1 Poisson Equation

Our first boundary value problem will be the steady-state heat equation, which in two dimensions has the form

$$-\left(\frac{\partial}{\partial x}k\frac{\partial T}{\partial x}+\frac{\partial}{\partial y}k\frac{\partial T}{\partial y}\right)=q'''(\mathbf{x}),\qquad\text{plus BCs.}\tag{1}$$

If the thermal conductivity $k>0$ is constant, we can pull it outside of the partial derivatives and divide both sides by k to yield the 2D Poisson equation

$$-\left(\frac{\partial^2\tilde{u}}{\partial x^2}+\frac{\partial^2\tilde{u}}{\partial y^2}\right)=f(\mathbf{x}),\qquad\text{plus BCs.}\tag{2}$$

with $f:=q'''/k$ and $\tilde{u}:=T$. (We typically use u for scalar fields throughout the course for notational convenience.)

The short-hand notation for the Poisson equation (2) is

$$-\nabla^2\tilde{u}\quad=\quad f(\mathbf{x})\text{ in }\Omega,\qquad\tilde{u}=\tilde{u}_b\text{ on }\partial\Omega_D,\ \nabla\tilde{u}\cdot\hat{\mathbf{n}}=g\text{ on }\partial\Omega_N,$$

which applies in any number of space dimensions d. Here, we've indicated a mixture of boundary conditions: Dirichlet on $\partial\Omega_D$, where u is prescribed, and Neumann on $\partial\Omega_N$, where the normal component of the gradient is prescribed. We take $\hat{\mathbf{n}}$ to be the outward pointing normal on the domain boundary $\partial\Omega=\partial\Omega_D\bigcup\partial\Omega_N$.

We will work with the Poisson equation and extensions throughout the course. At this point we want to introduce some simple cases in order to understand optimal solution strategies in the 3D case, which is arguably the most important in terms of compute cycles consumed throughout the world.

1.1 Finite Differences in 1D

The basic idea behind the finite difference approach to solving differential equations is to replace the differential operator with difference operators at a set of n gridpoints. In 1D, it is natural to order the points sequentially, as illustrated in Fig. 1. Here, we consider the two-point boundary value problem

$$-\frac{d^2\tilde{u}}{dx^2}\quad=\quad f(x),\qquad\tilde{u}(0)=\tilde{u}(1)=0.\tag{3}$$

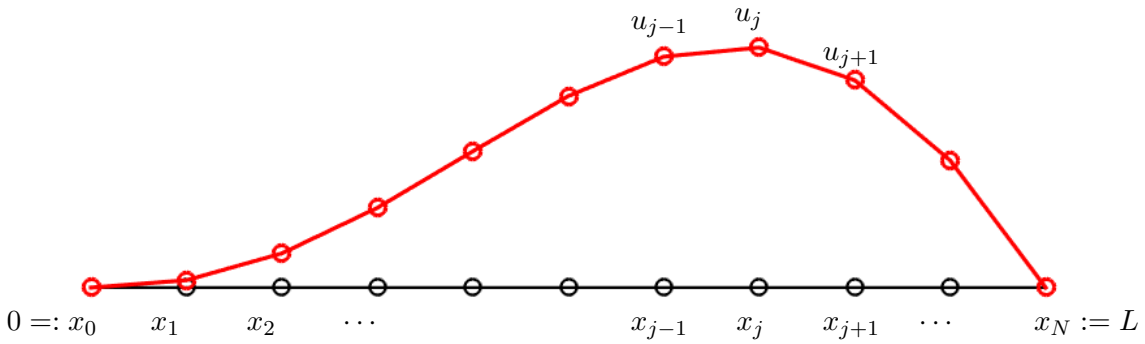


Figure 1: Finite difference grid on $\Omega:=[0,L]$ with grid-spacing $\Delta x=L/N$.

We use the second-order centered difference approximation to the second derivative, which for uniform grid spacing $x_j - x_{j-1} = \Delta x = L/(n+1)$ is,

$$-\frac{\tilde{u}_{j+1} - 2\tilde{u}_j + \tilde{u}_{j-1}}{\Delta x^2} = -\left(\left.\frac{d^2\tilde{u}}{dx^2}\right|_{x_j} + \frac{\Delta x^2}{12}\left.\frac{d^4\tilde{u}}{dx^4}\right|_{x_j} + O(\Delta x^4) \right) \tag{4}$$

$$= f(x_j) - \frac{\Delta x^2}{12}\left.\frac{d^4\tilde{u}}{dx^4}\right|_{x_j} - O(\Delta x^4) \tag{5}$$

$$\approx f(x_j). \tag{6}$$

Note that (4)–(6) comprises three steps. First, we use the Taylor series expansion to express our difference formula in terms of the desired derivative plus higher-order corrections. Second, we use the differential equation to replace the desired derivative by the data ($f_j := f(x_j)$). Finally, we incur truncation error by dropping the higher-order terms in the Taylor series. We equate the results to arrive at a discete system for the *numerical approximation*, u_j,

$$-\frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta x^2} = f_j, \qquad j = 1, \ldots, n. \tag{7}$$

The set of equations (7) represents our discretization of the original differential equation and is an algebraic system consisting of n equations in n unknowns, u_j, j=1,...,n. Each equation j relates u_{j-1}, u_j, and u_{j+1} to f_j. For this reason, the resulting matrix system is *tridiagonal*,

$$\underbrace{\frac{1}{\Delta x^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{pmatrix}}_{A_x} \underbrace{\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_n \end{pmatrix}}_{\underline{u}} = \underbrace{\begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_n \end{pmatrix}}_{\underline{f}}, \tag{8}$$

which has the shorthand $A_x\underline{u} = \underline{f}$, where $\underline{u}$ is the vector of unknowns and $\underline{f}$ the vector of data values, as indicated in (8). Note that, because we have two Dirichlet conditions, we have $n = (N+1) - 2 = N - 1$ unknowns. We will often simply use A (or, sometimes $\hat{A}$) instead of A_x for notational convenience. Here, we explicitly call out the x-coordinate association in preparation for the 2D development coming in the next section.

We list several attributes of $A = A_x$ that carry over to higher space dimensions.

- A is *symmetric*, which implies it has real eigenvalues and an orthonormal set of eigenvectors satisfying $A\underline{s}_j = \lambda_j \underline{s}_j$, $\underline{s}_j^T \underline{s}_i = \delta_{ij}$, where the Kronecker δ_{ij} equals 1 when $i = j$ and 0 when $i \neq j$.
- A is also *positive definite*, which means that $\underline{x}^T A\underline{x} > 0$ for all $\underline{x} \neq 0$. It also implies $\lambda_j > 0$. Symmetric positive definite (SPD) systems are particularly attractive because they can be solved without pivoting using Cholesky factorization, $A = LL^T$, or iteratively using preconditioned conjugate gradient (PCG) iteration. (For large sparse systems in space dimension $d \geq 3$, PCG is typically the best option.)
- A is *sparse*. It has a fixed maximal number of nonzeros per row (3, in the case of A_x), which implies that the total number of nonzeros in A is linear in the problem size, n. We say that the storage cost for A is $O(n)$, meaning that there exists a constant C independent of n such that the total number of words to be stored is $< Cn$.
- A is *banded* with bandwidth $w = 1$, which implies that $a_{ij} = 0$ for all $|i - j| > w$. A consequence is that the storage bound for the Cholesky factor L is $< (w + 1)n$. For the 1D case with w=1, the storage for L is thus $O(n)$. As we shall see, the work to compute the factors is $O(w^2 n)$.
- A^{-1} is completely full. We know this from the physics of the problem. Consider $\underline{f} \equiv 0$ save for one point, where $f_j = 1$ (i.e., f_j is the nth column of the $n \times n$ identity matrix). This case corresponds to a point heat source at x_j and, as we know, the temperature will be nonzero everywhere except at the endpoints. In fact, it will exhibit a linear decay from x_j to the boundaries. This is the exact Green's function for both the continuous and the discrete case. It's an easy exercise to show that, for any matrix $A = (\underline{a}_1\, \underline{a}_2\, \ldots\, \underline{a}_n)$ we have $\underline{a}_j = A\underline{e}_j$ when $\underline{e}_j$ is the jth column of I. The preceding arguments establish that A^{-1} must be completely full.

1.2 Eigenvalues in $\mathbb{R}^1$

One of the most attractive features of (constant coefficient, uniformly-spaced) finite differences is that we have closed-form expressions for their eigenfunctions and eigenvalues. To start, we return to our original BVP (3) and consider the associated eigenvalue problem,

$$-\frac{d^2\tilde{u}}{dx^2} = \tilde{\lambda}\tilde{u}(x), \qquad \tilde{u}(0) = \tilde{u}(L) = 0. \tag{9}$$

The solutions to (9) are $\tilde{u} = \sin k\pi x/L$ and the eigenvalues are $\tilde{\lambda}_k = k^2\pi^2/L^2$.

The discrete counterpart to (9) is

$$A\underline{u} = \lambda\underline{u}. \tag{10}$$

Remarkably, the eigenvectors for the finite difference (and linear finite element) method with uniform grid spacing are the same as their continuous counterparts. That is, they are sine functions with wavenumber k. If we denote the kth eigenvector of A by $\underline{z}_k$, then its jth component is

$$(\underline{z}_k)_j = \sin k\pi x_j/L, \qquad x_j = j\Delta x = jL/(n+1). \tag{11}$$

To find the associated eigenvalues, we apply A to $\underline{z}_k$. Let $\underline{u} = \underline{z}_k$ and $\underline{w} = A\underline{u}$. Then

$$\begin{aligned} w_j &= -\frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta x^2} \\ &= -\frac{1}{\Delta x^2}\left[\sin(k\pi x_{j+1}/L) - 2\sin(k\pi x_j/L) + \sin(k\pi x_{j-1}/L)\right]. \end{aligned} \tag{12}$$

At this point, we invoke a trigonometric identity,

$$\sin(a+b) + \sin(a-b) = 2\sin(a)\cos(b), \tag{13}$$

and note that $x_{j\pm1} = x_j \pm \Delta x$. Using these two results (12) becomes,

$$\begin{aligned} w_j &= -\frac{1}{\Delta x^2}\left[2\sin(k\pi x_j/L)\cos(k\pi\Delta x/L) - 2\sin(k\pi x_j/L)\right] \\ &= \frac{2}{\Delta x^2}\left[1 - \cos(k\pi\Delta x/L)\right]\sin(k\pi x_j/L) \\ &= \frac{2}{\Delta x^2}\left[1 - \cos(k\pi\Delta x/L)\right]u_j \\ &= \lambda_k\, u_j. \end{aligned} \tag{14}$$

We have thus established that the eigenvalues of A for the 1D case are

$$\begin{aligned} \lambda_k &= \frac{2}{\Delta x^2}\left[1 - \cos(k\pi\Delta x/L)\right] \\ &= \frac{2}{\Delta x^2}\left[1 - \cos(\pi k/N)\right] \\ &= \frac{4}{\Delta x^2}\sin^2(\pi k/2N) \end{aligned} \tag{15}$$

Like their continuous counterpart, the discrete eigenvalues λ_k are positive, which implies that A is positive definite.

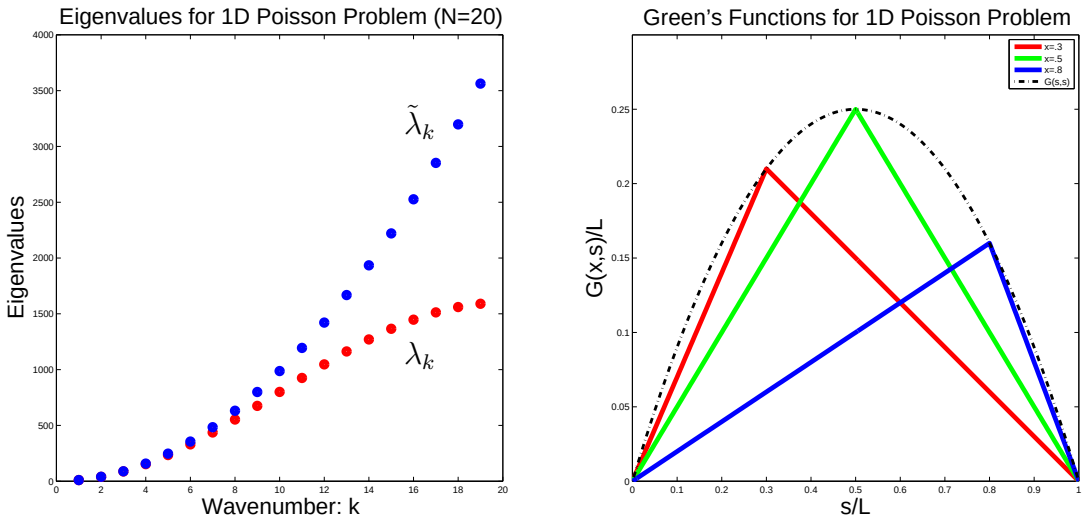


Figure 2: Left: eigenvalues for the continuous and discrete problems (9) and (10). Right: Green's functions (76) for several values of x. The dashed curve is the locus of Green's function maxima.

We are typically interested in the extreme ends of the spectrum, that is, the smallest and largest (in magnitude) eigenvalues. For small values of $k\Delta x$, we have $\lambda_k = \tilde{\lambda}_k + O(k^2\Delta x^2)$ as can be seen from a Taylor series expansion for (15).

$$\begin{aligned} \lambda_k &= \frac{2}{\Delta x^2}\left[1 - \left(1 - \frac{k^2\pi^2\Delta x^2}{2!L^2} + \frac{k^4\pi^4\Delta x^4}{4!L^4} + \text{h.o.t.}\right)\right] \\ &= \frac{k^2\pi^2}{L^2}\left(1 - \frac{\pi^2(k\Delta x)^2}{12L^2} + \text{h.o.t.}\right). \end{aligned} \tag{16}$$

The smaller eigenvalues ($k\Delta x \ll L$) are quite close to their physical counterparts, $k^2\pi^2/L^2$. On the other hand, as $k \longrightarrow n$, we have from (15) $\lambda_k \longrightarrow 4/\Delta x^2 = 4N^2/L^2$ and $\tilde{\lambda}_k \longrightarrow \pi^2 n^2$. In summary, for low wavenumbers k, we have

$$\lambda_k \sim \tilde{\lambda}_k = k^2\pi^2/L^2. \tag{17}$$

For large wavenumbers, we have $\lambda_k \sim 4/\Delta x^2 \sim 4n^2/L^2$. We plot $\tilde{\lambda}_k$ and λ_k in Fig. 2. We remark that the ratio $\tilde{\lambda}_n/\lambda_n \sim \pi^2/4$ is bounded, independent of n, which is not the case for the advection operator as we shall see later in the course. We also note that the lower eigenvalues of the continuous Poisson problem are well-approximated by virtually every worthy numerical method (i.e., as in (17)).

2 Poisson Equation in $\mathbb{R}^2$

Our principal concern at this point is to understand the (typical) matrix structure that arises from the 2D Poisson equation and, more importantly, its 3D counterpart. The essential features of this structure will be similar for other discretizations (i.e., FEM, SEM), other PDEs, and other space dimensions, so there is merit to starting with this relatively simple system.

The steady-state heat equation in two dimensions is:

$$-\nabla \cdot k\nabla T = q'''(x,y), \qquad \text{plus BCs.} \tag{18}$$

For constant thermal conductivity k this equation reduces to the standard Poisson equation in $\Omega := [0,1]^2 \subset \mathbb{R}^2$, which we usually express in terms of u for notational convenience:

$$\begin{aligned} -\nabla^2 u &= f(x,y), \qquad \text{plus BCs} \\ &= -\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \\ &= -\left(\frac{\delta^2 u}{\delta x^2} + \frac{\delta^2 u}{\delta y^2}\right) + O(h^2), \end{aligned} \tag{19}$$

where we have substituted the finite difference approximations, assumed to be about the point $\mathbf{x}_{ij} := (x_i, y_j)$,

$$\begin{aligned} \frac{\delta^2 u}{\delta x^2} &:= \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} \\ \frac{\delta^2 u}{\delta y^2} &:= \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2}, \end{aligned} \tag{20}$$

with the further assumption of uniform grid spacing, $\Delta x = \Delta y = h$. We'll also consider homogeneous Dirichlet boundary conditions, that is, $u(x,y)|_{\partial\Omega} \equiv 0$. The respective unknowns and data in this case are u_{ij} and f_{ij}, governed by the following system of equations

$$-\left(\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2}\right) = f_{ij}, \tag{21}$$

for $i, j \in [1, \ldots, n_x] \times [1, \ldots, n_y]$.

A picture of the 5-point "stencil" representing the interactions implied by (21) is shown below and a typical grid with equal spacing $\Delta x = \Delta y$ is shown in Fig. 4.

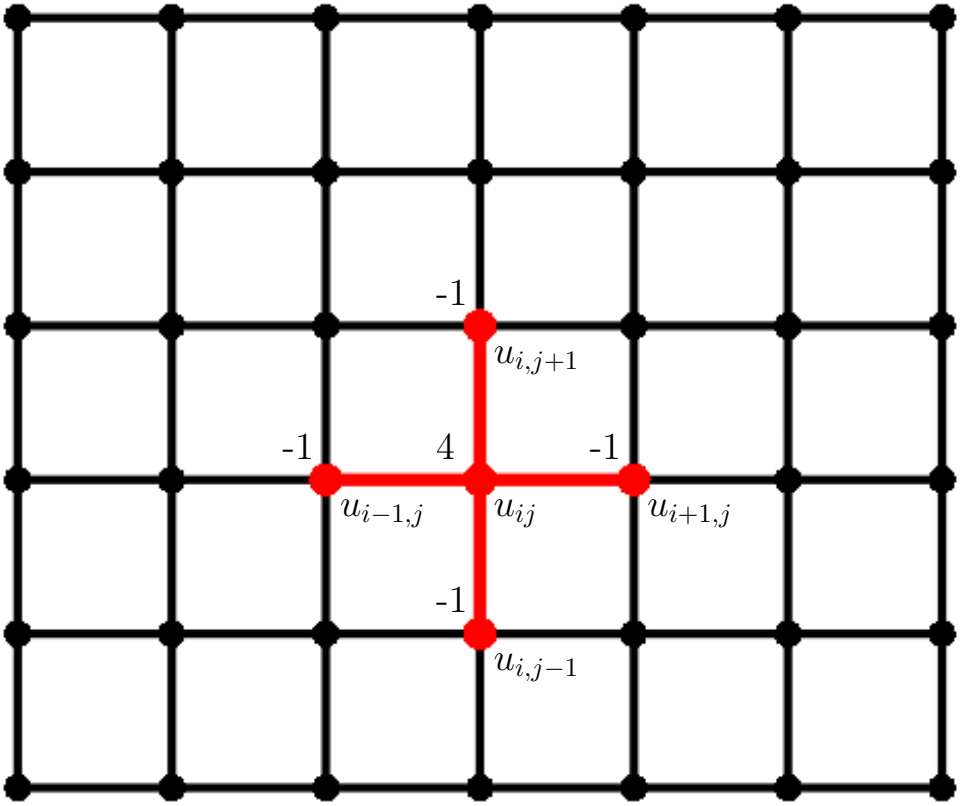


Figure 3: 5-point stencil associated with gridpoint (x_i, y_j) indicates how (21) links u_{ij} to its neighboring values. Other (e.g., 9-point) stencils are also possible.

$$-\left(\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{\Delta x^2}+\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{\Delta y^2}\right) \;=\; f_{ij},$$

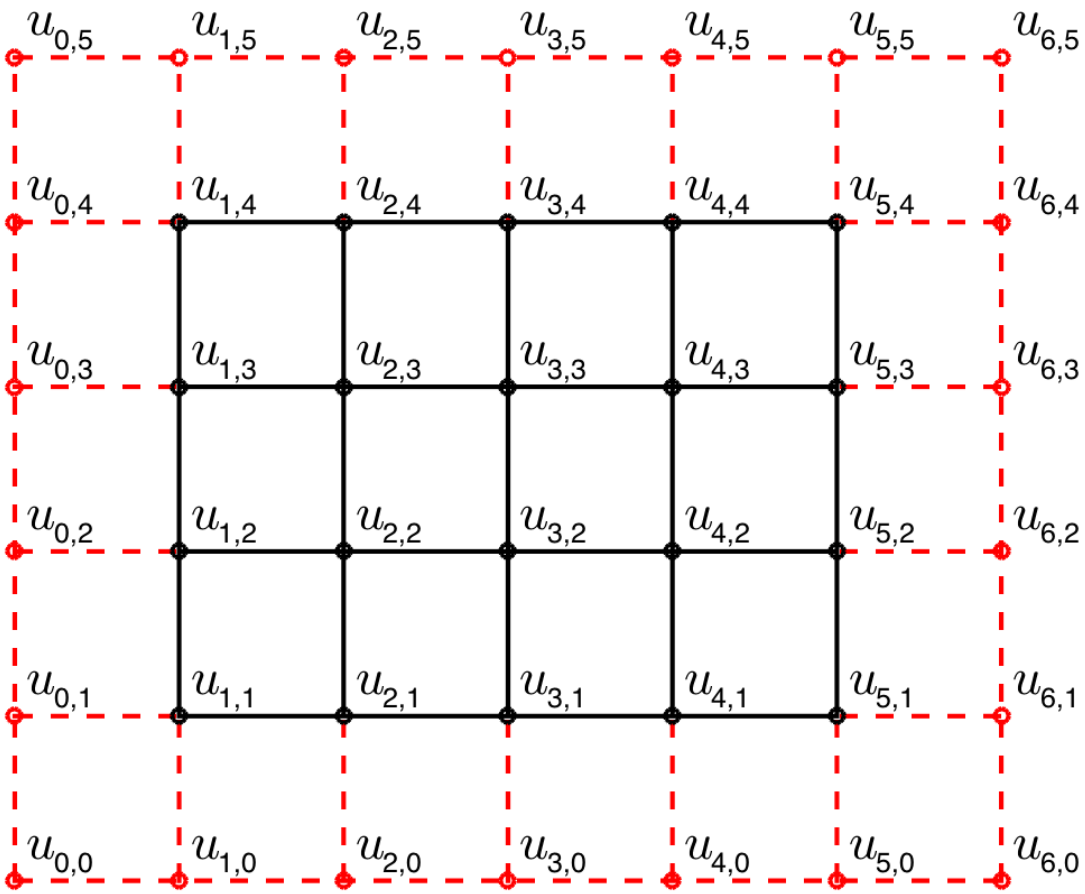


Figure 4: Finite-difference grid for 2D Poisson problem with $N_x = 6$ and $N_y = 5$. The black portion of the grid shows the actual degrees-of-freedom, whereas the red indicates known boundary values. Note that $n_x = N_x - 1$ and $n_y = N_y - 1$.

Assuming a **lexicographical ordering** in which the i- (x-) index advances fastest, the system takes on the following matrix structure for $\Delta x = \Delta y = h$.

$$\frac{1}{h^2}\underbrace{\left(\begin{array}{cccc|cccc|cccc|cccc}
4 & -1 & & & -1 & & & & & & & & & & & \\
-1 & 4 & -1 & & & -1 & & & & & & & & & & \\
 & -1 & \ddots & \ddots & & & \ddots & & & & & & & & & \\
 & & \ddots & \ddots & -1 & & & \ddots & & & & & & & & \\
 & & & -1 & 4 & & & & -1 & & & & & & & \\
\hline
-1 & & & & 4 & -1 & & & \ddots & & & & & & & \\
 & -1 & & & -1 & 4 & -1 & & & \ddots & & & & & & \\
 & & \ddots & & & -1 & \ddots & \ddots & & & \ddots & & & & & \\
 & & & \ddots & & & \ddots & \ddots & -1 & & & \ddots & & & & \\
 & & & & -1 & & & -1 & 4 & & & & \ddots & & & \\
\hline
 & & & & \ddots & & & & \ddots & & & & -1 & & & \\
 & & & & & \ddots & & & & \ddots & & & & -1 & & \\
 & & & & & & \ddots & & & & \ddots & & & & \ddots & \\
 & & & & & & & \ddots & & & & \ddots & & & & -1 \\
 & & & & & & & & \ddots & & & & \ddots & & & \\
\hline
 & & & & & & & & -1 & & & & 4 & -1 & & \\
 & & & & & & & & & -1 & & & -1 & 4 & \ddots & \\
 & & & & & & & & & & \ddots & & & \ddots & \ddots & -1 \\
 & & & & & & & & & & & -1 & & & -1 & 4
\end{array}\right)}_{A_{2D}}
\underbrace{\left(\begin{array}{c}
u_{11} \\ u_{21} \\ \vdots \\ \vdots \\ u_{n_x 1} \\ \hline u_{12} \\ u_{22} \\ \vdots \\ \vdots \\ u_{n_x 2} \\ \hline \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \hline u_{1n_y} \\ u_{2n_y} \\ \vdots \\ \vdots \\ u_{n_x n_y}
\end{array}\right)}_{\underline{u}}
=
\underbrace{\left(\begin{array}{c}
f_{11} \\ f_{21} \\ \vdots \\ \vdots \\ f_{n_x 1} \\ \hline f_{12} \\ f_{22} \\ \vdots \\ \vdots \\ f_{n_x 2} \\ \hline \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \hline f_{1n_y} \\ f_{2n_y} \\ \vdots \\ \vdots \\ f_{n_x n_y}
\end{array}\right)}_{\underline{f}}$$

We see that u_{11} interacts with u_{21} by virtue of the -1s on the first superdiagonal and with u_{12} through the -1s on the second nontrivial superdiagonal, which is $+n_x$ columns away from the main diagonal.

Q: What is the matrix bandwidth in this case?

Q: Assuming that $n_x = n_y$ and $n = n_x \times n_y$, what is the matrix bandwidth in terms of n?

Q: How many nonzeros will be in the LU factors, L and U?

Q: What is the cost of solving $L\underline{v} = \underline{f}$? and $U\underline{u} = \underline{v}$?

Q: What is the cost of computing the LU factors?

It is instructive to note that A_{2D} can be expressed as the sum of two systems, one associated with A_x coming from $\frac{\delta^2 u}{\delta x^2}$, and one associated with one associated with A_y coming from $\frac{\delta^2 u}{\delta y^2}$. Specifically, we can write

$$A_{2D} \;=\; (I_y \otimes A_x) \;+\; (A_y \otimes I_x), \tag{22}$$

where we introduce the Kronecker (or *tensor*) product, $\otimes$.

For two matrices A and B, their Kronecker product $C = A \otimes B$ is defined as the block matrix

$$C \;:=\; \begin{pmatrix} a_{11}B & a_{12}B & \cdots & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & \cdots & a_{2n}B \\ \vdots & \vdots & & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & \cdots & a_{mn}B \end{pmatrix}. \tag{23}$$

We will soon explore a few properties of this form, but for now simply note that it allows a clean expression of the discretized Poisson operator in 2D. Consider the following splitting of A_{2D}.

The decomposition

$$A_{2D} \;=\; (I_y \otimes A_x) \;+\; (A_y \otimes I_x), \tag{24}$$

has the following explicit form.

$$A_{2D} = \frac{1}{h^2}\left(\begin{array}{c|c|c|c}
\begin{array}{ccccc} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{array} & & & \\ \hline
& \begin{array}{ccccc} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{array} & & \\ \hline
& & \begin{array}{ccccc} \ddots & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & \ddots \end{array} & \\ \hline
& & & \begin{array}{ccccc} 2 & -1 & & & \\ -1 & 2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{array}
\end{array}\right)$$

$$+ \frac{1}{h^2}\left(\begin{array}{c|c|c|c}
\begin{array}{ccccc} 2 & & & & \\ & 2 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & 2 \end{array} & \begin{array}{ccccc} -1 & & & & \\ & -1 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & -1 \end{array} & & \\ \hline
\begin{array}{ccccc} -1 & & & & \\ & -1 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & -1 \end{array} & \begin{array}{ccccc} 2 & & & & \\ & 2 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & 2 \end{array} & \begin{array}{ccccc} \ddots & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & \ddots \end{array} & \\ \hline
& \begin{array}{ccccc} \ddots & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & \ddots \end{array} & \begin{array}{ccccc} \ddots & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & \ddots \end{array} & \begin{array}{ccccc} -1 & & & & \\ & -1 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & -1 \end{array} \\ \hline
& & \begin{array}{ccccc} -1 & & & & \\ & -1 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & -1 \end{array} & \begin{array}{ccccc} 2 & & & & \\ & 2 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & 2 \end{array}
\end{array}\right)$$

$$
\begin{aligned}
A_{2D} &= \begin{pmatrix} A_x & & & \\ & A_x & & \\ & & \ddots & \\ & & & A_x \end{pmatrix} + \frac{1}{h^2} \begin{pmatrix} 2I_x & -I_x & & \\ -I_x & 2I_x & \ddots & \\ & \ddots & \ddots & -I_x \\ & & -I_x & 2I_x \end{pmatrix} \\
&= (I_y \otimes A_x) + (A_y \otimes I_x)
\end{aligned} \tag{25}
$$

We see that we can express A_{2D} as $(I_y \otimes A_x) + (A_y \otimes I_x)$. The first term is nothing other than $-\frac{\delta^2}{\delta x^2}$ being applied to each row (j) of u_{ij} and the second term amounts to applying $-\frac{\delta^2}{\delta y^2}$ to each column (i) on the grid.

Note that our finite-difference stiffness matrix in matlab would be written as

```
A = kron(Iy,Ax) + kron(Ay,Ix)
```

where `Ax` and `Ay` are formed using the matlab `spdiags` command (`help spdiags`), and Iy and Ix are formed using `speye`.

It is important to use *sparse matrices* in matlab for these higher-dimensional (2D and 3D) problems or you will run out of memory and it will take *very long* to solve these problems.

Even with sparse matrices, the solve times will be long. This problem is known in scientific computing and *the curse of dimensionality*.

- Note that our lexicographical ordering

$$\underline{u} = [u_{11}\ u_{21}\ \cdots\ u_{n_x n_y}]^T$$

  is in effect nothing other than an association of an array of indices, $[i, j]$ to a position in memory.

- We could also view this ordering as assigning id's to the vertices of our original graph, as illustrated below.

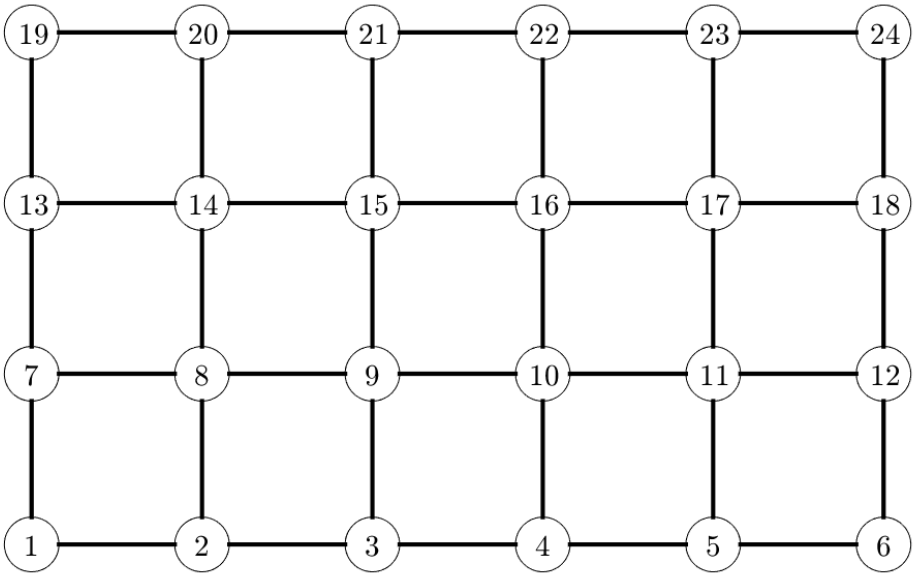


Figure 5: Lexicographical assignment of vertex id's to nodes of a finite-difference graph.

- **Q:** What is the bandwidth of A_{2D} in this case?

- **Q:** What is the bandwidth of A_{2D} in *this* case?

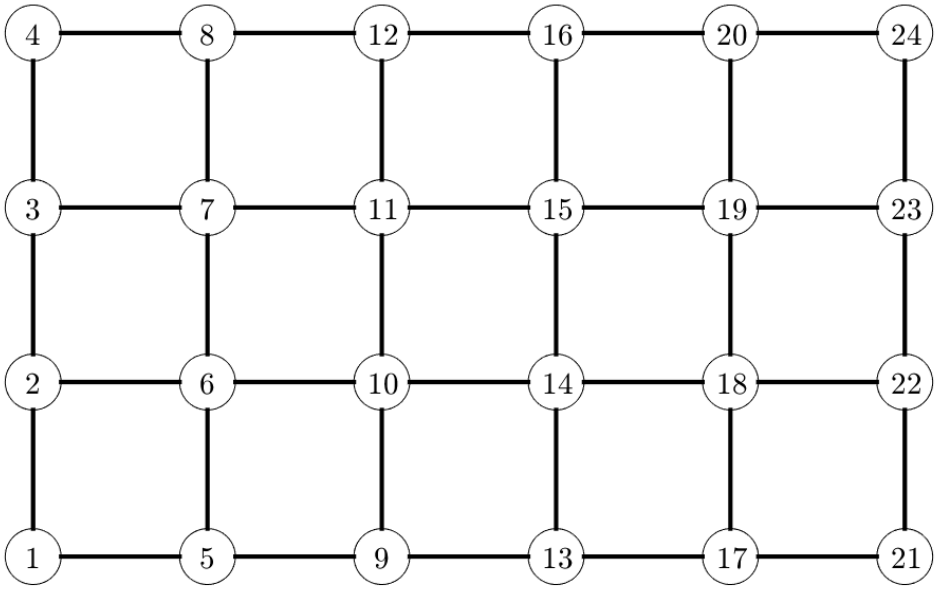


Figure 6: Lexicographical assignment of vertex id's to nodes of a finite-difference graph.

- It is important to note that the graphs of these meshes are isomorphic to the fill pattern of the corresponding sparse matrix.

- To see this, associate each node (or vertex) i in the graph with the diagonal entry, a_{ii}.

- Off-diagonal entries, a_{ij}, are zero if vertex i is not connected to vertex j, and nonzero if i is connected to j.

- For this reason, graph partitioning and graph re-ordering is an important topic for sparse matrix solution.

- Consider the example below from Saad.

- What is the associated sparse matrix (just the pattern)?

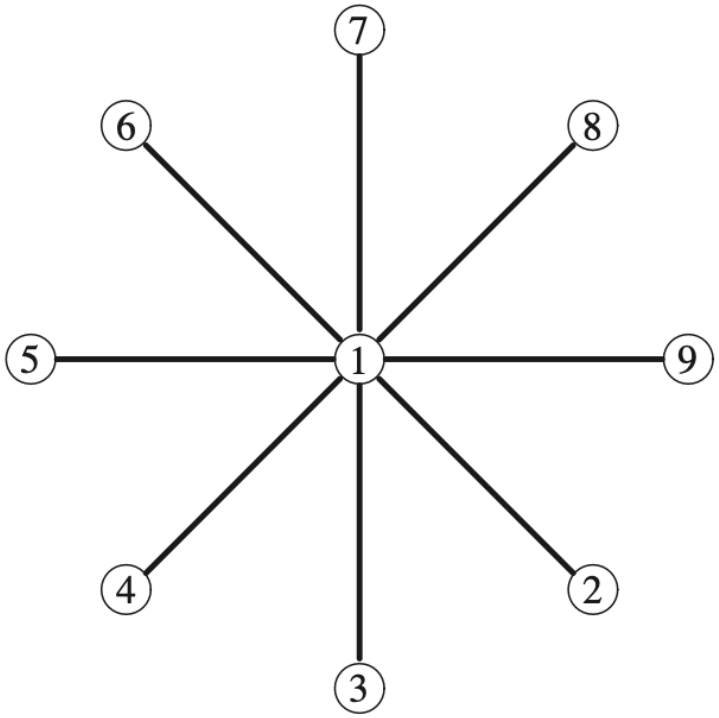


Figure 7: Star graph 1.

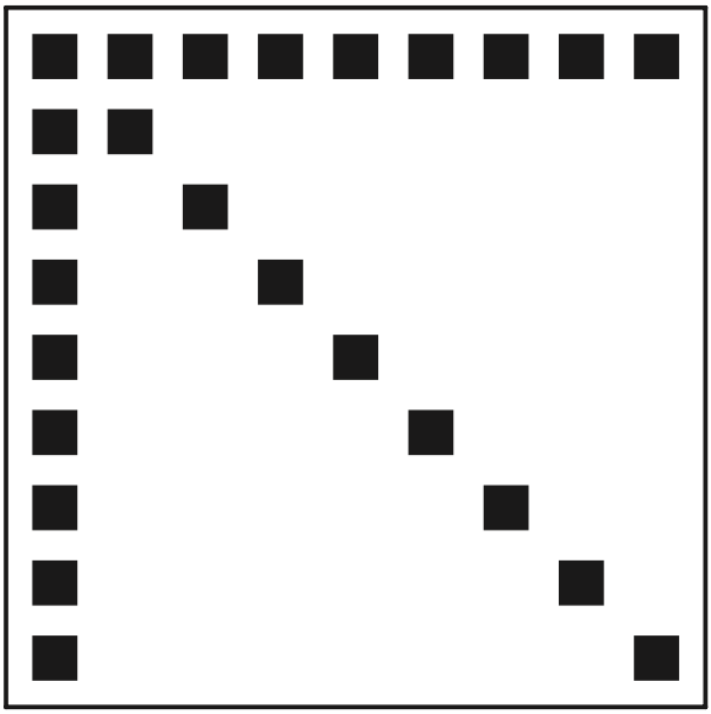


Figure 8: Arrow matrix 1.

- **Q:** What happens if we perform LU in this case?

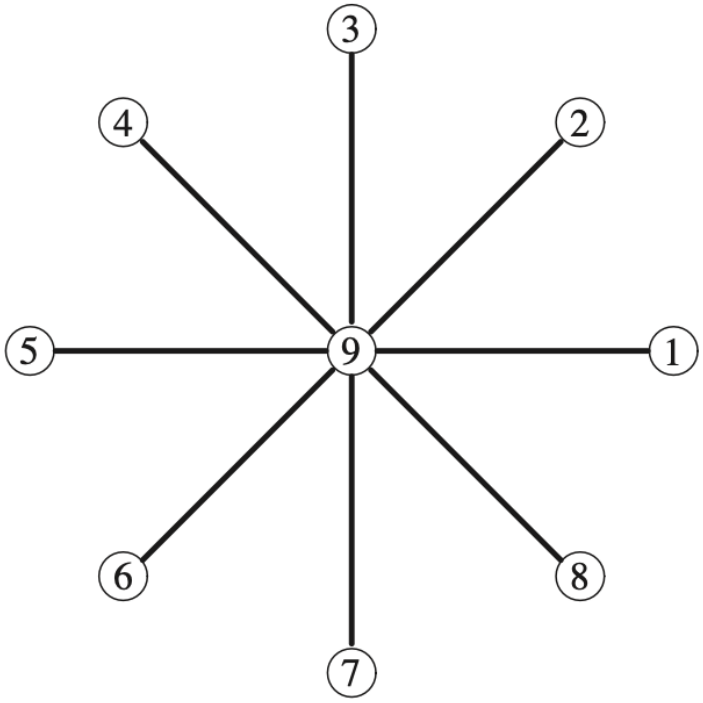


Figure 9: Star graph 2.

- **Q:** What is the associated sparse matrix in this case?

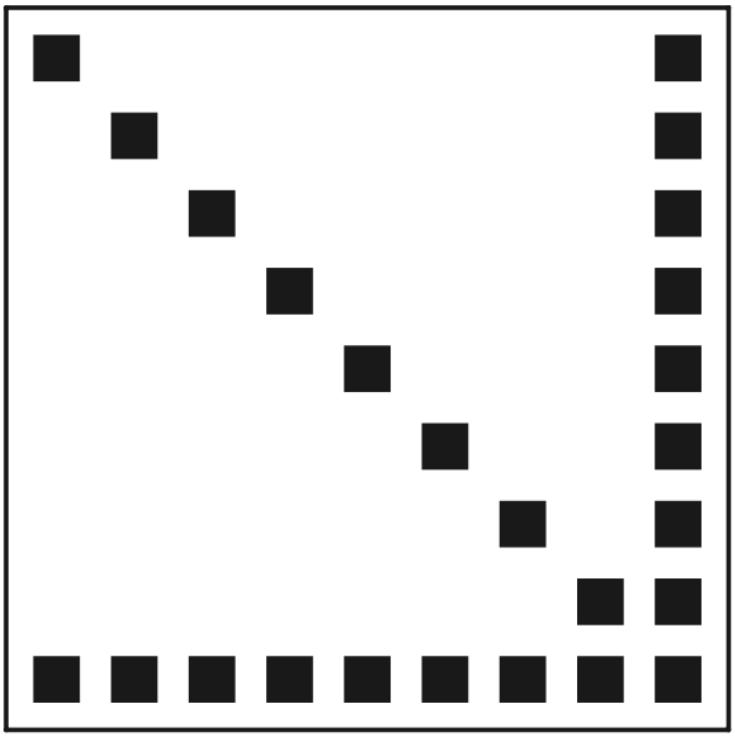


Figure 10: Arrow matrix 2.

- **Q:** What happens if we perform LU in this case?

- In general, ordering weakly connected vertices first is a good idea.

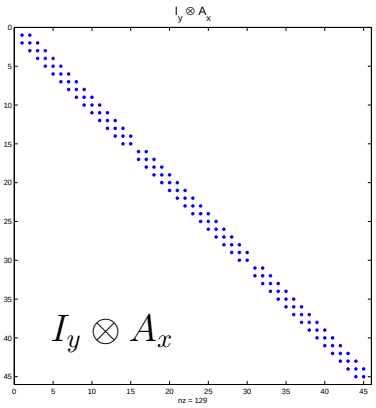
- Matlab supports multiple options, including *symmetric approximate minimum degree* (`p=symamd(A)`), which is quite effective in reducing fill in the factors L and U.

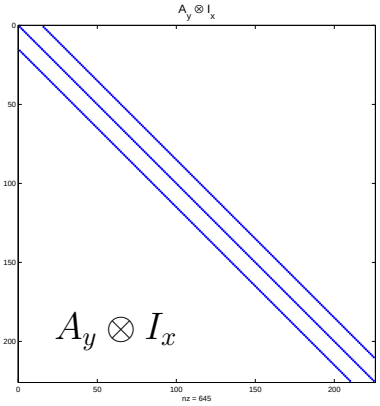
- Let's look at a few demos on the consequence of reording in our original graph.

  `cs556/notes/nest/test3.m`

2.1 Matlab Kronecker Product Demos

```
close all; format compact;
% Kronecker Product Demo
%
%    NOTE:  It is important to use SPARSE matrices throughout.
%
%           Otherwise, your run times will be very long and
%           you will likely run out of memory!

Lx=2; Ly=1;
nx=15; ny=3;  % Number of _interior_ points

dx=Lx/(nx+1); dy=Ly/(ny+1);

%  USE help spdiags

e = ones(nx,1); Ax = spdiags([-e 2*e -e], -1:1, nx, nx)/(dx*dx);
e = ones(ny,1); Ay = spdiags([-e 2*e -e], -1:1, ny, ny)/(dy*dy);

Ix=speye(nx); Iy=speye(ny);

A = kron(Iy,Ax) + kron(Ay,Ix);   %%% FINITE DIFFERENCE STIFFNESS MATRIX

% A couple of demo cases without the 1/(dx*dx) scaling.

nd= 5;
e = ones(nd,1); Ad = spdiags([-e 2*e -e], -1:1, nd, nd);
T = kron(Iy,Ad); full(T)

nd= 15;
e = ones(nd,1); Ad = spdiags([-e 2*e -e], -1:1, nd, nd);
T = kron(Iy,Ad); spy(T)
title('I_y \otimes A_x','fontsize',16)
set(gcf,'PaperUnits','normalized');set(gcf,'PaperPosition',[0 0 1 1])
print -dpdf iyax.pdf

pause; figure
nd= 5;
e = ones(nd,1); Ad = spdiags([-e 2*e -e], -1:1, nd, nd);
T = kron(Ad,Ix); full(T)

nd= 15;
e = ones(nd,1); Ad = spdiags([-e 2*e -e], -1:1, nd, nd);
T = kron(Ad,Ix); spy(T)
title('A_y \otimes I_x','fontsize',16)
set(gcf,'PaperUnits','normalized');set(gcf,'PaperPosition',[0 0 1 1])
print -dpdf ayix.pdf
```





2.2 Poisson Equation in $\mathbb{R}^3$

We now extend the 1D and 2D concepts to the most important 3D case. The short story is that the 3D stiffness matrix has the beautifully symmetric form

$$\begin{aligned} A_{3D} &= (I_z \otimes A_{2D}) + (A_z \otimes I_{2D}) \\ &= (I_z \otimes I_y \otimes A_x) + (I_z \otimes A_y \otimes I_x) + (A_z \otimes I_y \otimes I_x). \end{aligned} \tag{26}$$

and the discrete system is, as before, $A_{3D}\underline{u} = \underline{f}$. This of course is the form that arises for a finite difference discretization of $-\nabla^2 u = f$ in $\Omega = [0,1]^3$, $u = 0$ on $\partial\Omega$, or, more explicitly,

$$-\left(\frac{\delta^2 u}{\delta x^2} + \frac{\delta^2 u}{\delta y^2} + \frac{\delta^2 u}{\delta z^2}\right) = f(x_i, y_j, z_k), \tag{27}$$

with

$$\left.\frac{\delta^2 u}{\delta z^2}\right|_{ijk} := \frac{u_{ij,k+1} - 2u_{ijk} + u_{ij,k-1}}{\Delta z^2}, \tag{28}$$

and equivalent expressions for $\frac{\delta^2 u}{\delta x^2}$ and $\frac{\delta^2 u}{\delta y^2}$.

Note that, with (26), we really have not restricted ourselves to uniform mesh spacing in each direction. We could have different grid spacing Δx, Δy, and Δz and a different number of mesh points n_x, n_y, and n_z, in each of the space directions. Then, I_x would be the $n_x \times n_x$ identity matrix and A_x would be the corresponding stiffness matrix, as would also be the case for y and z. (Note also that we could even relax the condition of uniform spacing in *each* of the space directions.)

Quiz:

- Assume that we solve the Poisson problem on $\Omega = [0, L]^d$ with grid spacing $h = L/N$ in each of d space dimensions, $d = 2$ or 3. If the unknowns are ordered lexicographically, what is the *matrix bandwidth* of the system matrix A for the case $d = 2$?
- For $d = 3$?
- What is the number of unknowns, n for each case, $d = 2$ and 3?

3 Kronecker Product Matrix Properties

Kronecker products have several useful properties in the the solution of PDEs and (more recently) in machine learning, which is driving the development of specialized software and hardware for Kronecker product application and manipulation. (Unfortunately, much of this development is targeting 16-bit operations, which is not sufficient for most scientific computing applications.) There are two main properties of interest, the matrix-product rule and matrix-vector products.

For, completeness, we recall that the the Kronecker product of two matrices A and B is defined as the block matrix

$$C \;=\; A \otimes B \;:=\; \begin{pmatrix} a_{11}B & a_{12}B & \cdots & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & \cdots & a_{2n}B \\ \vdots & \vdots & & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & \cdots & a_{mn}B \end{pmatrix}. \tag{29}$$

We note that if A and B are each $N \times N$ matrices, then C is a much larger $N^2 \times N^2$ matrix, with N^4 nonzeros in the case when A and B are full. We also note that $I \otimes B$ and $A \otimes I$ have a special structure.

$$I \otimes B \;=\; \begin{pmatrix} B & & & \\ & B & & \\ & & \ddots & \\ & & & B \end{pmatrix}, \tag{30}$$

$$A \otimes I \;=\; \begin{pmatrix} a_{11}I & a_{12}I & \cdots & a_{1n}I \\ a_{21}I & a_{22}I & \cdots & a_{2n}I \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}I & a_{m2}I & \cdots & a_{mn}I \end{pmatrix}. \tag{31}$$

Thus, $I \otimes B$ is block-diagonal, whereas $A \otimes I$ comprises diagonal blocks.

The most important property of Kronecker products for the purposes of PDEs is the matrix-product rule. Suppose we have C and F satisfying

$$C \;=\; A \otimes B, \qquad F \;=\; D \otimes E.$$

Then, straightforward application of (29) reveals that the matrix product CF is given by

$$CF \;=\; (A \otimes B)(D \otimes E) \;=\; AD \otimes BE, \tag{32}$$

under the assumption that the dimensions of (A,D) and (B,E) are such that the products AD and BE make sense.

The product rule (32) leads to several notable properties that we now describe. To simplify the exposition, we'll assume that A and B are $N \times N$ matrices unless otherwise noted.

- **Inverse.** The inverse of $C = A \otimes B$ is $C^{-1} = A^{-1} \otimes B^{-1}$. Thus, the inverse of an $N^2 \times N^2$ matrix can be found by inverting (or, more typically, factoring) two much smaller $N \times N$ matrices.

- **Eigenvalues.** Let $C = B_y \otimes B_x$ and suppose that there exists a diagonalization of B_x and B_y given, for B_x, by $S_x^{-1} B_x S_x = \Lambda_x$, with S_x the matrix of eigenvectors and Λ_x the diagonal matrix containing, with a similar form for B_y. Then

$$C = B_y \otimes B_x = (S_y \Lambda_y S_y^{-1}) \otimes (S_x \Lambda_x S_x^{-1}) \tag{33}$$

$$= \underbrace{(S_y \otimes S_x)}_{S} \underbrace{(\Lambda_y \otimes \Lambda_x)}_{\Lambda} \underbrace{(S_y^{-1} \otimes S_x^{-1})}_{S^{-1}}. \tag{34}$$

  Thus, the $N^2 \times N^2$ eigenvalue problem $CS = S\Lambda$ is solved by solving two small $(N \times N)$ eigenvalue problems, which lead to $S = S_y \otimes S_x$ and $\Lambda = \Lambda_y \otimes \Lambda_x$.

- **Matrix-Vector Product.** Suppose $S = S_y \otimes S_x$ and we wish to evaluate the matrix-vector product $\underline{w} = S\underline{f}$, where $\underline{f}$ is assumed to have a natural *dual-subscript* ordering, $\{f_{ij}\}$, as in Fig. 4. If we evaluate $\underline{w}$ by first forming S, then the storage and work rises sharply from $O(N^2)$ to $O(N^4)$. Instead, we evaluate the product in *factored form*,

$$\underline{w} \;=\; (S_y \otimes I_x)\,(I_y \otimes S_x)\underline{f}. \tag{35}$$

The first evaluation generates the vector $\underline{v} := (I_y \otimes S_x)\underline{f}$, which has entries

$$v_{ij} \;=\; \sum_{p=1}^{n_x} (S_x)_{ip} f_{pj}, \quad i \in [1,\ldots,n_x],\; j \in [1,\ldots,n_y]. \tag{36}$$

which can be seen by inspecting the figure preceding Eq. (22). Assuming S_x is full, the cost of computing $\underline{v}$ is $2n_x$ for each of the $n_y n_x$ entries, or $O(N^3)$.

The next step is evaluation of $\underline{w} = (S_y \otimes I_x)\underline{v}$.

$$w_{ij} \;=\; \sum_{q=1}^{n_y} (S_y)_{jq} v_{iq}, \tag{37}$$

which has cost $2n_y^2 n_x$.

Note that *significant performance gains* (up to an order-of-magnitude) can be realized by recognizing that the doubly-indexed vectors, $\underline{w}$, $\underline{v}$, and $\underline{f}$ can be viewed as corresponding $n_y \times n_x$ matrices, W, V, and F. In this case, the *matrix-vector product* evaluation (35) can be expressed in *matrix-matrix* product form:

$$W = S_x F S_y^T. \tag{38}$$

Or, in general, for any $\underline{v} = (A \otimes B)\underline{u}$, we have $V = B\,U\,A^T$. Matrix-matrix products are some of the most efficient operations possible in numerical computation because the require only $O(N^2)$ memory references for $O(N^3)$ operations, so the form (38) is generally very fast.

Matrix-vector products involving third-order tensors of the form $A = A_z \otimes A_y \otimes A_x$ can be evaluated with similar efficiencies. In partcular, $\underline{z} = A\underline{u}$ would be evaluated as

$$v_{ijk} \;=\; \sum_{p=1}^{n_x} (A_x)_{ip} u_{pjk} \qquad \underline{v} \;=\; (I_z \otimes I_y \otimes A_x)\underline{u} \tag{39}$$

$$w_{ijk} \;=\; \sum_{q=1}^{n_y} (A_y)_{jq} v_{iqk} \qquad \underline{w} \;=\; (I_z \otimes A_y \otimes I_x)\underline{v} \tag{40}$$

$$z_{ijk} \;=\; \sum_{r=1}^{n_z} (A_z)_{kr} w_{ijr} \qquad \underline{z} \;=\; (A_z \otimes I_y \otimes I_x)\underline{w}, \tag{41}$$

which has a total operation count of $O(N^4)$ and storage count of $O(N^3)$ for the input and output data. It is also possible, with minor effort, to recast (39)–(41) in terms of fast matrix-matrix products. (Recall, in 3D, $N^3 \approx n$.)

3.1 Fast Diagonalization Method (FDM)

We will use the tools of the preceding section to develop fast solvers for the systems in (22) and (26). This idea originated in a 1964 paper by Birkhoff, Lynch, and Rice that predates the FFT by one year. In the present exposition we assume that A_* is the tridiagonal 1D Poisson operator of the preceding section, but the method carries through with the same complexity even if A_* is full (e.g., as might be the case for a high-order approximation to the Poisson operator).

To start, we assume that the 1D matrices A_* have the similarity transform $A_* = S_* \Lambda_* S_*^{-1}$, for $*$=x, y, and z, where Λ_* is the diagonal matrix of eigenvalues and S_* the corresponding matrix of eigenvectors. For the 2D case, we have

$$\begin{aligned} A_{2D} &= (I_y \otimes A_x) \; + \; (A_y \otimes I_x) && (42)\\ &= (S_y S_y^{-1} \otimes S_x \Lambda_x S_x^{-1}) \; + \; (S_y \Lambda_y S_y^{-1} \otimes S_x S_x^{-1}) && (43)\\ &= (S_y \otimes S_x)\,(I_y \otimes \Lambda_x \; + \; \Lambda_y \otimes I_x)\,(S_y^{-1} \otimes S_x^{-1}), && (44) \end{aligned}$$

whose inverse is given by

$$\begin{aligned} A_{2D}^{-1} &= (S_y \otimes S_x)\, D^{-1}\, (S_y^{-1} \otimes S_x^{-1}), && (45)\\ & && (46) \end{aligned}$$

with the trivially inverted diagonal matrix

$$D \;:=\; (I_y \otimes \Lambda_x \; + \; \Lambda_y \otimes I_x). \qquad (47)$$

Aside from the preprocessing costs to find $[S_*, \Lambda_*]$, the total work is $\approx 8N^3$, assuming $N_x = N_y = N$.

Let's look more closely at the steps of the FDM.

Starting with known grid data, $\underline{f} = [f_{11}\ f21\ \cdots\ f_{n_x n_y}]$, we compute the solution to $A\underline{u} = \underline{f}$ as

$$\underline{u} \;=\; \underbrace{(S_y \otimes S_x)}_{S}\ \underbrace{D^{-1}}_{\Lambda^{-1}}\ \underbrace{(S_y^{-1} \otimes S_x^{-1})}_{S^{-1}}\ \underline{f}, \tag{48}$$

where we have emphasized that $S := (S_y \otimes S_x)$ is the matrix of eigenvectors of $A := A_{2D}$ and Λ is the corresponding matrix of eigenvalues.
Let's break this operation down further:

$$\begin{array}{lrcll} \textit{i)} & \hat{\underline{f}} & = & (S_y^{-1} \otimes S_x^{-1})\,\underline{f} & \text{Fourier transform of data} \\ \textit{iii)} & \hat{\underline{u}} & = & D^{-1}\hat{\underline{f}} & \text{divide by wavenumbers squared} \\ \textit{iii)} & \underline{u} & = & (S_y \otimes S_x)\,\hat{\underline{u}} & \text{combine eigenvectors to construct solution.} \end{array} \tag{49}$$

The "Fourier" transform requires two tensor contractions of the form (38), which are implemented as

$$\hat{F} \;=\; S_x^{-1}\, F\, S_y^{-T}, \tag{50}$$

requiring $\sim 2(2N^3)$ floating point operations. Division by the eigenvalues requires $\sim N^2 \ll N^3$ operations. Finally, the inverse transform is expressed as a second pair of tensor contractions,

$$U \;=\; S_x\, \hat{U}\, S_y, \tag{51}$$

such that the total complexity is $\sim 8N^3$. Here, we are neglecting the cost to find the eigenpairs as these are known in closed-form. For more general cases, we could call an eigenvalue solver to find (S_x, Λ_x) and (S_y, Λ_y) as part of a one-time setup.

It is important to note that, for the uniformly-spaced grid case, the $O(N^3)$ complexity can be reduced to $O(N^2 \log N)$ by use of the FFT. This approach is used by `fishpack`, which s about 10–20× faster than multigrid for 2D problems.

An interesting aside about the fast diagonalization method is that it allows us to easily solve systems of the form $f(A)\underline{u} = \underline{b}$. In 2D, the answer is

$$\underline{u} \;=\; (S_y \otimes S_x)\, F^{-1}\, (S_y^{-1} \otimes S_x^{-1})\underline{u}, \tag{52}$$

where F is the diagonal matrix with entries $F_{ss} = f(\lambda_{x,k} + \lambda_{y,l})$, with $s = k + n_x(l-1)$ for $k = 1, \ldots, n_x$ and $l = 1, \ldots, n_y$.

The 3D complexity is similar, with

$$A_{3D}^{-1} = (S_z \otimes S_y \otimes S_x)\, D^{-1}\, (S_z^{-1} \otimes S_y^{-1} \otimes S_x^{-1}), \tag{53}$$

$$\tag{54}$$

with

$$D := (I_z \otimes I_y \otimes \Lambda_x + I_z \otimes \Lambda_y \otimes I_x + \Lambda_z \otimes I_y \otimes I_x). \tag{55}$$

The work is $\approx 12N^4$.

Fast diagonalization is readily extended to the more general case that arises in finite element methods,

$$A_{3D} = (B_z \otimes B_y \otimes A_x) + (B_z \otimes A_y \otimes B_x) + (A_z \otimes B_y \otimes B_x), \tag{56}$$

where B_* is the mass matrix in the "*" direction. The corresponding one-dimensional eigenpairs are solutions to the *generalized eigenproblem*, $A_* S_* = B_* S_* \Lambda_*$, with eigenvectors normalized to satisfy $S_*^T B_* S_* = I_*$.

4 Green's Functions and A^{-1}

Note that for any system $A\underline{u} = \underline{f}$ we can explicitly identify the columns of A^{-1} by successively solving $A\underline{u} = \underline{f}$ with $\underline{f} = \underline{\hat{e}}_j$, $j = 1, \ldots, n$, where $\underline{\hat{e}}_j$ is the jth column of the $n \times n$ identity matrix. For the 1D Poisson problem this concept is particularly illuminating because we know that the (physical) solution to the problem $A\underline{u} = \underline{\hat{e}}_j = [0 \;\; 0 \ldots 1 \ldots 0]^T$ is a "tent" function that peaks at $\underline{x}_j$ and decays linearly to zero at x_0 and x_N. The linear decay represents a function whose second derivative is zero. This tent solution corresponds to the Green's function. If we denote by $\underline{\hat{u}}_j$ the solution to $A\underline{\hat{u}}_j = \underline{\hat{e}}_j$, then the solution to $A\underline{u} = \underline{f}$ for a general (discrete) function $[f_j]$ will be

$$\underline{u} \;\; = \;\; \sum_{j=1}^{n} \underline{\hat{u}}_j f_j \;\; = \;\; A^{-1}\underline{f}. \tag{57}$$

The first bit of insight gained from this perspective is that A^{-1} *is completely full.* Any point source $\underline{f} = \underline{\hat{e}}_j$ will generate a non-trivial solution $\underline{\hat{u}}_j$ throughout the domain. Also, A^{-1} is a positive matrix, $A^{-1} > O$.

Continuing with this thermal analogy, we can further assert that A^{-1} is full in higher space dimensions as well. From a parallel computing perspective, with $\underline{u}$ and $\underline{f}$ distributed vectors, we recognize that solving $A\underline{u} = \underline{f}$ (inescapably) involves all-to-all communication: each nontrivial right-hand-side value f_i has a nontrivial impact on each solution point u_j. In Fig. 11 we show Green's function examples for 1D and 2D.

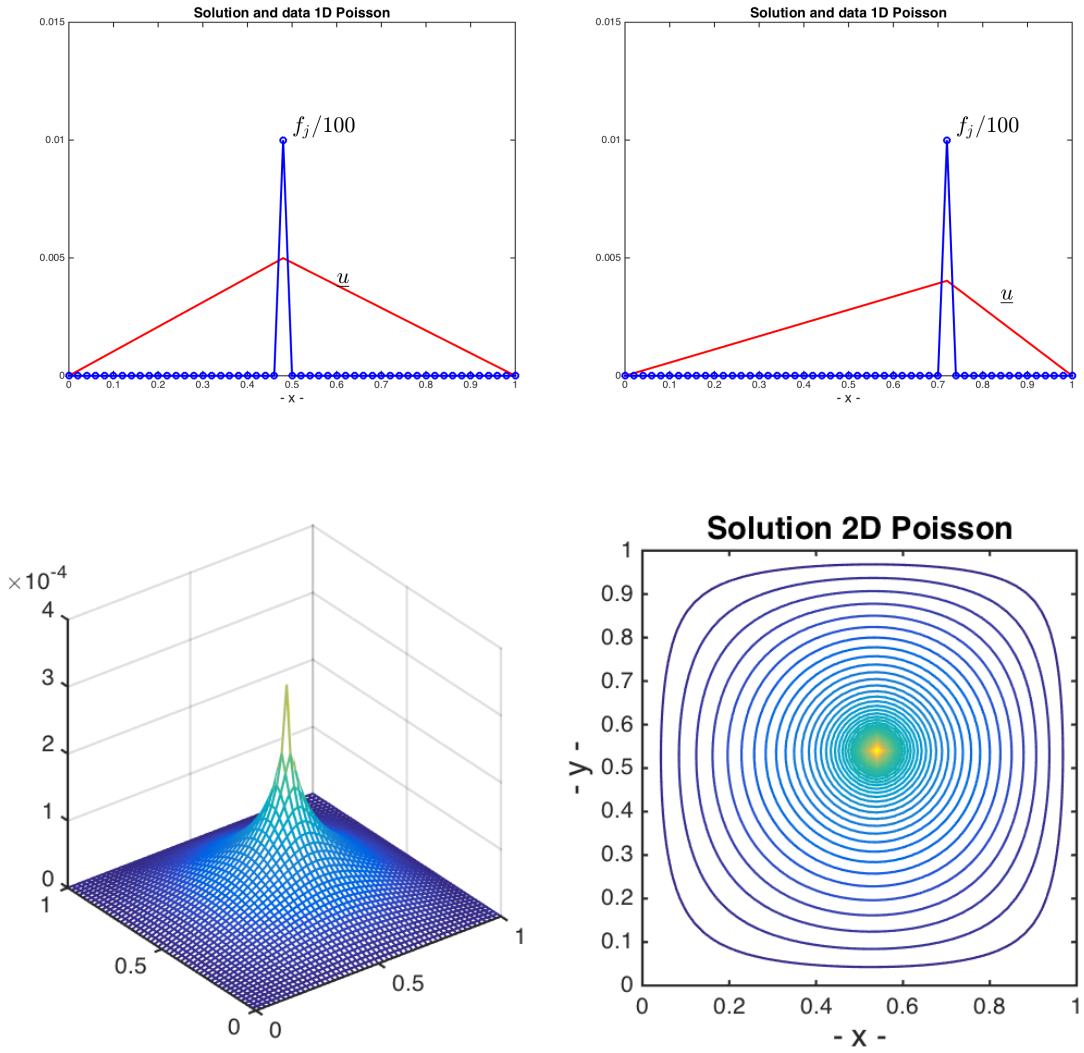


Figure 11: Green's function examples for (top) 1D and (bottom) 2D Poisson problems using finite differences with N=50.

5 Convergence Analysis

We saw in (4) that the stiffness matrix A applied to the vector $\underline{\tilde{u}} := [\tilde{u}_i]$ produces $O(\Delta x^2)$ approximations to $-\tilde{u}_i^{''}$ and, from (16), that the eigenvalues of A are $O(\Delta x^2)$ approximations to their continuous counterparts, $\tilde{\lambda}_k$, for sufficiently small k/N. To establish that $\underline{u} \approx \underline{\tilde{u}}$ we need one more piece of information, namely, that the discrete system is stable. Specifically, we will establish that $A\underline{u} = \underline{f}$ implies that the solution is bounded by the data,

$$||\underline{u}||_\infty \quad \leq \quad \frac{L^2}{8}||\underline{f}||_\infty. \tag{58}$$

Crucially, (58) holds *for all* n.

5.1 Application of Stability

Before deriving the stability result (58) we will show why it is useful. Let the pointwise error $e_j := \tilde{u}_j - u_j$. Subtracting (7) from (5), leads to

$$-\frac{\delta^2 e_j}{\delta x^2} \quad = \quad = \quad -\frac{\Delta x^2}{12}\frac{d^4\tilde{u}}{dx^4}\bigg|_{x_j} + O(h^4) \tag{59}$$

$$= \quad -\frac{\Delta x^2}{12}\frac{d^2 f}{dx^2}\bigg|_{x_j} + O(h^4). \tag{60}$$

Under the assumption that f is sufficiently smooth (i.e., $f^{''} < \infty$), we can apply the stability result (58) to (60),

$$||\underline{e}||_\infty \quad \leq \quad \frac{L^2}{8}\frac{\Delta x^2}{12}||\underline{f}^{''}||_\infty, \tag{61}$$

which implies that the *maximum pointwise error*, $||\underline{\tilde{u}} - \underline{u}||_\infty = O(\Delta x^2)$.

5.2 Derivation of Stability

We begin by deriving the continuous analogy of (58).[1] Let

$$\frac{d\tilde{u}}{dx} \quad =: \quad w(x) \tag{62}$$

be the solution to (3) such that

$$\frac{dw}{dx} \quad = \quad -f(x). \tag{63}$$

Integrate (63) to find

$$w(x) \quad = \quad c_2 \; - \; \underbrace{\int_0^x f(s)\,ds}_{F(x)} \; = \; c_2 \; - \; F(x). \tag{64}$$

[1] See Chap. 12 of Quarteroni, Sacco, and Saleri, *Numerical Mathematics*, Springer, 2007.

Inserting (64) into (62) and integrating yields

$$\tilde{u}(x) \;=\; c_1 \;+\; \int_0^x w(s)\,ds \tag{65}$$

$$\;=\; c_1 \;+\; c_2 x \;-\; \int_0^x \underbrace{F(s)}_{\text{“}v\text{”}}\,\underbrace{1\cdot ds}_{\text{“}du\text{”}}, \qquad u(0)=0 \implies c_1 = 0 \tag{66}$$

$$\;=\; c_2 x \;+\; \int_0^x s\,F'(s)\,ds \;-\; sF(s)|_0^x\,, \tag{67}$$

$$\;=\; c_2 x \;+\; \int_0^x s\,f(s)\,ds \;-\; x\int_0^x f(s)\,ds, \tag{68}$$

where we have used integration by parts to eliminate $\int F(s)ds$. Combining the pair of integrals in (68), we have an equation for the solution $\tilde{u}$ at any point x,

$$\boxed{\tilde{u}(x) \;=\; c_2 x \;+\; \int_0^x (s-x)\,f(s)\,ds.} \tag{69}$$

We proceed by using the boundary condition at $x = L$ to eliminate the integration constant c_2.

$$\tilde{u}(L) \;=\; c_2 L \;+\; \int_0^L (s-L)\,f(s)\,ds \;=\; 0, \tag{70}$$

from which

$$\tilde{u}(x) \;=\; \int_0^x (s-x)\,f(s)\,ds \;-\; \frac{x}{L}\int_0^L (s-L)\,f(s)\,ds \tag{71}$$

$$\;=\; \int_0^x (s-x)\,f(s)\,ds \;-\; \frac{x}{L}\int_0^x (s-L)\,f(s)\,ds \;-\; \frac{x}{L}\int_x^L (s-L)\,f(s)\,ds \tag{72}$$

$$\;=\; \int_0^x \left[(s-x) - \frac{x}{L}(s-L)\right]\,f(s)\,ds \;-\; \frac{x}{L}\int_x^L (s-L)\,f(s)\,ds \tag{73}$$

$$\;=\; \int_0^x s(1-x/L)\,f(s)\,ds \;+\; \int_x^L x(1-s/L)\,f(s)\,ds \tag{74}$$

$$\;=\; \int_0^L G(x,s)\,f(s)\,ds. \tag{75}$$

Here, the *Green's Function*, $G(x,s)$ is defined as

$$G(x,s) \;:=\; \begin{cases} s(1-x/L), & s \le x \\ x(1-s/L), & s \ge x \end{cases}. \tag{76}$$

The Green's function is plotted for several values of x in Fig. 2 (right). The maximum of G is $L/4$. Moreover,

$$\int_0^L G(x,s)\,ds \;=\; \frac{L^2}{2}\frac{x}{L}\left(1-\frac{x}{L}\right) \;\le\; \frac{L^2}{8}. \tag{77}$$

Thus, for any $x \in \Omega$, we have

$$|u(x)| = \left| \int_0^L G(x,s)\, f(s)\, ds \right| \tag{78}$$

$$\leq \int_0^L |G(x,s)|\, |f(s)|\, ds \tag{79}$$

$$\leq \frac{L^2}{8} \max_{s\in\Omega} |f(s)| = \frac{L^2}{8} ||f||_\infty, \tag{80}$$

which establishes the desired result.

For the discrete case, our solution is

$$\underline{u} = A^{-1}\underline{f} = G\underline{f} = \sum_{j=1}^{n} \underline{g}_j f_j, \tag{81}$$

where $G := A^{-1}$. Clearly, $\underline{u}$ is a linear combination of the columns of G, which constitute the discrete Green's functions. For the 1D case, each column j has entries[2]

$$g_{ij} = \begin{cases} \frac{L^2}{N}\frac{i}{N}\left(1-\frac{j}{N}\right), & i \leq j \\ \frac{L^2}{N}\frac{j}{N}\left(1-\frac{i}{N}\right), & i \geq j \end{cases}. \tag{82}$$

To see this, consider the product $W = AG$. Apply row i of A to column j of G, with $i < j$.

$$w_{ij} = -\frac{L^2}{N^2\Delta x^2}\left(1-\frac{j}{N}\right)[(i-1) - 2i + (i+1)] = 0, \tag{83}$$

with a similar result for $i > j$. For the case $i = j$, we have

$$w_{ii} = -\frac{L^2}{N^2\Delta x^2}\left[(i-1)\left(1-\frac{i}{N}\right) - 2i\left(1-\frac{i}{N}\right) + i\left(1-\frac{i+1}{N}\right)\right] \tag{84}$$

$$= 1, \tag{85}$$

which follows because $\Delta x = L/N$.

From (81), we have

$$||\underline{u}|| = ||G\underline{f}|| \leq ||G|| \cdot ||\underline{f}||, \tag{86}$$

which holds for any vector norm. The inequality on the right is nothing other than the definition of the matrix norm, $||G||$, induced by any chosen vector norm, $||\underline{f}||$. That is, for any vector norm $||\cdot||$ and G,

$$||G|| := \max_{\underline{u}\neq\underline{0}} \frac{||G\underline{u}||}{||\underline{u}||} = \max_{||\underline{u}||=1} ||G\underline{u}||. \tag{87}$$

In particular, with

$$||\underline{u}||_\infty := \max_i |u_i|, \tag{88}$$

[2]In 1D, the shape of the *discrete* Green's functions are identical to their continuous counterparts because the solution is *exact* on either side of x_j where $f_{i\neq j} = 0$. In these regions, the solution to $-\tilde{u}'' = 0$ is a polynomial of degree 1 and the truncation error is zero.

the corresponding infinity norm for the matrix G is readily shown to be,

$$||G||_\infty \;=\; \max_i \sum_{j=1}^{n} |g_{ij}|. \tag{89}$$

For (82), the maximum is realized for $i = N/2$ (for N even). Exploiting the symmetry of G, we have

$$||G||_\infty \;=\; \frac{L^2}{N^2}\left(1-\frac{1}{2}\right)\left[2\sum_{i=1}^{N/2-1} i \;+\; \frac{N}{2}\right] \tag{90}$$

$$\;=\; \frac{L^2}{2N^2}\left[\frac{N}{2}\left(\frac{N}{2}-1\right) \;+\; \frac{N}{2}\right] \;=\; \frac{L^2}{8}. \tag{91}$$

We reiterate that the crucial result is that $||G||_\infty$ is a constant that does not depend on the grid spacing (i.e., on N). Because of (59), this result is sufficent to bound the solution error, $||\underline{e}||_\infty \equiv ||\underline{\tilde{u}} - \underline{u}||_\infty \leq C\Delta x^2$, with a constant that is independent of N.

Remarks. We make a few closing remarks concerning the 1D analysis.

- An intuitive derivation of G follows from the fact that in intervals where $f(x) \equiv 0$, $\tilde{u}(x)$ will be a straight line as will $u(x_j) := u_j$. So, if f_j is the jth column of the identity matrix, u_i, $i < j$ will ascend linearly from $u_0 = 0$ to u_j and will then descend linearly from u_j to $u_N = 0$ for $i > j$. The value of u_j is set by the need for the jth column of G to satisfy the difference equation at row i. In any case, each entry of G is positive.

- In this example, in addition to being symmetric positive definite (SPD), A is what is known as an M-matrix. It has positive entries on the diagonal, all off-diagonal entries are ≤ 0, and the row-sum is ≥ 0 for each row. One property of M matrices is that their inverses ($G = A^{-1}$) are positive: $g_{ij} > 0$.

- If G is positive, then $||G||_\infty = ||G\underline{f}||_\infty$, when $\underline{f} = \underline{1}$ is the vector of all ones. That is, if $\underline{f} = \underline{1}$, and $\underline{u}$ is the solution to $A\underline{u} = \underline{f}$, then $||G||_\infty = \max_i u_i$. This result extends to multiple space dimensions whenever G is positive or A is an M-matrix.