Thursday, 9/26/13 Math 471, Fall 2013, Section 001 Lecture 4

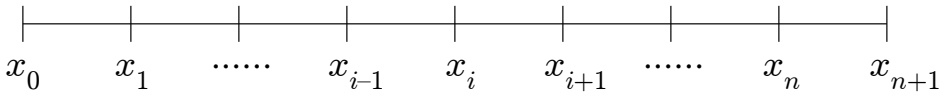
1 2-point boundary value problems

Text: sections 8.1 & 8.2

Find $y(x)$ on $0 \le x \le 1$ such that y satisfies the ODE $-y'' + q(x)y = r(x)$, subject to boundary conditions $y(0) = \alpha$, $y(1) = \beta$, where $q(x)$ and $r(x)$ are given. This equation represents a steady-state convection-reaction-diffusion system, and $y(x)$ may represent temperature or velocity, for example.

Finite difference schemes

Choose an integer n and set $h = \frac{1}{n+1}$. Now, instead of a step size, h is called the mesh size. Set $x_i = ih$ for $i = 0, 1, 2, \ldots, n, n+1$. The collection $\{x_i\}_{i=0}^{n+1}$ are the mesh points. Note: $x_0 = 0$ and $x_{n+1} = 1$.



Let $y(x_i) = y_i$ denote the exact solution at grid point i, $q_i = q(x_i)$ and $r_i = r(x_i)$.

Recall HW 2: $D_+ y_i = \dfrac{y_{i+1} - y_i}{h}$, $\quad D_- y_i = \dfrac{y_i - y_{i-1}}{h}$

$$D_+ D_- y_i = D_+ (D_- y_i) = D_+ \left(\frac{y_i - y_{i-1}}{h} \right) = \frac{1}{h} (D_+ y_i - D_+ y_{i-1})$$

$$= \frac{1}{h} \left(\frac{y_{i+1} - y_i}{h} - \left(\frac{y_i - y_{i-1}}{h} \right) \right) = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} \approx y''(x_i) [0.5\text{em}]$$

Question 1: How accurate is this approximation?

$y_{i+1} = y(x_{i+1}) = y(x_i + h)$: expand in a Taylor series about $x = x_i$

$$y_{i+1} = y_i + hy_i' + \tfrac{h^2}{2} y_i'' + \tfrac{h^3}{3!} y_i''' + \tfrac{h^4}{4!} y_i^{(4)} + \tfrac{h^5}{5!} y_i^{(5)} + O(h^6)$$

$$y_{i-1} = y_i - hy_i' + \tfrac{h^2}{2} y_i'' - \tfrac{h^3}{3!} y_i''' + \tfrac{h^4}{4!} y_i^{(4)} - \tfrac{h^5}{5!} y_i^{(5)} + O(h^6)$$

$$D_+ D_- y_i = \underbrace{\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}}_{\text{discrete approximation}} = \underbrace{y_i''}_{\substack{\uparrow \\ \text{exact value}}} + \underbrace{\frac{h^2}{12} y_i^{(4)} + O(h^4)}_{\text{discretization error}} : \text{2nd order accurate}$$

Tuesday, 10/1/13

Let w_i denote the numerical (approximate) solution at grid point i, so that $w_i \approx y_i$. We can set the boundary values exactly: $w_0 = \alpha$, $w_{n+1} = \beta$.

The equation for the ith grid point, $i = 2, \ldots, n-1$ is

$$-\left(\frac{w_{i+1} - 2w_i + w_{i-1}}{h^2} \right) + q_i w_i = r_i$$: rewrite so that the unknown w_i are on the left hand side

$$\frac{1}{h^2}\left(-w_{i+1} + (2+q_i h^2)w_i - w_{i-1}\right) = r_i$$

Special case 1: left boundary $i = 1 \Rightarrow w_{i-1} = w_0 = \alpha$.

$$\frac{1}{h^2}\left(-w_2 + (2+q_1 h^2)w_1 - \alpha\right) = r_1 \Rightarrow \frac{1}{h^2}\left(-w_2 + (2+q_1 h^2)w_1\right) = r_1 + \frac{\alpha}{h^2}$$

Special case 2: right boundary $i = n \Rightarrow w_{i+1} = w_{n+1} = \beta$.

$$\frac{1}{h^2}\left(-\beta + (2+q_n h^2)w_n - w_{n-1}\right) = r_n \Rightarrow \frac{1}{h^2}\left((2+q_n h^2)w_n - w_{n-1}\right) = r_n + \frac{\beta}{h^2}$$

We can rewrite the above equations for $i = 1, \ldots, n$ in matrix form.

$$\frac{1}{h^2}\begin{pmatrix} 2+q_1h^2 & -1 & & & & & \\ -1 & 2+q_2h^2 & -1 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & -1 & 2+q_{n-1}h^2 & -1 \\ & & & & & -1 & 2+q_nh^2 \end{pmatrix}\begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ \vdots \\ \vdots \\ w_{n-1} \\ w_n \end{pmatrix} = \begin{pmatrix} r_1 + \alpha/h^2 \\ r_2 \\ \vdots \\ \vdots \\ \vdots \\ r_{n-1} \\ r_n + \beta/h^2 \end{pmatrix}$$

Thus, the finite difference scheme we have derived approximates the BVP with a system of linear equations that depends on the mesh size h.

$$A_h w_h = r_h$$

The matrix A_h has structure: it is both tridiagonal and symmetric.

Question 2: Is A_h invertible for all h, $q(x)$, and $r(x)$?

Question 3: Can w_h be computed efficiently (faster than $O(n^3)$)?

Question 4: Does $w_h \to y$ as $h \to 0$, i.e., does the numerical solution converge to the exact solution as the mesh is refined?

1.1 LU factorization of a tridiagonal system

Text: appendix B

$$\begin{pmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & c_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & & a_n & b_n \end{pmatrix} = \begin{pmatrix} 1 & & & & & \\ l_2 & 1 & & & & \\ & \ddots & \ddots & & & \\ & & \ddots & \ddots & & \\ & & & \ddots & \ddots & \\ & & & & l_n & 1 \end{pmatrix}\begin{pmatrix} u_1 & c_1 & & & & \\ & u_2 & c_2 & & & \\ & & \ddots & \ddots & & \\ & & & \ddots & \ddots & \\ & & & & \ddots & c_{n-1} \\ & & & & & u_n \end{pmatrix}$$

At row i of tridiagonal matrix A, the entries a have indices $i, i-1$. All b entries are on the diagonal. All c entries have indices $i, i+1$.

Reading directly from the above system, we see that $b_1 = u_1 \Rightarrow u_1 = b_1$.

$$\left.\begin{array}{lll} a_k = l_k u_{k-1} & \Rightarrow & l_k = a_k/u_{k-1} \\ b_k = l_k c_{k-1} + u_k & \Rightarrow & u_k = b_k - l_k c_{k-1} \end{array}\right\} \text{ for } \quad k = 2:n$$

Algorithm 1: Tridiagonal Solver (Thomas Algorithm)

Input : Tridiagonal square matrix A, matrix size n, right-hand side vector r
Output: Vector w such that $Aw = r$

`% Part 1: Find L, U`

$u_1 = b_1$
for $k = 2 : n$ **do**
  $l_k = a_k / u_{k-1}$
  $u_k = b_k - l_k c_{k-1}$
end

`% Part 2: Solve L z = r (forward substitution)`

$z_1 = r_1$
for $k = 2 : n$ **do**
  $z_k = r_k - l_k z_{k-1}$
end

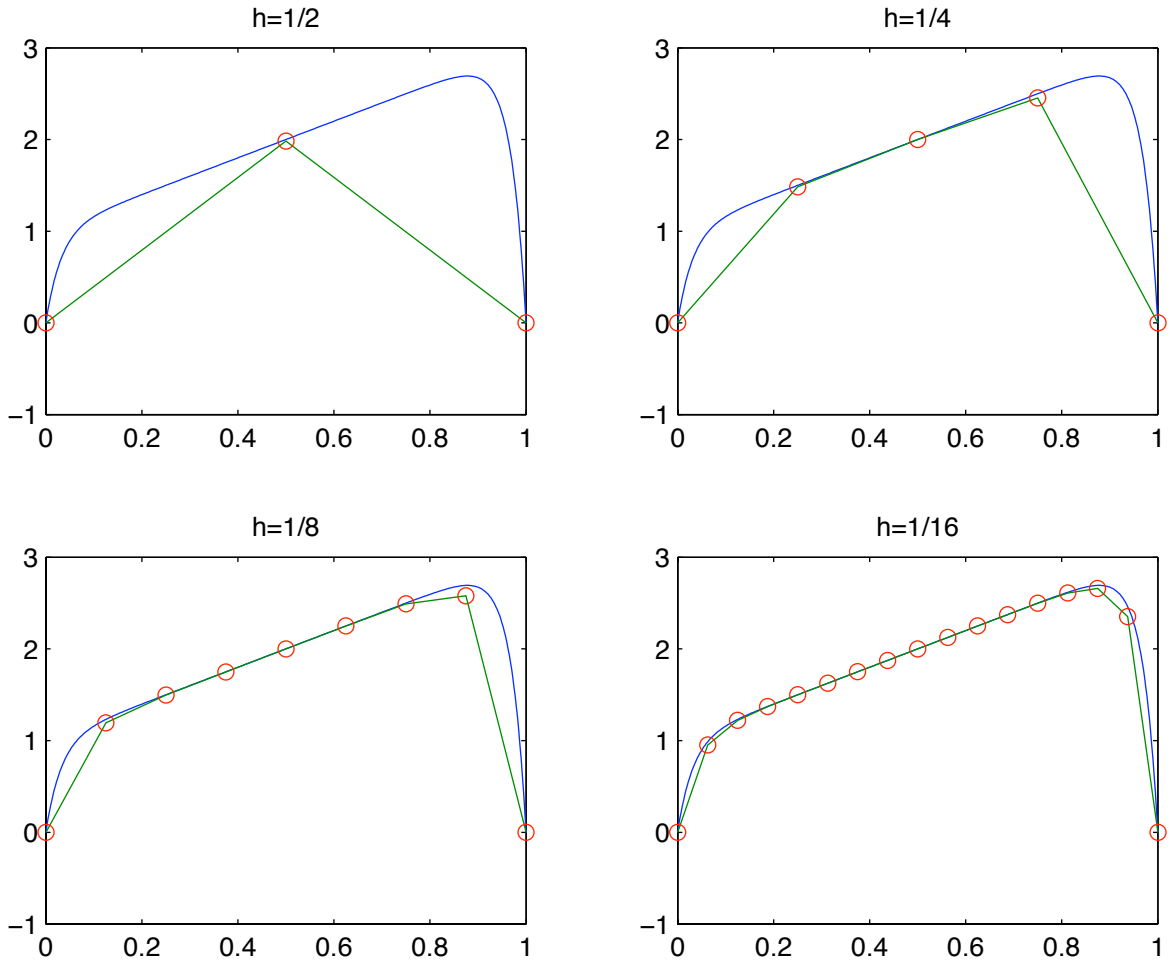
`% Part 3: Solve U w = z (backward substitution)`

$w_n = z_n / u_n$
for $k = n - 1 : -1 : 1$ **do**
  $w_k = (z_k - c_k w_{k+1}) / u_k$
end
return w

Notes :

- operation count $= O(n)$
- memory required $= O(n)$ if vectors are used instead of full matrices

Example 1: $-\epsilon y'' + y = 2x + 1$, $0 \le x \le 1$, $y(0) = 0$, $y(1) = 0$, $\epsilon = 10^{-3}$ solution : $y(x) = 2x + 1 - \left(\sinh \frac{1-x}{\sqrt{\epsilon}} + 3 \sinh \frac{x}{\sqrt{\epsilon}}\right) \Big/ \sinh \frac{1}{\sqrt{\epsilon}}$, check : computer project 1B



h	$\|y_h - w_h\|_\infty$	$\frac{\|y_h - w_h\|_\infty}{h}$	$\frac{\|y_h - w_h\|_\infty}{h^2}$	$\frac{\|y_h - w_h\|_\infty}{h^3}$
0.50000000	0.015872	0.031744	0.0634890	0.126979
0.25000000	0.045420	0.181682	0.7267300	2.906920
0.12500000	0.113164	0.905315	7.2425200	57.94016
0.06250000	0.107705	1.723287	27.572593	441.1615
0.03125000	0.041333	1.322659	42.325086	1354.402
0.01562500	0.010982	0.702852	44.982586	2878.885
0.00781250	0.002790	0.357233	45.725862	5852.910
0.00390625	0.000701	0.179364	45.917351	11754.84

Notes:

- For $\frac{1}{8} \leq h \leq \frac{1}{2}$, the error increases as h decreases. This is due to the presence of boundary layers (look closely at the plots).
- For $h \leq \frac{1}{32}$, if h decreases by a factor of 2, then the error decreases by approximately $\frac{1}{4}$.
- We see that $\|y_h - w_h\|_\infty = O(h^2)$ (column 4), so the method is second order accurate.

△

2 Iterative methods

Text : section 3.8

Gaussian elimination is an example of a direct method for solving $Ax = b$, in the sense that the exact solution is obtained after a finite number of steps. In practice, the $O(n^3)$ operation count is a serious obstacle (and storage can be an issue too). Now we consider a class of methods called iterative methods which generate sequences of approximate solutions x_k such that $\lim_{k\to\infty} x_k = x$. As we shall see, iterative methods have some advantages over direct methods.
Recall: We have already seen iterative methods for nonlinear scalar equations (rootfinding).

A direct method solves the problem $Ax - b = 0$.
We now rewrite the problem into an equivalent linear system $x = Bx + c$, for some new matrix B and vector c. Then $x_{k+1} = Bx_k + c$ is a fixed-point iteration. The matrix B is known as an iteration matrix.

2.1 Jacobi method

Matrix splitting: $A = L + D + U$: Note: these matrices are **not** the same as LU factorization.
$D = \text{diag}(a_{11}, a_{22}, \ldots, a_{nn})$, assume $a_{ii} \neq 0$ for $i = 1 : n$.

$$L = \begin{pmatrix} 0 & & & & \\ a_{21} & 0 & & & \\ \vdots & \ddots & \ddots & & \\ \vdots & & \ddots & \ddots & \\ a_{n1} & \ldots & \ldots & a_{n,n-1} & 0 \end{pmatrix}, U = \begin{pmatrix} 0 & a_{12} & \ldots & \ldots & a_{1n} \\ & 0 & \ddots & & \vdots \\ & & \ddots & \ddots & \vdots \\ & & & \ddots & a_{n-1,n} \\ & & & & 0 \end{pmatrix}$$

$$\begin{aligned} Ax = b &\Leftrightarrow (L + D + U)x = b \\ &\Leftrightarrow Dx = -(L + U)x + b \\ &\Leftrightarrow x = -D^{-1}(L + U)x + D^{-1}b \quad \Rightarrow B_J = -D^{-1}(L + U), \quad c = D^{-1}b \end{aligned}$$

$$Dx^{(k+1)} = -(L + U)x^{(k)} + b$$

Jacobi method in component form ($n = 3$)

$$L = \begin{pmatrix} 0 & 0 & 0 \\ a_{21} & 0 & 0 \\ a_{31} & a_{32} & 0 \end{pmatrix}, \quad D = \begin{pmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{pmatrix}, \quad U = \begin{pmatrix} 0 & a_{12} & a_{13} \\ 0 & 0 & a_{23} \\ 0 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \quad &\Rightarrow a_{11}x_1^{(k+1)} = b_1 - \left(a_{12}x_2^{(k)} + a_{13}x_3^{(k)}\right) \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \quad &\Rightarrow a_{22}x_2^{(k+1)} = b_2 - \left(a_{21}x_1^{(k)} + a_{23}x_3^{(k)}\right) \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \quad &\Rightarrow a_{33}x_3^{(k+1)} = b_3 - \left(a_{31}x_1^{(k)} + a_{32}x_2^{(k)}\right) \end{aligned}$$

Example 2: $A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$, $b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. The exact solution is $x_1 = x_2 = 1$. Let $x_1^{(0)} = x_2^{(0)} = 0$ be the initial guess.

$$2x_1^{(1)} = 1 - ((-1) \cdot 0) \Rightarrow x_1^{(1)} = \frac{1}{2}$$
$$2x_2^{(1)} = 1 - ((-1) \cdot 0) \Rightarrow x_2^{(1)} = \frac{1}{2}$$

k	$x_1^{(k)}$	$x_2^{(k)}$	$\left\|x - x^{(k)}\right\|_\infty$
0	0	0	1
1	1/2	1/2	1/2
2	3/4	3/4	1/4
3	7/8	7/8	1/8

Hence the numerical solution converges to the exact solution as $k \to \infty$.

Theorem 1. *Consider a fixed-point iteration of the form $x^{(k+1)} = Bx^{(k)} + c$. Then*

1. $e^{(k+1)} = Be^{(k)}$
2. *If $\|B\| < 1$ in any matrix norm then $\lim_{k\to\infty} x^{(k)} = x$ for any initial guess $x^{(0)}$.*

Proof. 1. $e^{(k+1)} = x - x^{(k+1)} = (Bx + c) - (Bx^{(k)} + c) = B(x - x^{(k)}) = Be^{(k)}$

2. $e^{(k)} = Be^{(k-1)} = B^2e^{(k-2)} = \cdots = B^ke^{(0)} \Rightarrow \left\|e^{(k)}\right\| = \left\|B^ke^{(0)}\right\| \le \left\|B^k\right\| \cdot \left\|e^{(0)}\right\|$

   $\left\|B^2\right\| = \|B \cdot B\| \le \|B\| \cdot \|B\| = \|B\|^2 \Rightarrow \left\|B^k\right\| \le \|B\|^k$

   $\Rightarrow \left\|e^{(k)}\right\| \le \|B\|^k \cdot \left\|e^{(0)}\right\| \to 0$ as $k \to \infty$

□

Example 3:

$$A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \Rightarrow B_J = -D^{-1}(L + U) = -\begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}\begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{pmatrix}$$
$$\Rightarrow \|B_J\|_\infty = \frac{1}{2}$$

1. Since $\|B_J\|_\infty < 1$ the theorem implies that Jacobi's method converges.
2. In each step the norm of the error decreases by $\frac{1}{2}$.

△

2.2 Gauss-Seidel method

Thursday, 10/3/13

Gauss-Seidel is another matrix splitting method, which we can write in terms of a fixed point method $x = Bx + c$.

$A = L + D + U$, as before.

$Ax = b \Leftrightarrow (L + D + U)x = b$

$\Leftrightarrow (L + D)x + Ux = b$

$\Leftrightarrow (L + D)x = -Ux + b$

$\Leftrightarrow x = -(L + D)^{-1}Ux + (L + D)^{-1}b, \quad \Rightarrow B_{GS} = -(L + D)^{-1}U, \quad c = (L + D)^{-1}b$

$(L + D)x^{(k+1)} = -Ux^{(k)} + b$: lower triangular system, solve by forward substitution

Gauus-Seidel method in component form ($n = 3$)

$$L + D = \begin{pmatrix} a_{11} & 0 & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{pmatrix}, \quad U = \begin{pmatrix} 0 & a_{12} & a_{13} \\ 0 & 0 & a_{23} \\ 0 & 0 & 0 \end{pmatrix}$$

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 &\Rightarrow a_{11}x_1^{(k+1)} = b_1 - (a_{12}x_2^{(k)} + a_{13}x_3^{(k)}) \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 &\Rightarrow a_{21}x_1^{(k+1)} + a_{22}x_2^{(k+1)} = b_2 - a_{23}x_3^{(k)} \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 &\Rightarrow a_{31}x_1^{(k+1)} + a_{32}x_2^{(k+1)} + a_{33}x_3^{(k+1)} = b_3 \end{aligned}$$

$$\begin{aligned} a_{11}x_1^{(k+1)} &= b_1 - \left(a_{12}x_2^{(k)} + a_{13}x_3^{(k)}\right) \\ a_{22}x_2^{(k+1)} &= b_2 - \left(a_{21}x_1^{(k+1)} + a_{23}x_3^{(k)}\right) \\ a_{33}x_3^{(k+1)} &= b_3 - \left(a_{31}x_1^{(k+1)} + a_{32}x_2^{(k+1)}\right) \end{aligned}$$

Hence $x_i^{(k+1)}$ is used as soon as it's computed, in contrast with Jacobi, which does not use the $k + 1$ components until the entirety of the kth iteration is complete.

Example 4: $A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

$$\begin{aligned} 2x_1 - x_2 = 1 &\Rightarrow 2x_1^{(k+1)} = 1 + x_2^{(k)} \\ -x_1 + 2x_2 = 1 &\Rightarrow 2x_2^{(k+1)} = 1 + x_1^{(k+1)} \end{aligned}$$

k	$x_1^{(k)}$	$x_2^{(k)}$	$\left\|e^{(k)}\right\|_\infty$
0	0	0	1
1	1/2	3/4	1/2
2	7/8	15/16	1/8
3	31/32	63/64	1/32

We see that Gauss-Seidel converges faster than Jacobi.

$$A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \Rightarrow B_{GS} = -(L + D)^{-1}U = -\frac{1}{4}\begin{pmatrix} 2 & 0 \\ 1 & 2 \end{pmatrix}\begin{pmatrix} 0 & -1 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1/2 \\ 0 & 1/4 \end{pmatrix}$$

$$\Rightarrow \|B_{BS}\|_\infty = \frac{1}{2}$$

1. Since $\|B_{GS}\|_\infty < 1$, the theorem implies that Gauss-Seidel converges.
2. In each step the norm of the error decreases by a factor of $\frac{1}{4}$.

Summary:

$A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$

$B_J = \begin{pmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{pmatrix} \Rightarrow \|B_J\|_\infty = \frac{1}{2}, \quad \left\|e^{(k+1)}\right\|_\infty = \frac{1}{2}\left\|e^{(k)}\right\|_\infty$

$B_{GS} = \begin{pmatrix} 0 & \frac{1}{2} \\ 0 & \frac{1}{4} \end{pmatrix} \Rightarrow \|B_{GS}\|_\infty = \frac{1}{2}, \quad \left\|e^{(k+1)}\right\|_\infty = \frac{1}{4}\left\|e^{(k)}\right\|_\infty$

We see that $\left\|e^{(k+1)}\right\|_\infty \leq \|B\|_\infty \cdot \left\|e^{(k)}\right\|_\infty$ in both cases, but this bound is not "sharp" (because the error in Gauss-Seidel decreases faster than this bound would suggest, $\frac{1}{4} < \|B_{GS}\|_\infty$). To explain the faster convergence of Gauss-Seidel, we need to consider the eigenvalues of the iteration matrix.

Definition 2. If $Ax = \lambda x$, where $x \neq 0$ is a vector (real or complex) and λ is a scalar (real or complex), then λ is an eigenvalue of A and x is a corresponding eigenvector.

Example 5: $A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$: permutation matrix

$A\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \Rightarrow \lambda = 1$ is an eigenvalue with eigenvector $x = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

$A\begin{pmatrix} 1 \\ -1 \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \end{pmatrix} \Rightarrow \lambda = -1$ is an eigenvalue with eigenvector $x = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$

△

Notes:

- $Ax = \lambda x, \quad x \neq 0 \quad \Leftrightarrow \quad (A - \lambda I)x = 0, \quad x \neq 0 \quad \Leftrightarrow \quad \det(A - \lambda I) = 0$
- The polynomial $f_A(\lambda) = \det(A - \lambda I)$ is called the characteristic polynomial of A
- Hence the roots of the characteristic polynomial are the eigenvalues of A.

Example 6: $A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$: permutation matrix

$f_A(\lambda) = \det(A - \lambda I) = \det\begin{pmatrix} -\lambda & 1 \\ 1 & -\lambda \end{pmatrix} = \lambda^2 - 1, \qquad f_A(\lambda) = 0 \Rightarrow \lambda = \pm 1$

Theorem 3. *If A is upper-triangular, then its eigenvalues are the diagonal elements.*

Proof. $A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ & \ddots & \vdots \\ 0 & & a_{nn} \end{pmatrix} \Rightarrow A - \lambda I = \begin{pmatrix} a_{11} - \lambda & \dots & a_{1n} \\ & \ddots & \vdots \\ 0 & & a_{nn} - \lambda \end{pmatrix}$

$f_A(\lambda) = \det(A - \lambda I) = (a_{11} - \lambda)(a_{22} - \lambda)\dots(a_{nn} - \lambda) = 0 \Rightarrow \lambda = a_{ii}$ for some i

□

Recall: $A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \quad \Rightarrow \quad B_{GS} = \begin{pmatrix} 0 & \frac{1}{2} \\ 0 & \frac{1}{4} \end{pmatrix}$

$\lambda_1 = 0$ is an eigenvalue with eigenvector $v_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, check ...

$\lambda_2 = \frac{1}{4}$ is an eigenvalue with eigenvector $v_2 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$, check ...

$e^{(0)} = x - x_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \end{pmatrix} = v_2 - v_1$

$e^{(1)} = Be^{(0)} = B(v_2 - v_1) = Bv_2 - Bv_1 = \lambda_2 v_2 - \lambda_1 v_1 = \lambda_2 v_2$

$e^{(2)} = Be^{(1)} = B(\lambda_2 v_2) = \lambda_2 B v_2 = \lambda_2 \cdot \lambda_2 v_2 = \lambda_2^2 v_2$

$\vdots$

$e^{(k)} = \lambda_2^k v_2 = \left(\frac{1}{4}\right)^k v_2 \Rightarrow \left\|e^{(k)}\right\|_\infty = \left(\frac{1}{4}\right)^k \|v_2\|_\infty$

This explains why $\left\|e^{(k+1)}\right\|_\infty = \frac{1}{4}\left\|e^{(k)}\right\|_\infty$ in this case, even though $\|B_{GS}\|_\infty = \frac{1}{2}$.

Question 5: What determines the rate of convergence of an iterative method?

Definition 4. The spectral radius of a matrix A is the eigenvalue of A with the largest magnitude,

$$\rho(A) = \max\{|\lambda| : \lambda \text{ is an eigenvalue of } A\}.$$

The spectral radius of an iteration matrix B determines the asymptotic rate of convergence of an iterative method.

Theorem 5. *Let $x = Bx + c$ define an iterative method. Then*

1. $\left\|e^{(k+1)}\right\|_\infty \le \|B\|_\infty \cdot \left\|e^{(k)}\right\|_\infty$ *for all $k \le 0$: (error bound)*
2. $\left\|e^{(k+1)}\right\|_\infty \sim \rho(B) \cdot \left\|e^{(k)}\right\|_\infty$ *as $k \to \infty$: (asymptotic relation)*

   $\Rightarrow \lim_{k\to\infty} \dfrac{\left\|e^{(k+1)}\right\|_\infty}{\left\|e^{(k)}\right\|_\infty} = \rho(B).$

Proof. 1. Recall: $e^{(k+1)} = Be^{(k)} \Rightarrow \left\|e^{(k+1)}\right\|_\infty = \left\|Be^{(k)}\right\|_\infty \le \|B\|_\infty \cdot \left\|e^{(k)}\right\|_\infty$.

2. Math 571 (but the idea is similar to the example above).

□

Example 7: $A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \quad \Rightarrow \quad B_J = \begin{pmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{pmatrix} \quad \Rightarrow \quad \rho(B_J) = \frac{1}{2}$

$B_{GS} = \begin{pmatrix} 0 & \frac{1}{2} \\ 0 & \frac{1}{4} \end{pmatrix} \quad \Rightarrow \quad \rho(B_{GS}) = \frac{1}{4}$

Question 6: Can we design faster methods?

Jacobi (1804-1851), Gauss (1777-1855), Seidel (1821-1896)

Richardson (1881 - 1953) : numerical weather forecasting (1922)

Idea: Stick with matrix splitting methods, find a way to decrease the spectral radius of B.

$Ax = b, \quad A = L + D + U$ leads to an equivalent system $x = Bx + c$.
Recall the Gauss-Seidel method:

$$(L+D)x^{(k+1)} = -Ux^{(k)} + b \Leftrightarrow Dx^{(k+1)} = Dx^{(k)} - \left(\underbrace{Lx^{(k+1)} + (D+U)x^{(k)} - b}_{\text{GS update}} \right)$$

Interpretation: We can consider the map $x^{(k+1)} = Bx^{(k)} + c$ as a system with an equilibrium point that satisfies $x = Bx + c$. We say that the system "relaxes" toward this equilibrium (provided $\|B\| < 1$) as the iterations progress. The process of relaxing can be thought of as applying a correction, or update, to the current approximation $x^{(k)}$ to get $x^{(k+1)}$. The Gauss-Seidel method is also called the method of successive relaxation.
Idea: Slow convergence could be caused by taking updates that are too conservative; perhaps we can improve convergence if we take larger updates.
Let ω be a free parameter and consider a modified iteration:

$$(L+D)x^{(k+1)} = -Ux^{(k)} + b \Leftrightarrow Dx^{(k+1)} = Dx^{(k)} - \omega \left(Lx^{(k+1)} + (D+U)x^{(k)} - b \right)$$

Note that if $\omega = 1$, this scheme is exactly the Gauss-Seidel method. If $\omega > 1$, we create a larger update, or larger relaxation, at step k and this method is called SOR, which stands for Successive OverRelaxation.

SOR method in component form ($n = 3$)

$$\begin{aligned}
a_{11}x_1^{(k+1)} &= a_{11}x_1^{(k)} - \omega \left(a_{11}x_1^{(k)} + a_{12}x_2^{(k)} + a_{13}x_3^{(k)} - b_1 \right) \\
a_{22}x_2^{(k+1)} &= a_{22}x_2^{(k)} - \omega \left(a_{21}x_1^{(k+1)} + a_{22}x_2^{(k)} + a_{23}x_3^{(k)} - b_2 \right) \\
a_{33}x_3^{(k+1)} &= a_{33}x_3^{(k)} - \omega \left(a_{31}x_1^{(k+1)} + a_{32}x_2^{(k+1)} + a_{33}x_3^{(k)} - b_3 \right)
\end{aligned}$$

Example 8:

$$2x_1 - x_2 = 1 \quad \Rightarrow 2x_1^{(k+1)} = 2x_1^{(k)} - \omega \left(2x_1^{(k)} - x_2^{(k)} - 1 \right)$$

$$-x_1 + 2x_2 = 1 \Rightarrow 2x_2^{(k+1)} = 2x_2^{(k)} - \omega \left(-x_1^{(k+1)} + 2x_2^{(k)} - 1 \right)$$

Matrix form:

$$(\omega L + D)x^{(k+1)} = \left((1-\omega)D - \omega U \right) x^{(k)} + \omega b$$

$$\Rightarrow B_\omega = (\omega L + D)^{-1} \left((1-\omega)D - \omega U \right)$$

Example 9: $A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

$$\begin{pmatrix} 2 & 0 \\ -\omega & 2 \end{pmatrix} \begin{pmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \end{pmatrix} = \begin{pmatrix} 2(1-\omega) & \omega \\ 0 & 2(1-\omega) \end{pmatrix} \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \end{pmatrix} + \omega \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$B_\omega = \begin{pmatrix} 2 & 0 \\ -\omega & 2 \end{pmatrix}^{-1} \begin{pmatrix} 2(1-\omega) & \omega \\ 0 & 2(1-\omega) \end{pmatrix} = \begin{pmatrix} 1-\omega & \frac{1}{2}\omega \\ \frac{1}{2}\omega(1-\omega) & \frac{1}{4}\omega^2 + 1 - \omega \end{pmatrix}$$

check : $\omega = 1 \Rightarrow B_\omega = \begin{pmatrix} 0 & \frac{1}{2} \\ 0 & \frac{1}{4} \end{pmatrix} = B_{GS}$

Question 7: Can we choose ω so that $\rho(B_\omega)$ is smaller?

Theorem 6 (Young, 1950)**.** *Choosing the relaxation parameter ω for SOR.*

1. *If $\rho(B_\omega) < 1$, then $0 < \omega < 2$.*
2. *Assume A is block tridiagonal, symmetric and positive definite (we'll define this term soon).*
   Then $\omega^* = \dfrac{2}{1 + \sqrt{1 - \rho(B_J)^2}}$ is the optimal SOR parameter in the sense that
   $\rho(B_{\omega^*}) = \min\limits_{0<\omega<2} \rho(B_\omega) = \omega^* - 1 < \rho(B_{GS}) < \rho(B_J) < 1.$

Proof. Math 571 or 572 (depends on professors).

Definition 7. A matrix A is positive definite if $x^T A x > 0$ for all $x \neq 0$ (section 3.7).

Example 10: $A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$ is positive definite.

Proof.

$$x^T A x = (x_1, x_2) \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = (x_1, x_2) \begin{pmatrix} 2x_1 - x_2 \\ -x_1 + 2x_2 \end{pmatrix}$$
$$= 2(x_1 + x_2)^2 - 2x_1 x_2 = x_1^2 + x_2^2 + (x_1 - x_2)^2 \geq 0$$

If $x \neq 0$, then either $x_1 \neq 0$ or $x_2 \neq 0$, but in any case $x^T A x > 0$.

△

Example 11: $A = \begin{pmatrix} 2 & 1 \\ -1 & 2 \end{pmatrix}$ is positive definite. (hw)

Example 12: $A = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$ is **not** positive definite.

Proof.

$$x^T A x = (x_1, x_2) \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = (x_1, x_2) \begin{pmatrix} x_1 + 2x_2 \\ 2x_1 + x_2 \end{pmatrix}$$
$$= x_1^2 + x_2^2 + 4x_1 x_2 : \text{indefinite}$$

For example : $x = [1, 0]^T \Rightarrow x^T A x = 1$, but $x = [1, -1]^T \Rightarrow x^T A x = -2$.

△

Tuesday, 10/8/13

Return to example: $A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \quad \Rightarrow \quad B_J = \begin{pmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{pmatrix} \quad \Rightarrow \quad \rho(B_J) = \frac{1}{2}$

$$\Rightarrow \omega^* = \frac{2}{1+\sqrt{1-(\frac{1}{2})^2}} = \frac{4}{2+\sqrt{3}} = 1.0718$$

k	$x_1^{(k)}$	$x_2^{(k)}$	$\left\|e^{(k)}\right\|_\infty$	$\left\|e^{(k)}\right\|_\infty / \left\|e^{(k-1)}\right\|_\infty$
0	0	0	1	...
1	0.5359	0.8231	0.4641	0.4641
2	0.9385	0.9798	0.0615	0.1325
3	0.9936	0.9980	0.0064	0.1047
↓	↓	↓	↓	↓
∞	1	1	0	$\rho(B_\omega) = \omega^* - 1 = 0.0718$

Hence optimal SOR converges faster than GS.

Help for computing project 1:

- Eigenvalues of the D_N matrix :
  - Start with small N, find characteristic polynomial, look for patterns
  - Use Matlab, make tables of eigenvalues, or plot eigenvalues (`real(x)`, `im(x)`, `abs(x)`).
  - What are $\max|\lambda|$ and $\min|\lambda|$ as functions of N?
- BVP notes :
  - Don't forget the ϵ! Remember: $y'' + q(x)y = r(x) \Rightarrow q(x) = \frac{1}{\epsilon}$.
  - Problem 1 and in-class discussion : Dirichlet boundary conditions. Problem 2: mixed BC, Dirichlet at $x = 0$, Neumann at $x = 1$.

    With Dirichlet conditions we are able to prescribe the boundary values w_0 and w_{n+1} in advance; with Neumann boundaries, we have to solve additional equations for w_0 and w_{n+1}; therefore, we augment the matrix with 2 additional equations, one for w_0 and one for w_{n+1}.

    Discretizing the grid with x_i, $i = 1, \ldots, n$ as the interior points and $x_0 = 0$, $x_{n+1} = 1$ gives a matrix of size $(n+2) \times (n+2)$.
  - Use the tridiagonal solver (it's fast!)

Example 13: Young's theorem and 2-pt BVP.

We have seen how the centered approximation of a 2nd derivative $-f''(x) \approx -D_+D_-f(x)$ gives rise to the matrix

$$A_h = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{pmatrix} : \text{dimension } n \times n \text{ with } h = \tfrac{1}{n+1}$$

Question 8: Can we use SOR with such matrices? A_h is clearly tridiagonal and symmetric. Is it positive definite? Yes. Maybe we'll show that in homework.

Definition 8. A matrix is diagonally dominant if

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|,$$

i.e., the magnitude of the diagonal element on each row is greater than the sum of the magnitudes of all other elements in that row.

Definition 9. A matrix is A is symmetric positive definite if

1. $A^T = A$
2. $x^T A x > 0$ for all $x \neq 0$

Theorem 10. *If a matrix A is **either** diagonally dominant **or** symmetric positive definite, then A is*

1. *nonsingular*
2. *Gaussian elimination (or LU factorization) can be performed without pivoting*

Theorem 11. *Let A be a symmetric matrix. If all of its eigenvalues are positive, then A is positive definite.*

Theorem 12. *Let A be a symmetric matrix. If A is strictly diagonally dominant and $a_{ii} > 0$ for all i, then A is positive definite.*

The eigenvalues of $A_h = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{pmatrix}$ with $h = \frac{1}{n+1}$

are given by $\lambda_p = \frac{2}{h^2}(1 - \cos(p\pi h))$ for $p = 1, 2, \ldots, m$ $\Rightarrow$ they are all positive.

△

Return to iterative methods. The real advantage of iterative methods, as compared to direct methods, occurs for BVPs in more than one dimension.

3 Two-dimensional BVP

Text: section 9.1

Problem: A metal plate has the shape of a square. The plate is heated by internal sources and the edges of the plate are held at given temperatures. Find the temperature at points inside the plate.

$D = \{(x, y) : 0 \leq x, y \leq 1\}$: plate domain

$\phi(x, y)$: plate temperature

$f(x, y)$: heat sources, $\quad g(x, y)$: boundary temperature

Then $\phi(x, y)$ satisfies the following two equations:

$$-\nabla^2\phi = -\left(\frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2}\right) = f \text{ for } (x, y) \text{ in } D : \underline{\text{Poisson equation}}$$

$\uparrow$

Laplace operator

$\phi(x, y) = g$ for (x, y) on ∂D : Dirichlet boundary condition.

Note: Lots of applications: field theory, quantum mechanics, fluid vorticity, electric potentials, temperature, etc.

Finite difference scheme

$h = \dfrac{1}{n+1}$: mesh size, n = number of points in each direction

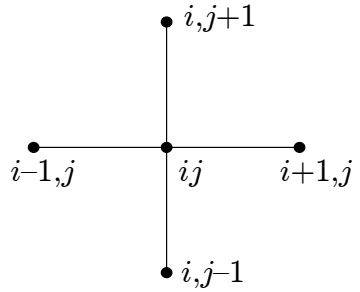
$(x_i, y_j) = (ih, jh), \quad 0 \le i, j \le n+1$: grid points, or mesh points

$\phi(x_i, y_i)$: exact solution

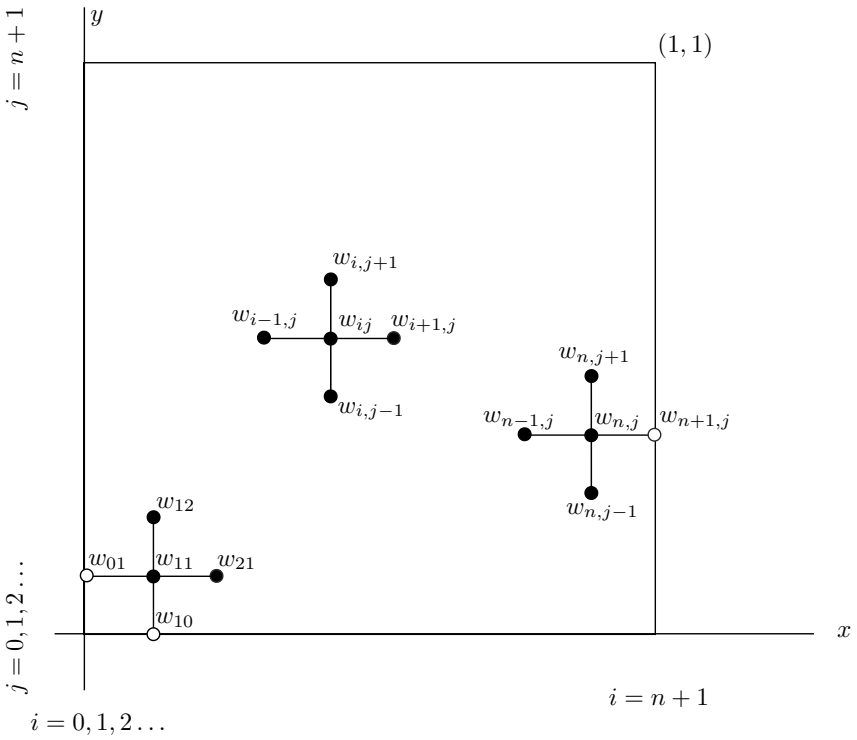
w_{ij} : approximation

To approximate the Laplace operator, we apply centered differences in each direction so that

$$-\left(D_+^x D_-^x + D_+^y D_-^y\right) w_{ij} \approx -\nabla^2\phi(x_i, y_j)$$



5 point stencil



The discrete system becomes

$-\left(D_+^x D_-^x + D_+^y D_-^y\right) w_{ij} = f_{ij}$ in the interior of the domain

$$-\frac{1}{h^2}\left(w_{i-1,j} - 2w_{ij} + w_{i+1,j} + w_{i,j-1} - 2w_{ij} + w_{i,j+1}\right) = f_{ij}$$

$$\frac{1}{h^2}\left(4w_{ij} - w_{i-1,j} - w_{i+1,j} - w_{i,j-1} - w_{i,j+1}\right) = f_{ij}$$

Now we look at the boundary:

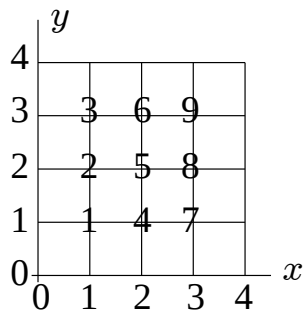
$$(i,j) = (1,1) \Rightarrow \frac{1}{h^2}\left(4w_{11} - w_{0,1} - w_{2,1} - w_{1,0} - w_{1,2}\right)$$

$$\Rightarrow \frac{1}{h^2}\left(4w_{11} - w_{21} - w_{12}\right) = f_{11} + \frac{1}{h^2}\left(g_{01} + g_{10}\right)$$

Question 9: We have a 2d domain, but we want to write the solution as a 1d vector : $\boldsymbol{w} = [w_0, w_1, \ldots, w_m]^T$. How?

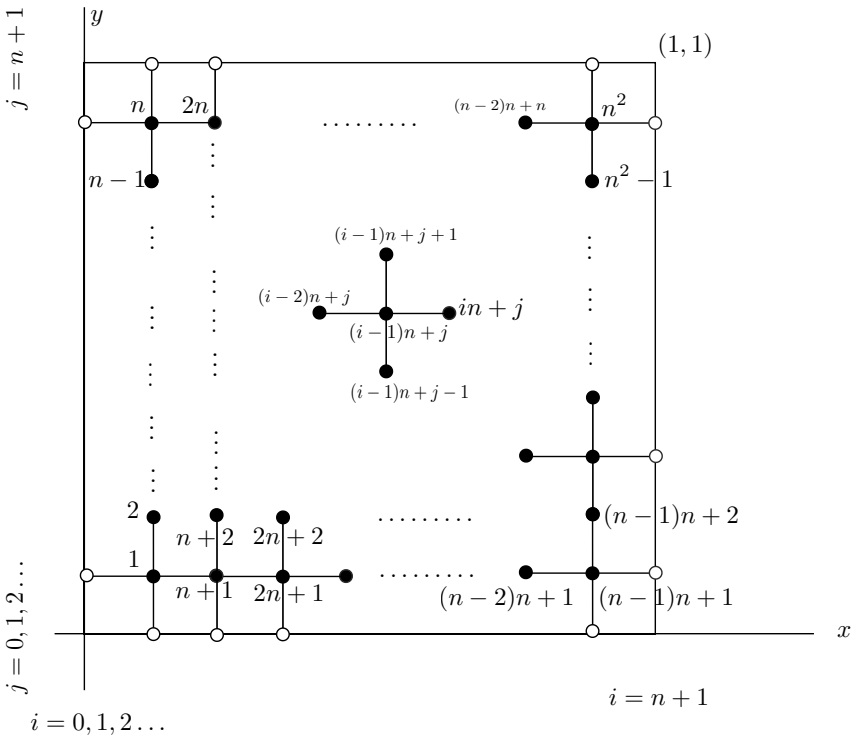
Impose and ordering on the mesh points (reorganize the i, j indices into a 1d list).

Example 14: $n = 3$, $h = \frac{1}{4}$, column ordering



The particular ordering is up to you: Matlab and Fortran prefer column-major order. C/C++ prefer row-major order. Parallel implementations are best ordered in a way that fits the computational topology. Some orderings create matrices with more desirable structure (e.g. sparse banded matrices).

Column numbering: $k = (i-1)n + j$ so that $W_k = w_{ij}$.



Now we can put the w_{ij} equations in matrix form. For $n = 3$:

1	2	3	4	5	6	7	8	9
w_{11}	w_{12}	w_{13}	w_{21}	w_{22}	w_{23}	w_{31}	w_{32}	w_{33}
4	-1		-1					
-1	4	-1		-1				
	-1	4			-1			
-1			4	-1		-1		
	-1		-1	4	-1		-1	
		-1		-1	4			-1
			-1			4	-1	
				-1		-1	4	-1
					-1		-1	4

This structure is constant as $h \to 0$ and $n \to \infty$:

$$A_h w_h = f_h, \quad A_h = \frac{1}{h^2} \begin{pmatrix} T & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -I \\ & & & -I & T \end{pmatrix}$$

$T : n \times n$, symmetric, tridiagonal, positive definite. $I : n \times n$ identity matrix

$\Rightarrow A : n^2 \times n^2$: block tridiagonal, symmetric, positive definite (proof omitted) and the theorems for iterative methods apply.

Thursday, 10/10/13

Question 10: Do the iterative methods converge (is $\|B\| < 1$ in some matrix norm, where B is one of the iteration matrices associated with Jacobi, Gauss-Seidel, or SOR)?

Theorem 13. *Let A be a strictly diagonally dominant matrix and B_J be the Jacobi iteration matrix associated with A. Then $\|B\|_\infty < 1$ and the Jacobi method $x^{(k+1)} = B_J x^{(k)} + c$ is guaranteed to converge for any initial guess $x^{(0)}$.*

Proof.

$B_J = -D^{-1}(L+U) \Rightarrow \|B_J\|_\infty \leq \left\|D^{-1}\right\|_\infty \cdot \|L+U\|_\infty$

$$\|L+U\|_\infty = \max_i \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \quad \|D\|_\infty = \max_i |a_{ii}|, \quad \left\|D^{-1}\right\|_\infty = \frac{1}{\|D\|_\infty}$$

Since A is strictly diagonally dominant, $\|D\|_\infty > \|L+U\|_\infty$,

$\Rightarrow \|B_J\|_\infty \leq \frac{\|L+U\|_\infty}{\|D\|_\infty} < 1$ □

Notes:

- The spectral radii of B_{GS} and B_ω are smaller than the spectral radius of B_J, so Gauss-Seidel and SOR also converge.

- We needed the matrix to show that the methods converge, but we don't have to use the matrix to perform the iterations. It's often more straightforward to apply the methods element-by-element across the mesh. In fact, since the matrix is sparse, it can be much more efficient to use the mesh element form (which avoids constructing the whole matrix... unless you learn about sparse matrix storage and manipulation algorithms).

3.1 Iterative methods for 2D Poisson problem

The general equation for the Poisson problem using the finite difference scheme defined in the previous section is $\frac{1}{h^2}\left(4w_{ij} - w_{i-1,j} - w_{i+1,j} - w_{i,j-1} - w_{i,j+1}\right) = f_{ij}$.

Solve for w_{ij} :
$$w_{ij} = \frac{1}{4}\underbrace{\left(w_{i-1,j} + w_{i+1,j} + w_{i,j-1} + w_{i,j+1}\right)}_{-(L+U)} - \frac{h^2}{4}\underbrace{f_{ij}}_{b}$$
$$\uparrow \qquad\qquad\qquad\qquad\qquad\qquad \uparrow$$
$$D$$

recall: Jacobi method $Dx^{(k+1)} = -(L+U)x^{(k)} + b$, hence,

Jacobi :
$$w_{ij}^{(k+1)} = \frac{1}{4}\left(w_{i-1,j}^{(k)} + w_{i+1,j}^{(k)} + w_{i,j-1}^{(k)} + w_{i,j+1}^{(k)}\right) - \frac{h^2}{4}f_{ij}$$

Similarly,

Gauss-Seidel :
$$w_{ij}^{(k+1)} = \frac{1}{4}\left(w_{i-1,j}^{(k+1)} + w_{i+1,j}^{(k)} + w_{i,j-1}^{(k+1)} + w_{i,j+1}^{(k)}\right) - \frac{h^2}{4}f_{ij}$$

SOR :
$$w_{ij}^{(k+1)} = (1-\omega)w_{ij}^{(k)} + \frac{\omega}{4}\left(w_{i-1,j}^{(k+1)} + w_{i+1,j}^{(k)} + w_{i,j-1}^{(k)} + w_{i,j+1}^{(k)}\right) - \omega\frac{h^2}{4}f_{ij}$$

Example 15: Temperature distribution on a metal plate, no internal sources.

no heat sources : $f(x,y) = 0 \Rightarrow \phi_{xx} + \phi_{yy} = 0$

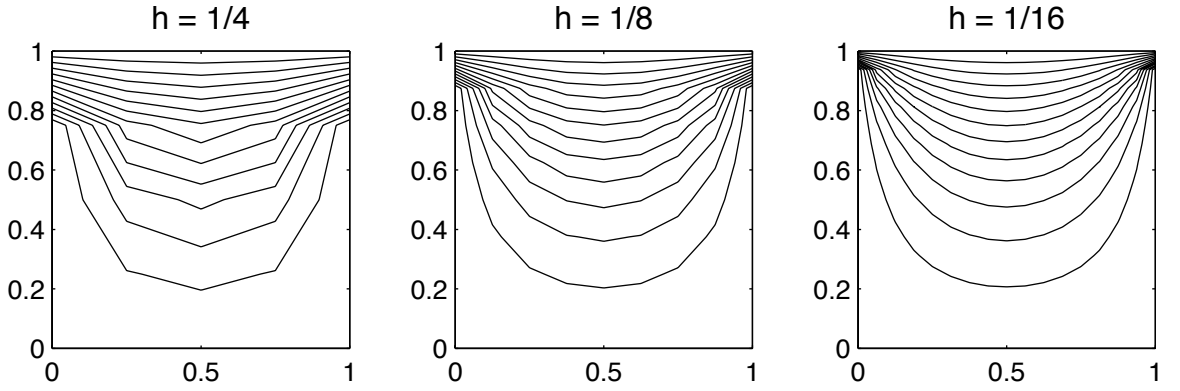
boundary conditions : $\phi(x,1) = 1, \quad \phi(x,0) = \phi(0,y) = \phi(1,y) = 0$

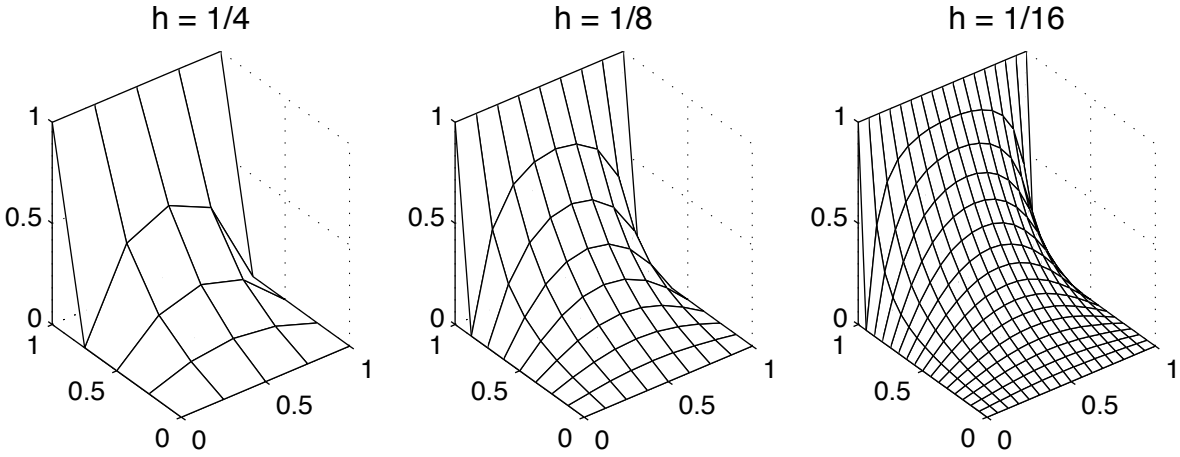
finite difference scheme: $4w_{ij} - w_{i+1,j} - w_{i-1,j} - w_{i,j+1} - w_{i,j-1} = 0$

Let's take a look at the iterative methods' performance on this problem.

Stopping criteria : If $\frac{\|r_k\|_\infty}{\|r_{k-1}\|_\infty} < \text{tol}$, or $k = k_{\max} = 3000$ and $\text{tol} = 1 \times 10^{-4}$.

Matlab demo

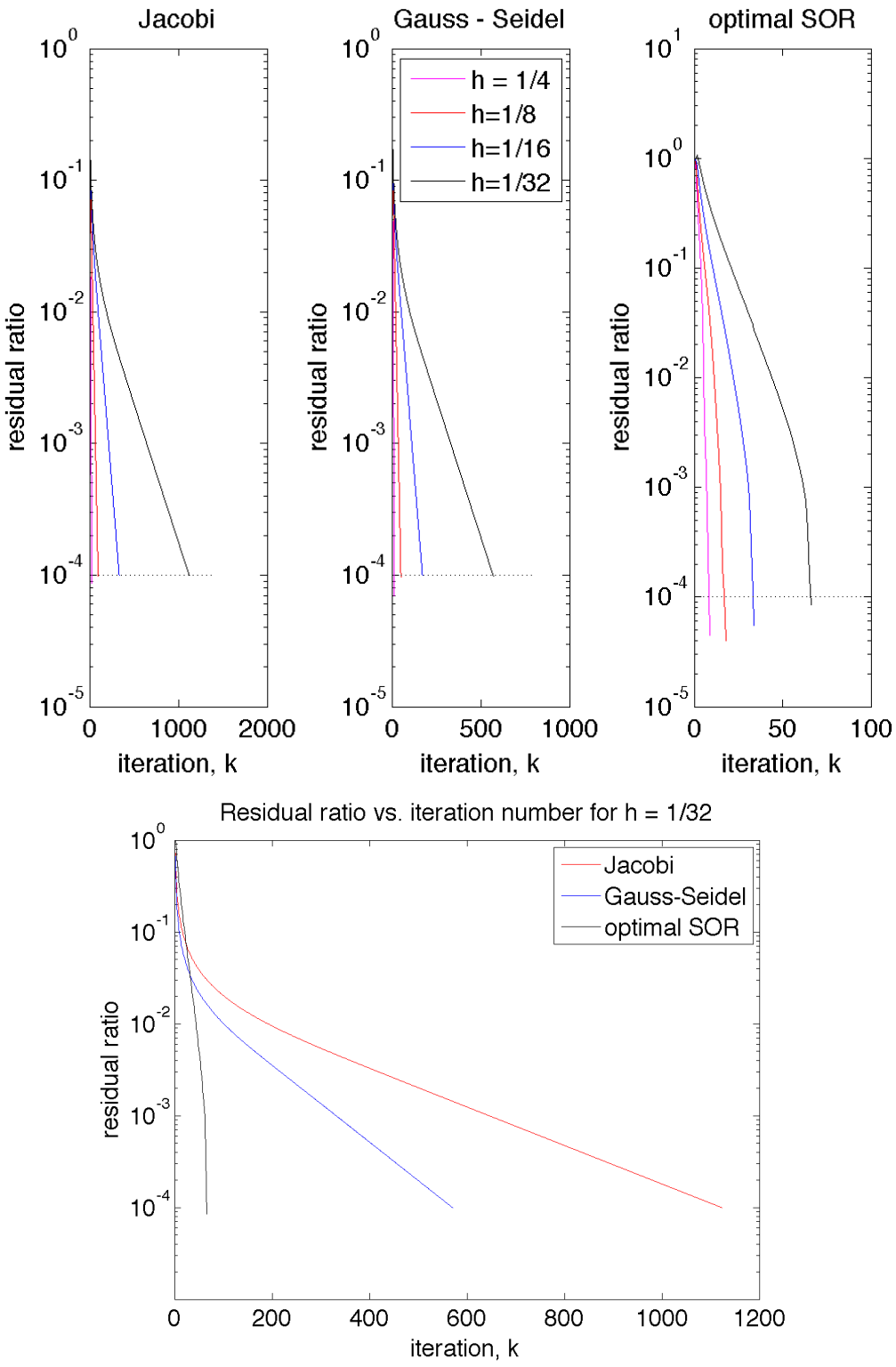




	h	k	$\|r_k\|_\infty / \|r_{k-1}\|_\infty$	$\rho(B)$
	1/4	26	0.70711	0.70711
Jacobi:	1/8	96	0.92388	0.92388
	1/16	334	0.98078	0.98078
	1/32	1123	0.99518	0.99518

	h	k	$\|r_k\|_\infty / \|r_{k-1}\|_\infty$	$\rho(B)$
	1/4	15	0.5000	0.5000
Gauss-Seidel:	1/8	51	0.85355	0.85355
	1/16	172	0.96194	0.96194
	1/32	571	0.99039	0.99039

	h	k	$\|r_k\|_\infty / \|r_{k-1}\|_\infty$	$\rho(B)$
	1/4	9	0.18192	0.17157
optimal SOR:	1/8	18	0.38999	0.44646
	1/16	34	0.49112	0.673513
	1/32	66	0.54676	0.82146



△

Notes:

- For a given value of h, GS requires fewer iterations than J, and SOR requires fewer than

GS, but regardless of the method used, more iterations are needed as the mesh size $h \to 0$. Hence decreasing h leads to smaller truncation error but larger computational cost.

- The ratio of successive residuals converges to the spectral radius of the iteration matrix as $k \to \infty$.

- Explicit formulas for the spectral radius $\rho(B)$ can be derived for this example:

  $\rho(B_J) = \cos \pi h \sim 1 - \frac{1}{2}\pi^2 h^2$ as $h \to 0$

  $\rho(B_{GS}) = \cos^2 \pi h \sim 1 - \pi^2 h^2$

  $$\rho(B_{\omega^*}) = \omega^* - 1 = \frac{2}{1 + \sqrt{1 - \rho(B_J)^2}} - 1 = \frac{2}{1 + \sqrt{1 - \cos^2 \pi h}} - 1 = \frac{2}{1 + \sin \pi h} - \frac{1 + \sin \pi h}{1 + \sin \pi h}$$

  $$\Rightarrow \rho(B_{\omega^*}) = \frac{1 - \sin \pi h}{1 + \sin \pi h} \sim \frac{1 - \pi h}{1 + \pi h} \sim 1 - 2\pi h$$

  These results confirm the observations that SOR is faster than GS and GS is faster than Jacobi. It also confirms that the methods converge more slowly as $h \to 0$, since $\rho(B) \to 1$ in that limit. SOR is least affected by this, followed by GS, and then Jacobi:
  $\rho(B_{\omega^*}) < \rho(B_{GS}) < \rho(B_J) < 1$ as $h \to 0$.

Now we consider what happens if we use Gaussian elimination to solve the same problem, $A_h w_h = f_h$.

Definition 14. Let A be an $n \times n$ matrix and $m < n$. If A has elements $a_{ij} = 0$ for all i, j such that $|i - j| > m$ then A is called a banded matrix with bandwidth m.

The matrix A_h is a banded matrix ($n = 3$, $h = \frac{1}{4}$ example):

$$A_h = \frac{1}{16} \begin{pmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{pmatrix}$$

in this case the bandwidth $m = 3$.

Notes:

1. When Gaussian elimination is performed on a banded matrix, the zeros inside the band become non-zero (this is called fill-in) but zeros outside the band are preserved. Hence we can adjust the limits of the Gaussian elimination loops to reduce the operation count from $O(n^3)$ to $O(mn^2)$.

   (Computing project B, problem 2, with D_2 for the $x = 1$ boundary)

2. Due to fill-in, more memory is required for L and U than for the original matrix A. This is a disadvantage in comparison with iterative methods of the form $x^{(k+1)} = Bx^{(k)} + c$ which preserve the sparsity of A.

4 Final comments on linear systems

Comparison of operation counts. Recall: the 2d BVP leads to a matrix A_h that has dimension $n^2 \times n^2$ with mesh size $h = \frac{1}{n+1}$. The bandwidth of A is $m = n$, and the typical equation is $\frac{1}{h^2}(4w_{ij} - w_{i+1,j} - w_{i-1,j} - w_{i,j+1} - w_{i,j-1})$.

1. Direct Gaussian elimination : $O(n^6)$ ops.

2. Banded Gaussian elimination : $O(n^2m^2) = O(n^4)$ ops
   Note: if you set up your sparse matrices with Matlab's sparse matrix commands (e.g., `spdiags`, `spy`, etc.) then Matlab's backslash operator will automatically take advantage of the sparsity and use banded elimination or other high performance sparse matrix operations. Try typing `help sparse` to see what's available.

3. Iterative methods
   stopping criterion : $\dfrac{\|r_k\|_\infty}{\|r_{k-1}\|_\infty} = \epsilon_{\text{tol}} \Rightarrow \rho(B)^k = \epsilon_{\text{tol}} \Rightarrow k = \dfrac{\log \epsilon_{\text{tol}}}{\log \rho(B)}$

   (a) Jacobi and Gauss-Seidel: $\rho(B) \sim 1 - ch^2 \Rightarrow \log \rho(B) \sim \log(1 - ch^2) \sim -ch^2$
   $\Rightarrow k \sim \dfrac{\log \epsilon_{\text{tol}}}{-ch^2} = O(n^2)$ iterations, cost per iteration $= O(n^2)$
   $\Rightarrow$ total cost $= O(n^4)$ ops

   (b) SOR : $\rho(B) \sim 1 - ch$
   $\Rightarrow k \sim \dfrac{\log \epsilon_{\text{tol}}}{-ch} = O(n)$ iterations, cost per iteration $= O(n^2)$
   $\Rightarrow$ total cost $= O(n^3)$ ops

4. Developments after SOR (1950 - present)

   (a) Multigrid methods: large h (fast, but low accuracy) + correction from grid with small h (slow, high accuracy)

   (b) Conjugate gradient method and GMRES : energy minimization

   (c) Preconditioning + iteration: $Ax = b \mapsto MAx = Mb$

   (d) Optimized software / hardware combinations. (e.g. BLAS, LAPACK)
   Example 16: As an example of what BLAS and LAPACK can do, let's compare the time taken to multiply 2 matrices, each of size $n \times n$, with a large n, using 3 different computational procedures in Fortran.

      i. Allocate memory for two $n \times n$ matrices A and B; use random numbers to fill the matrix elements.

      ii. Procedure 1: compute $C = AB$ using a triple do-loop:

```
do j=1,n
   do i=1,n
      do k=1,n
         c(i,j) = c(i,j) + a(i,k)*b(k,j)
      enddo
   enddo
enddo
```

iii. Compute $C = AB$ using the Fortran instrinsic procedure `C=MATMUL(A,B)`.

```
c2 = 0.0d0
c2 = MATMUL(a,b)
```

iv. Compute $C = AB$ using the BLAS/LAPACK subroutine provide by Intel's Math Kernel Library.

```
alpha = 1.0d0   ! these parameters are values defined
beta = 1.0d0    ! (and required by) BLAS/LAPACK
lda = n
ldb = n
ldc = n
c3 = 0.0d0
call DGEMM('N','N',n,n,n,alpha,a,lda,b,ldb,beta,c3,ldc)
```

v. Compile and link code using Intel compilers:
`ifort -O2 -mkl matrixMultiplyTiming.f90 -o matMulTiming.exe`

vi. Run code (example uses $n = 2000$, 2012 iMac Desktop 3.4GHz Intel Core i7 using 1 core in serial):

```
The triple do-loop takes    3.655 seconds.
The matmul(a,b) function takes    3.681 seconds.
The DGEMM subroutine takes    0.663 seconds.
```

△

(e) Parallel algorithms. In parallel with large matrices, "simple" operations such as transpose and matrix multiplication become less straightforward – if you want to do them quickly. Let M denote the fundamental operation count of an algorithm; for example, LU factorization of a matrix of size $n \times n$ would have $M = n^3$.

Perfect parallelism would reduce the time required for LU decomposition to $O(M/p)$, where p is the number of processors (EECS 586). Large, modern software on supercomputers can use $p \sim 10^3 - 10^5$ cores efficiently.