5 On eigenvalue problems for PDEs

Consider the one-dimensional wave equation

$$\frac{\partial^2 U}{\partial t^2} - c^2 \frac{\partial^2 U}{\partial x^2} = F(x,t). \tag{40}$$

Let us assume that f is of the form $F(x,t) = f(x)e^{-\mathrm{i}\omega t}$, with given $\omega \in \mathbb{R}$.
We seek the solution in form $U(x,t) = \mathrm{Re}\left(u(x)e^{-\mathrm{i}\omega t}\right)$. Then, inserting this into (40) we obtain

$$-\omega^2 u(x)e^{-\mathrm{i}\omega t} - c^2 \frac{\partial^2 u(x)}{\partial x^2} e^{-\mathrm{i}\omega t} = f(x)e^{-\mathrm{i}\omega t} \tag{41}$$

Dividing (41) by $e^{-\mathrm{i}\omega t}$ we obtain *Helmholtz equation*

$$-c^2 \frac{\partial^2 u}{\partial x^2} - \omega^2 u = f. \tag{42}$$

In case of *free vibration* $f \equiv 0$, we again search the solution in the form $U(x,t) = \mathrm{Re}\left(u(x)e^{-\mathrm{i}\omega t}\right)$ but now ω is an *unknown* parameter.
Denoting $\lambda := (\omega/c)^2$ we obtain the following eigenvalue problem for differential equation

$$-\frac{\partial^2 u}{\partial x^2} = \lambda u.$$

Example 5.1. Consider the following eigenvalue problem

$$\begin{cases} -u''(x) = \lambda u & 0 < x < 1 \\ u(0) = u(1) = 0. \end{cases} \tag{43}$$

Its analytical solution is

$$\begin{cases} \lambda_j = (j\pi)^2, & j = 1, 2, \dots \\ u_j(x) = \sin(j\pi x), & j = 1, 2, \dots \end{cases}$$

Problem (43) can be approximately solved by using finite difference method, i.e.

$$-\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = \lambda u_i, \quad i = 1, \dots, n, \quad h = \frac{1}{n+1}.$$

This is an algebraic eigenvalue problem and it can be written in matrix form

$$\frac{1}{h^2} \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & & \ddots & & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} = \lambda \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} \tag{44}$$

The exact eigenvalues of the algebraic problem (44) are

$$\lambda_j^h = 2(n+1)^2 - 2(n+1)^2 \cos\left(\frac{j\pi}{n+1}\right), \quad j = 1, \dots, n.$$

Thus we have the following error estimate

$$\begin{aligned}\lambda_j - \lambda_j^h &= (j\pi)^2 - 2(n+1)^2 + 2(n+1)^2\left[1 - \frac{1}{2}\left(\frac{j\pi}{n+1}\right)^2 + \frac{1}{24}\left(\frac{j\pi}{n+1}\right)^4 + \mathcal{O}\left(\frac{j\pi}{n+1}\right)^6\right] \\ &= \frac{1}{12}\lambda_j^2 h^2 + \mathcal{O}(h^4)\end{aligned}$$

Note that the smallest eigenvalues are approximated better than bigger ones. No approximation is available for higher modes $j > n$.
In the general case, the algebraic eigenvalue problem must be solved numerically too.

6 Fast solution of the discrete Poisson equation

A solution algorithm for $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}$, where $\boldsymbol{A} \in \mathbb{R}^{N\times N}$, $u \in \mathbb{R}^N$, $\boldsymbol{f} \in \mathbb{R}^N$, is said *fast* if its computational complexity is $\mathcal{O}(N \log N)$ (or better).

6.1 Multigrid methods

The numerical solution of the discrete Poisson problem leads to the solution of a large and sparse system

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}. \tag{45}$$

If we have an approximate solution vector $\hat{\boldsymbol{u}} \approx \boldsymbol{u}$, then the error vector $\boldsymbol{e} := \boldsymbol{u} - \hat{\boldsymbol{u}}$ can be computed from

$$\boldsymbol{A}\boldsymbol{e} = \boldsymbol{r},$$

where $\boldsymbol{r} := \boldsymbol{f} - \boldsymbol{A}\hat{\boldsymbol{u}}$ (residual). Then we obtain the exact solution by

$$\boldsymbol{u} = \hat{\boldsymbol{u}} + \boldsymbol{e} = \hat{\boldsymbol{u}} + \boldsymbol{A}^{-1}\boldsymbol{r}.$$

Now, if we have cheap approximation $\boldsymbol{B}$ to $\boldsymbol{A}^{-1}$, we can improve the approximation $\hat{\boldsymbol{u}}$ by

$$\bar{\boldsymbol{u}} \leftarrow \hat{\boldsymbol{u}} + \boldsymbol{B}\boldsymbol{r}.$$

Next we present one way to construct such $\boldsymbol{B}$.

Consider one-dimensional Poisson problem

$$-u''(x) = f(x), \quad 0 < x < 1; \quad u(0) = u(1) = 0.$$

After finite difference discretization we obtain a linear algebraic system

$$\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & & \ddots & & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix} = h^2 \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n-1} \\ f_n \end{bmatrix}. \tag{46}$$

Let us apply a single classical Jacobi iteration to system (46):

$$u_i^{(k+1)} = \frac{1}{2}\left(h^2 f_i + u_{i-1}^{(k)} + u_{i+1}^{(k)}\right), \quad i = 1, ..., n.$$

If we compare this to the exact solution to (46)

$$u_i = \frac{1}{2}\left(h^2 f_i + u_{i-1} + u_{i+1}\right)$$

the error $e_i^{(k)} := u_i - u_i^{(k)}$ reads

$$e_i^{(k+1)} = \frac{1}{2}\left(e_{i-1}^{(k)} + e_{i+1}^{(k)}\right).$$

Thus, Jacobi iteration merely *smooths* the error, i.e. it reduces the high frequence components of the error. On the other hand, if we restrict a smooth error component into a coarser grid, it

appears more oscillating. Thus performing Jacobi iteration on a coarser grid reduces lower frequency error components.
The idea of reducing different frequency error components on different grids forms the basis of the multigrid method.
Consider one iteration of a *two-grid* method:

$$\begin{aligned}
&\boldsymbol{u}_h = \text{current approximation} && \\
&\boldsymbol{u}_h = \text{Jac}(\boldsymbol{f}_h, \boldsymbol{u}_h) && \text{one Jacobi iteration} \\
&\boldsymbol{r}_h = \boldsymbol{f}_h - \boldsymbol{A}\boldsymbol{u}_h && \text{residual} \\
&\boldsymbol{r}_H = \boldsymbol{I}_h^H \boldsymbol{r}_h && \text{restrict residual into coarse grid} \\
&\boldsymbol{A}_H \boldsymbol{e}_H = \boldsymbol{r}_H && \text{solve error on coarse grid} \\
&\boldsymbol{e}_h = \boldsymbol{I}_H^h \boldsymbol{e}_H && \text{interpolate error to fine grid} \\
&\boldsymbol{u}_h = \boldsymbol{u}_h + \boldsymbol{e}_h && \text{correction}
\end{aligned} \tag{47}$$

The steps can be combined into a single matrix-vector product

$$\boldsymbol{u}_h = \boldsymbol{u}_h + \underbrace{\boldsymbol{I}_H^h \boldsymbol{A}_H^{-1} \boldsymbol{I}_h^H}_{=:\boldsymbol{B}} \boldsymbol{r}_h. \tag{48}$$

Thus we have a cheap approximation of $\boldsymbol{A}^{-1}$ as $\boldsymbol{A}_H$ corresponds to a problem discretized on a coarse grid.
The same idea can now be applied recursively to (47) resulting a *multigrid* method.
Multigrid methods are very efficient. Some variants are optimal in terms of computational complexity requiring $\mathcal{O}(N)$ arithmetic operations where N is the number of unknowns.

6.2 Methods based on separation of variables

The *tensor product* of matrices $\boldsymbol{A} \in \mathbb{R}^{m_1 \times n_1}$, $\boldsymbol{B} \in \mathbb{R}^{m_2 \times n_2}$ is defined by

$$\boldsymbol{A} \otimes \boldsymbol{B} := \begin{bmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \dots a_{1n_1}\boldsymbol{B} \\ \vdots & & \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & \dots a_{2n_1}\boldsymbol{B} \\ a_{m_1 1}\boldsymbol{B} & a_{m_1 2}\boldsymbol{B} & \dots a_{m_1 n_1}\boldsymbol{B} \end{bmatrix} \in \mathbb{R}^{m_1 m_2 \times n_1 n_2}. \tag{49}$$

The tensor product has the properties

$$(\boldsymbol{A} \otimes \boldsymbol{B})(\boldsymbol{C} \otimes \boldsymbol{D}) = \boldsymbol{AC} \otimes \boldsymbol{BD} \tag{50}$$

$$(\boldsymbol{A} \otimes \boldsymbol{B})^{-1} = \boldsymbol{A}^{-1} \otimes \boldsymbol{B}^{-1}. \tag{51}$$

Let us assume a two-dimensional Poisson problem discretized in a uniform grid of $N = n^2$ unknowns. Moreover, we assume natural numbering of the unknowns by grid rows (see eq. (16)). The discrete problem can then represented in the form

$$(\boldsymbol{T} \otimes \boldsymbol{I} + \boldsymbol{I} \otimes \boldsymbol{T})\boldsymbol{u} = \boldsymbol{f}, \tag{52}$$

where $\boldsymbol{T} = \text{tridiag}\{-1, 2, -1\} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{I} \in \mathbb{R}^{n \times n}$ is the identity matrix, and $\boldsymbol{f} = [h^2 f_1, ..., h^2 f_N]^T$.
Let $\boldsymbol{\Lambda}$ be a diagonal matrix containing the eigenvalues of $\boldsymbol{T}$ and let matrix $\boldsymbol{W}$ contain the orthonormal eigenvectors as its columns. Then $\boldsymbol{W}^{\mathrm{T}}\boldsymbol{T}\boldsymbol{W} = \boldsymbol{\Lambda}$ and $\boldsymbol{W}^{\mathrm{T}}\boldsymbol{W} = \boldsymbol{I}$.

Multiplying equation (52) from left by $\boldsymbol{W}^{\mathrm{T}} \otimes \boldsymbol{I}$ and denoting $\boldsymbol{u} := (\boldsymbol{W} \otimes \boldsymbol{I})\boldsymbol{v}$ we obtain

$$(\boldsymbol{W}^{\mathrm{T}} \otimes \boldsymbol{I})(\boldsymbol{T} \otimes \boldsymbol{I} + \boldsymbol{I} \otimes \boldsymbol{T})(\boldsymbol{W} \otimes \boldsymbol{I})\boldsymbol{v} = (\boldsymbol{W}^{\mathrm{T}} \otimes \boldsymbol{I})\boldsymbol{f}.$$

Using (50) we get after some manipulation

$$(\boldsymbol{\Lambda} \otimes \boldsymbol{I} + \boldsymbol{I} \otimes \boldsymbol{T})\boldsymbol{v} = (\boldsymbol{W}^{\mathrm{T}} \otimes \boldsymbol{I})\boldsymbol{f} =: \hat{\boldsymbol{f}}. \tag{53}$$

Let us write (53) in block form:

$$\left(\begin{bmatrix} \lambda_1 \boldsymbol{I} & & & \\ & \lambda_2 \boldsymbol{I} & & \\ & & \ddots & \\ & & & \lambda_n \boldsymbol{I} \end{bmatrix} + \begin{bmatrix} \boldsymbol{T} & & & \\ & \boldsymbol{T} & & \\ & & \ddots & \\ & & & \boldsymbol{T} \end{bmatrix} \right) \begin{bmatrix} \boldsymbol{v}^{(1)} \\ \boldsymbol{v}^{(2)} \\ \vdots \\ \boldsymbol{v}^{(n)} \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{f}}^{(1)} \\ \hat{\boldsymbol{f}}^{(2)} \\ \vdots \\ \hat{\boldsymbol{f}}^{(n)} \end{bmatrix}. \tag{54}$$

The nodal values of the modified Poisson equation can be computed by rows by solving n independent tridiagonal systems

$$(\boldsymbol{T} + \lambda_j \boldsymbol{I})\boldsymbol{v}^{(j)} = \hat{f}^{(j)}, \quad j = 1, \ldots, n.$$

The cost of a single tridiagonal solution is $\mathcal{O}(n)$.
Matrix-vector products $\hat{\boldsymbol{f}} = (\boldsymbol{W}^{\mathrm{T}} \otimes \boldsymbol{I})\boldsymbol{f}$ and $\boldsymbol{u} = (\boldsymbol{W} \otimes \boldsymbol{I})\boldsymbol{v}$ can be evaluated using the discrete sine transformation. As

$$\hat{f}_j^{(l)} = \sum_{k=1}^{n} w_k^{(l)} f_j^{(k)}, \quad w_k^{(l)} = \sin\left(\frac{kl\pi}{n+1}\right)$$

we see that $\hat{\boldsymbol{f}}$ can be evaluated by columns by applying each column of $\boldsymbol{f}$ the discrete sine transform. Similarly, the columns of $\boldsymbol{u}$ are obtained by applying each column of $\boldsymbol{v}$ the discrete inverse sine transform.
The discrete sine transform (and its inverse) can be computed with $\mathcal{O}(n \log n)$ arithmetic operations using the fast Fourier transformation (FFT).
The total number of arithmetic operations to solve (52) equals

$$n \cdot \mathcal{O}(n) + 2n \cdot \mathcal{O}(n \log n) = \mathcal{O}(n^2 \log n) = \mathcal{O}(N \log \sqrt{N}).$$