

This means that we need \( C_\delta \tilde{\kappa}^{1/2} \) iterations to achieve \( \| u_k - u_\star \|_A \leq \delta \).

For many problems the original CG method performs poorly because \((u, u)_A\) and \((u, u)_B\) are “far apart”, i.e., the condition number \( \kappa = \frac{\lambda_{\text{max}}}{\lambda_{\text{min}}} \) is large. The idea of the preconditioned conjugate gradient method (PCG) is to use a symmetric positive definite matrix \( B \) such that

- The bilinear form \((\cdot, \cdot)_B\) is “close” to the bilinear form \((\cdot, \cdot)_A\):

\[
\tilde{\lambda}_{\text{min}} (u, u)_B \leq (u, u)_A \leq \tilde{\lambda}_{\text{max}} (u, u)_B \quad \text{with} \quad \tilde{\kappa} := \frac{\tilde{\lambda}_{\text{max}}}{\lambda_{\text{min}}} \ll \kappa
\]

- It is “easy” to find \( B^{-1} v \), i.e., to solve a linear system \( B u = v \)

Note that (9), (10) give

\[
Au_\star = B\tilde{b} \quad \text{i.e.,} \quad \tilde{b} = B^{-1} b
\]

\[
B\tilde{r} = B\tilde{b} - Au \quad \text{i.e.,} \quad \tilde{r} = B^{-1} (b - Au)
\]

So we must be able to compute \( B^{-1} v \) in order to implement the PCG.

Let \( r_k := b - A x_k \). We then have \((x, v)_A - (\tilde{b}, v)_B = (\tilde{r}, v)_B\) with \( \tilde{r}_k := B^{-1} r_k \), and \((\tilde{r}_k, \tilde{r}_k)_B = (B\tilde{r}_k, \tilde{r}_k) = (r_k, \tilde{r}_k)\).

Therefore we obtain the algorithm for the preconditioned conjugate gradient method:

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r_0 := b - A x_0 ), ( \tilde{r}_0 := B^{-1} r_0 ), ( d_0 := \tilde{r}_0 ).</td>
<td>For ( k = 0, 1, 2, \ldots ) do</td>
</tr>
<tr>
<td>( \alpha_k := \frac{r_k^T \tilde{r}<em>k}{d_k^T (A d_k)} ), ( u</em>{k+1} := u_k + \alpha_k d_k ), ( r_{k+1} := r_k - \alpha_k (A d_k) ), ( \tilde{r}<em>{k+1} := B^{-1} r</em>{k+1} )</td>
<td></td>
</tr>
</tbody>
</table>

The cost of each step:
• 1 matrix vector product: compute $A d_k$

• 2 dot products: compute $d_k^\top (A d_k), r_{k+1}^\top \tilde{r}_{k+1}$

• 1 solution of $B u = v$: compute $B^{-1} r_{k+1}$

Error estimate:

$$\|u_k - u_*\|_A \leq 2 \left(1 - \frac{2}{\kappa^{1/2} + 1}\right)^k \|u_0 - u_*\|_A$$

Choosing a preconditioner

Choosing a good preconditioner $B$ can make a huge difference and enable us to solve much larger problems. The preconditioner matrix $B$ should somehow “mimic” the matrix $A$, but $B v = f$ should be much cheaper to solve that $A v = f$.

There is no universal recipe for a good preconditioner. This depends on the orginal problem which the matrix $A$ describes (e.g., elliptic PDE). We give three examples for choosing a preconditioner:

1. **Incomplete Cholesky decomposition**: For a sparse matrix $A$ we can compute the Cholesky decomposition $A = C^\top C$: use $C = \text{chol}(A)$ in Matlab. The matrix $C$ has typically many more nonzeros than the matrix $A$ (so-called “fill-in”), and this makes the direct solver expensive. We can “cheat” in the Cholesky algorithm and only allow a limited amount of fill-in: this is the incomplete Cholesky decomposition. It gives an upper triangular matrix $\tilde{C}$, but $\tilde{C}^\top \tilde{C}$ is not equal to $A$. In Matlab use $Ct = \text{ichol}(A)$ (there are options to specify how much fill-in to allow).

   Then use $B = \tilde{C}^\top \tilde{C}$: We can solve $B v = f$ as follows: solve $\tilde{C}^\top y = f$ by forward substitution, then solve $\tilde{C} v = y$ by back substitution.

2. **Use “similar problem” for which there is a fast algorithm**

   Example: Solve elliptic PDE with nonconstant coefficient $c(x)$ on square $\Omega = (0,1)^2$

   $$-\text{div} (c(x) \nabla u) = f(x)$$

   using a uniform mesh.

   For $c(x) = 1$ we can use FFT to solve this problem: In 1D use discrete sine transform. In 2D, 3D apply this for each direction. Since the discrete problem is translation invariant for a uniform mesh: for the new basis the matrix becomes diagonal. Hence the work is $C n \log n$.

3. **Multigrid**: diffusion problem in $\mathbb{R}^d$, uniform mesh refinement: e.g. bisect all edges, $h_{\ell+1} = \frac{1}{2} h_\ell$, gives nested spaces

   $$V_{h_0} \subset V_{h_1} \subset \cdots \subset V_{h_L}, \quad N_\ell := \dim V_{h_\ell}$$

   Use nodal basis. Consider function $v_{h_\ell} \in V_{h_\ell}$ with nodal values $\tilde{v}(\ell)$. This function is in $V_{h_{\ell+1}}$ with nodal values

   $$\tilde{v}^{(\ell+1)} := J_\ell \tilde{v}(\ell), \quad J_\ell \in \mathbb{R}^{N_{\ell+1} \times N_\ell}$$

   Define the preconditioning operator $M_\ell = B^{-1}$ as follows: “BPX preconditioner”

   $$M_\ell f := \begin{cases} I & \text{if } \ell = 0 \\ 2^{(d-2)\ell} I + J_{\ell-1} M_{\ell-1} J_{\ell-1}^\top & \text{otherwise} \end{cases}$$