Chapter 5

SPARSE DIRECT METHODS

Sparse matrices are matrices containing mostly zeros—so many that we prefer not to store them all.

5.1 Banded Storage
5.2 Profile Storage
5.3 Sparse Storage
5.4 Reordering
5.5 Fast Direct Methods

Sparse symmetric positive definite matrices arise from

• network problems,
• discretized field problems.

For a field problem one wants to find temperature/potential/displacement/hydraulic head at each point in some region:

\[
\text{partial differential equation} \quad \rightarrow \quad \text{finite element method} \quad \rightarrow \quad \text{system of linear equations}
\]

Non-symmetric matrices arise in computational fluid dynamics, and semiconductor device simulation.
5.1 Banded Storage

\[ u = \text{temperature} \quad u^h \approx \text{temperature at gridpoints} \]

The five-point difference operator

\[- \frac{u^h(x + h, y) - 2u^h(x, y) + u^h(x - h, y)}{h^2} - \frac{u^h(x, y + h) - 2u^h(x, y) + u^h(x, y - h)}{h^2} = 0 \]

for interior mesh points \((x, y)\).

\[ \exists \text{ natural correspondence between unknowns and equations} \]
Suppose we order the unknowns thus:

\[
\begin{array}{ccccccc}
+ & + & + & + & + \\
10 & 11 & 12 & & & \\
+ & + & + & + & + \\
7 & 8 & 9 & & & \\
+ & + & + & + & + \\
4 & 5 & 6 & & & \\
+ & + & + & + & + \\
1 & 2 & 3 & & & \\
+ & + & + & + & + \\
+ & + & + & + & + \\
\end{array}
\]

Then

\[
\begin{align*}
\frac{4u_8 - u_7 - u_9 - u_5 - u_{11}}{h^2} &= 0, \\
\frac{4u_1 - 0 - u_2 - 1 - u_4}{h^2} &= 0.
\end{align*}
\]

\[
\begin{bmatrix}
4 & -1 & & & & & & & \\
-1 & 4 & -1 & & & & & & \\
& -1 & 4 & -1 & & & & & \\
& & -1 & 4 & -1 & & & & \\
& & & -1 & 4 & -1 & & & \\
\end{bmatrix}
\begin{bmatrix}
u_1 \\
u_2 \\
u_3 \\
u_4 \\
u_5 \\
u_6 \
\end{bmatrix}
= \begin{bmatrix} 1 \\
1 \\
0 \\
0 \\
0 \
0 
\end{bmatrix}
\]

The upper bandwidth is 3 and the lower bandwidth is 3. The (overall) bandwidth = 3 + 1 + 3 = 7. This is also a 4×4 block tridiagonal matrix with 3×3 blocks. Other orderings of
Definition A matrix $A$ for which

$$a_{ij} = 0 \text{ for } j < i - p \text{ or } j > i + q$$

has lower bandwidth $p$, upper bandwidth $q$, and bandwidth $p + 1 + q$:

- $p$ subdiagonals
- 1 diagonal
- $q$ superdiagonals
If $a_{ij}$ is stored in an array $A[n][p + 1 + q]$ how does one access $a_{ij}$?
Symmetric positive definite matrices

To do a Cholesky factorization $A = GG^\text{T}$,

1. change range of loops to avoid zeros (nonstored zeros),
2. then adjust indexing as above.

**Strictly diagonally dominant matrices.** An example is a tridiagonal matrix from a simple 2 point boundary value problem. One can do Gaussian elimination without pivoting.

| example | ×: element of reduced matrix | 0: multiplier |
**general matrices** Use partial pivoting. Here is an example for \( p = 2, q = 1 \) of “worst” case interchanges:

\[ F: \text{fill in} \]

*Fill in* is the introduction of a value not known to be zero into a nonstored zero. *More generally* reduced upper triangular matrix has same bandwidth as \( A \).

storage for \( A \): \( n(p + q + 1) \)

storage for decomposed \( A \):

\( n(2p + q + 1) \) **floats**

\( n - 1 \) **ints**

worst case:

row 1 \( \leftrightarrow \) row \( p + 1 \)

row 2 \( \leftrightarrow \) row \( p + 2 \)

\[ : \]
operation count—partial pivoting

$k$th stage: $p(p + q)$ mults. Ignoring end effects ($p, q \ll n$)

$p(p + q)n$ mults

<table>
<thead>
<tr>
<th></th>
<th>full matrix</th>
<th>banded matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>factorization</td>
<td>$n^3/3$</td>
<td>$n^2$</td>
</tr>
<tr>
<td>backsolve</td>
<td>$n^2$</td>
<td>$2np$</td>
</tr>
</tbody>
</table>

Operation count for case $q = p$

For Laplace’s equation on square $p = n^{1/2}$
no pivoting  partial pivoting
factorization  \( n^2 \)  \( 2n^2 \)
backsolve  \( 2n^{3/2} \)  \( 3n^{3/2} \)

tridiagonal

\[
\begin{bmatrix}
a_1 & c_1 & & \\
b_2 & a_2 & c_2 & \\
& \ddots & \ddots & \ddots \\
& & b_{n-1} & a_{n-1} & c_{n-1} \\
& & & b_n & a_n
\end{bmatrix}
\]

\[
A x = f
\]

\[
\text{forward\_elimination\_on\_matrix}()\
\begin{array}{l}
\text{for } k = 1, 2, \ldots, n-1 \text{ do } \\
\quad b_{k+1} = b_{k+1}/a_k; \\
\quad a_{k+1} = a_{k+1} - b_{k+1} \cdot c_k;
\end{array}
\]

\[
\text{forward\_elimination\_on\_f}(f)\
\begin{array}{l}
\text{for } k = 1, 2, \ldots, n-1 \text{ do } f_{k+1} = f_{k+1} - b_{k+1} \cdot f_k;
\end{array}
\]

\[
\text{back\_substitution}()\
\begin{array}{l}
\quad x_n = f_n/a_n; \\
\quad \text{for } k = n-1, n-2, \ldots, 1 \text{ do } x_k = (f_k - c_k \cdot x_{k+1})/a_k;
\end{array}
\]

**Review questions**

1. What does it mean for the lower bandwidth of a matrix to be \( p \)? the upper bandwidth to be \( q \)?

2. If one uses banded storage to store a matrix \( A \) of lower bandwidth \( p \) and upper bandwidth \( q \), in which element of the array is \( a_{ij} \) stored? (Assume that array indexing starts at 1.)

3. For a symmetric positive definite matrix \( A \) of bandwidth \( 2p + 1 \), which elements do we store? Which elements of its Cholesky factor \( G \) do we store?

4. What is fillin?
5. How much memory would you need to solve a tridiagonal system using Gaussian elimination with partial pivoting?

6. For a matrix of lower bandwidth $p$ and upper bandwidth $q$, what fillin might be introduced by Gaussian elimination without pivoting? by Gaussian elimination with partial pivoting?

7. For an $n$ by $n$ matrix of lower bandwidth $p$ and upper bandwidth $q$, approximately how many multiplications are required for the factorization stage of Gaussian elimination without pivoting? by its backsolve stage? by the factorization stage of Gaussian elimination with partial pivoting? by its backsolve stage?

8. Construct the algorithm for Gaussian elimination without pivoting for a tridiagonal matrix.

**Exercises**

1. Complete the following algorithm for performing Gaussian elimination with partial pivoting on a tridiagonal matrix. The original matrix and the reduced matrix are stored as

\[
\begin{bmatrix}
  a_1 & c_1 \\
  b_2 & a_2 & c_2 \\
  & \ddots & \ddots & \ddots \\
  & & \ddots & \ddots & \ddots \\
  & & & b_{n-1} & a_{n-1} & c_{n-1} \\
  & & & b_n & a_n
\end{bmatrix}
\quad \text{and} \quad
\begin{bmatrix}
  a_1 & c_1 & d_1 \\
  a_2 & c_2 & d_2 \\
  & \ddots & \ddots & \ddots \\
  & & \ddots & \ddots & \ddots \\
  & & & a_{n-1} & c_{n-1} \\
  & & & a_n
\end{bmatrix}
\]

with multipliers in $b_2, b_3, \ldots, b_n$ and row interchange information in $r_1, r_2, \ldots, r_{n-1}$.

\[
\text{for } k = 1, 2, \ldots, n - 2 \text{ do } \{
\]
\[
  r_k = k; \quad d_k = 0;
\]
\[
  \text{if } (|b_{k+1}| > |a_k|) \{
  \]
\[
    \text{interchange } a_k \text{ and } b_{k+1};
  \]
\[
    \text{interchange } c_k \text{ and } a_{k+1};
  \]
\[
    \text{interchange } d_k \text{ and } c_{k+1};
  \]
\[
    r_k = k + 1;
  \]
\[
}\}
\]

131
\[ r_{n-1} = n - 1; \]
\[ \text{if } (|b_n| > |a_{n-1}|) \{ \]
\[ \quad \text{interchange } a_{n-1} \text{ and } b_n; \]
\[ \quad \text{interchange } c_{n-1} \text{ and } a_n; \]
\[ \quad r_{n-1} = n; \]
\[ \} \]
\[ b_n = b_n / a_{n-1}; \]
\[ a_n = a_n - b_n * c_{n-1}; \]

2. Given below is an algorithm for the Choleski factorization of a dense symmetric positive definite matrix \( A \):

\[
\text{for } j = 1, 2, \ldots, n \text{ do }
\begin{align*}
    a_{jj} &= (a_{jj} - \sum_{k=1}^{j-1} a_{jk}^2)^{1/2}; \\
    \text{for } i = j + 1, j + 2, \ldots, n \text{ do }
    a_{ij} &= (a_{ij} - \sum_{k=1}^{j-1} a_{ik}a_{jk}) / a_{jj};
\end{align*}
\]

Consider now a symmetric positive definite tridiagonal matrix whose subdiagonal elements are \( b_2, b_3, \ldots, b_n \) and whose diagonal elements are \( c_1, c_2, \ldots, c_n \). Create an algorithm (in a nonambiguous algorithmic language) which overwrites the arrays \( b \) and \( c \) with the Cholesky “triangle” of the tridiagonal matrix. Use no arrays other than \( b \) and \( c \) in your algorithm.

3. Let \( A \) be the matrix of the five point difference operator on a uniform square mesh of \( N^2 \) interior points. Show that \( A \) is nonsingular. This is usually done as follows:

Let \( u = [u_{11}, u_{21}, \ldots; u_{1J}, u_{2J}, \ldots; u_{IJ}] \) be a vector satisfying \( Au = 0 \). Let \( u_{IJ} \) be a component of \( u \) such that

\[ |u_{IJ}| = \|u\|_{\infty} = \max_{1 \leq i, j \leq N} |u_{ij}|. \]

If \( I \) or \( J = 1 \) or \( N \), then it follows that \( u_{IJ} = 0 \). (Show this.) Otherwise, it follows that the neighboring values

\[ |u_{I \pm 1, J}| = |u_{I, J \pm 1}| = \|u\|_{\infty}. \]

(Show this)

Clearly we can repeat this argument as many times as necessary to get that

\[ \|u\|_{\infty} = |u_{IJ}| = |u_{I+1, J}| = \cdots = |u_{NJ}| = 0. \]

Hence we have shown that the only solution of \( Au = 0 \) is \( u = 0 \), so \( A \) is nonsingular.
5.2 Profile\(^1\) Storage

Consider only symmetric positive definite matrices.

An example from ODEs with periodic BCs:

\[
\begin{bmatrix}
  d & -1 & \cdots & -1 \\
  -1 & d & \ddots & \cdots \\
  \vdots & \ddots & \ddots & \cdots \\
  -1 & \cdots & \cdots & -1 \\
\end{bmatrix}
\]

where \(d > 2\). For profile storage scheme: in each row store from 1st nonzero through diagonal:

\[
\begin{array}{cccc}
  d & -1 & d & -1 \\
  -1 & d & \ddots & \\
  \vdots & \ddots & \ddots & -1 \\
  -1 & 0 & \cdots & -1 & d \\
\end{array}
\]

\(^1\)also called envelope, skyline, variable bandwidth
Another less special example:

```
+ + + + + +
 6  8  9
+ + + + + +
 3  5  7
+ + + + + +
 1  2  4
+ + + + + +
```

Use an array of variable length arrays:

```
a = {{4}, {-1, 4}, {-1, 0, 4}, {-1, -1, 0, 4},
     {-1, 0, 0, 4}, {-1, -1, 0, 4}, {-1, -1, -1, 0, 4}, {-1, -1, -1, 4}};
```

How do we reference an element \( a_{ij} \) where \( j \leq i \)?

\[ a[i] \text{ is } \{a_{i,j_{min}}, \ldots, a_{i,j}, a_{i,j+1}, \ldots, a_{ii}\} \text{ where } j_{min} = i - (a[i].\text{length} - 1). \]

The element \( a_{ij} \) is in \( a[i][j - j_{min}] \) if \( j \geq j_{min} \); otherwise \( a_{ij} = 0 \). (In C one can calculate \( a[i].\text{length} \) using pointer arithmetic.)

For a Cholesky factorization \( A = GG^T \), the Cholesky triangle \( G \) has same profile as \( A \).

For the 5 point difference operator on \( \sqrt{n} \times \sqrt{n} \) mesh, profile storage reduces storage by \( \frac{2}{3} \) and computation by \( \frac{1}{2} \) over banded storage.

Profile storage is commonly used for stiffness matrices arising from the application of the finite element method. In the finite element method the spatial domain is partitioned into triangular or rectangular elements. Unknown values are sought at nodes in the domain. The structure of the matrix is such that the \((i, j)\)th entry of the stiffness matrix is nonzero only if nodes \( i \) and \( j \) are in the same element of the spatial domain.
5.3 Sparse Storage

A possible storage scheme

\[ a = \{\{\times, \times, \times\}, \{\times, \times\}, \{\times\}, \{\times, \times\}, \{\times, \times\}\}; \]
\[ ja = \{\{1, 3, 5\}, \{1, 2\}, \{3\}, \{1, 4\}, \{2, 5\}, \{4, 6\}\}; \]

For a symmetric matrix . . . .

Additional storage needed for fill in. An example of no pivoting:

\[
\begin{bmatrix}
\times & \times & \times \\
0 & F & F \\
0 & F & F
\end{bmatrix}
\rightarrow
\begin{bmatrix}
\times & \times & \times \\
0 & \times & \times \\
0 & F & \times
\end{bmatrix}
\rightarrow
\begin{bmatrix}
\times & \times & \times \\
0 & 0 & \times \\
0 & 0 & \times
\end{bmatrix}
\]

et cetera.

**Review questions**

1. For general sparse storage of a matrix how would you access its \((i, j)\)th element?
2. Do symbolic Gaussian elimination on a sparse matrix to determine fillin.

**Exercises**

1. What is the sparsity pattern for the Cholesky triangle of a symmetric matrix whose lower half has the following sparsity pattern:

\[
\begin{bmatrix}
\times \\
\times \\
\times & \times \\
\times & \times & \times \\
\times & \times & \times \\
\end{bmatrix}
\]

Note: we leave blank those elements known to be zero; we put \(\times\)'s elsewhere.
2. For a matrix \( A \) with sparsity pattern

\[
A = \begin{bmatrix}
\times & \times & \times \\
\times & \times & \times \\
\times & \times & \times 
\end{bmatrix}
\]

determine the sparsity patterns of its L, U factors. (Assume no permutation of the rows or columns of \( A \).)

### 5.4 Reordering

Problem: how to number variables/equations so as to minimize bandwidth or profile or fill in.

If a permutation matrix \( P \) re-orders (i.e., re-numbers) the equations and \( Q \) re-orders variables, then the new matrix is \( PAQ \).

We consider symmetric positive definite matrices first. To preserve symmetry, choose \( Q = P^T \), i.e., maintain the same correspondence between equations and variables.

The sparsity structure of a symmetric matrix can be represented by an undirected graph.

- **graph**: nodes, edges
- **nodes**: 1, 2, \ldots, \( n \)
- **edges**: \( \{i, j\} | i \neq j \) and \( a_{ij} \neq 0 \)

Nodes \( i \) and \( j \) are *adjacent* if \( \{i, j\} \) is an edge. The fillin that occurs during the elimination of the \( k \)th unknown can be represented graphically by eliminating node \( k \) from the graph and connecting all unconnected nodes that are adjacent to node \( k \). Each new edge corresponds to a filled in matrix element.

\[
PAP^T \iff \text{relabeling the nodes}
\]

**reverse Cuthill-McKee ordering** A very successful algorithm for relabeling for profile storage is the *reverse Cuthill-McKee algorithm*:
label a node with an $n$;
\[ l = n - 1; \]
\textbf{for} $k = n, n - 1, \ldots, 2$ \textbf{do}
\begin{itemize}
  \item \textbf{while} (there are unlabeled nodes adjacent to $k$) \{ 
    \begin{itemize}
      \item label with an $l$ a node of lowest degree which is unlabeled & adjacent to $k$;
      \item $l = l - 1$;
    \end{itemize}
  \}
\end{itemize}

The \textit{degree} of a node is the number of adjacent nodes.

\textbf{nested dissection ordering}  Consider a $64 \times 64$ matrix with the following graph:

Order the nodes in the left part 1, 2, \ldots, 32. Order the nodes in the right part 33, 34, \ldots, 56. Order the nodes in the separator 57, 58, \ldots, 64. Apply this idea recursively to the left and right part; hence “nested.”
If one programs the Cholesky factorization to exploit all sparsity, then for $\sqrt{n} \times \sqrt{n}$ mesh

$\text{storage} = \frac{31}{8} n \log_2 n + O(n),$

$\text{mults} = \frac{829}{84} n^{3/2} + O(n \log_2 n).$


**minimum degree ordering**  This ordering is most popular for direct solvers for symmetric positive definite systems; the previous two orderings are preferred for iterative methods. At each step of Gaussian elimination, the pivot is chosen to correspond to a node of minimum degree in the elimination graph.

For a general matrix we may want to pivot (interchange rows and/or columns) for *numerical stability* (avoid small pivots). For a sparse matrix we also want to pivot to *preserve sparseness* e.g.,

\[
\begin{bmatrix}
\times & \times \\
\times & 0
\end{bmatrix} \quad \rightarrow \quad \text{fill in} \quad \begin{bmatrix}
\times & 0 \\
\times & \times
\end{bmatrix} \quad \text{no fill in}
\]

—2 conflicting considerations. Compromise: threshold pivoting—pivot for sparseness subject to the condition that the pivot is not too small:

$|\text{pivot}| \geq \tau |\text{largest possible pivot}|,$

e.g., $\tau = 0.1$. The choice $\tau = 1$ is the usual partial pivoting.
Review questions

1. What is the (primary) purpose of reordering a banded matrix? a general sparse matrix?
2. Explain how to construct nodes and edges of a graph representing the sparsity structure of a symmetric matrix.
3. What happens to the graph representing the sparsity structure of a symmetric matrix \( A \) if an unknown is eliminated from the linear system \( Ax = b \)? Interpret the change in the graph as a change in the matrix.
4. What matrix operation corresponds to a relabeling of the nodes of a graph representing the sparsity structure of a symmetric matrix?
5. What is the degree of a node? **the number of adjacent nodes**
6. What is a separator of a connected graph?
7. Explain the idea of nested dissection.
8. Illustrate the idea of nested dissection on an example.
9. If the graph of a symmetric positive definite matrix is a \( \sqrt{n} \) by \( \sqrt{n} \) mesh, the number of operations required for nested dissection to compute a Cholesky factorization is proportional to what power of \( n \)?
10. For a general sparse matrix what are two reasons for reordering?
11. What is threshold pivoting? When does it reduce to partial pivoting?

Exercises

1. Below is given a graph representing the sparsity structure of an 8 by 8 symmetric positive definite matrix. If one applies Cholesky decomposition (or equivalently Gaussian elimination), how many of the zeros in the lower triangular part of the matrix fill in? Show your work.
2. Consider a symmetric positive definite matrix $A$ having the following sparsity structure:

$$
\begin{bmatrix}
* & * & * & * \\
* & * & * & * \\
* & * & * & * \\
* & * & * & *
\end{bmatrix}
$$

By using a graph to represent the sparsity structure of $A$, determine a permutation matrix $P$ such that the bandwidth of $PAP^T$ is minimized.

3. What would be the sparsity pattern of the following symmetric matrix if we were to re-order rows and columns using the reverse Cuthill-McKee algorithm?

$$
\begin{bmatrix}
* & * & * & * \\
* & * & * & * \\
* & * & * & * \\
* & * & * & *
\end{bmatrix}
$$

5.5 Fast Direct Methods

odd/even reduction for tridiagonal matrices  

augmented matrix

$$
\begin{bmatrix}
* & * & * & * \\
* & * & * & * \\
* & * & * & * \\
* & * & * & *
\end{bmatrix} \quad \begin{bmatrix}
* \\
* \\
* \\
* 
\end{bmatrix}
$$

pivots
for $k = 2, 4, 6, 8$ do

use equation $k$ to eliminate $x_k$ from equations $k - 1$ and $k + 1$;
use equation 10 to eliminate $x_{10}$ from equation 9;

\[
\begin{bmatrix}
  x & 0 & x \\
  x & x & x \\
  x & 0 & 0 & x \\
  x & x & x \\
  x & 0 & 0 & x \\
  x & x & x \\
  x & 0 & 0 & x \\
  x & x & x \\
  x & x & x \\
\end{bmatrix}
\begin{bmatrix}
  x \ \\
  x \ \\
  x \ \\
  x \ \\
  x \ \\
  x \ \\
  x \ \\
  x \ \\
  x \ \\
\end{bmatrix}
\]

Note that equations 1, 3, 5, 7, 9 involve only $x_1, x_3, x_5, x_7, x_9$—a reduced problem. If we can solve this, we can get $x_2, x_4, x_6, x_8, x_{10}$ from equations 2, 4, 6, 8, 10. The reduced problem is

\[
\begin{bmatrix}
  x & x \\
  x & x & x \\
  x & x & x \\
\end{bmatrix}
\begin{bmatrix}
  x \ \\
  x \ \\
  x \ \\
\end{bmatrix}
\]

Eliminate $x_2$ and $x_4$ using equation 2 and 4:

\[
\begin{bmatrix}
  x & 0 & x \\
  x & x & x \\
  x & 0 & 0 & x \\
  x & x & x \\
  x & 0 & x & x \\
\end{bmatrix}
\begin{bmatrix}
  x \ \\
  x \ \\
  x \ \\
  x \ \\
  x \ \\
\end{bmatrix}
\]

If we can solve equations 1, 3, 5, then we can also get $x_2, x_4$:

\[
\begin{bmatrix}
  x & x \\
  x & x & x \\
\end{bmatrix}
\begin{bmatrix}
  x \ \\
  x \ \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
  x & 0 & x \\
  x & x & x \\
  x & 0 & x \\
\end{bmatrix}
\begin{bmatrix}
  x \ \\
  x \ \\
  x \ \\
\end{bmatrix}
\]

Solve this.
**block cyclic reduction**  This is a generalization of odd/even reduction to block tridiagonal matrices, e.g.,

\[
\begin{bmatrix}
D & F & 0 \\
F & D & \ddots \\
\ddots & \ddots & F \\
0 & F & D
\end{bmatrix}
\]

where \(DF = FD\), e.g.,

\[
D = \begin{bmatrix}
4 & -1 & 0 \\
-1 & 4 & \ddots \\
\ddots & \ddots & -1 \\
0 & -1 & 4
\end{bmatrix}
\]

and \(F = -I\).

**Fourier analysis**  If we have a known spectral decomposition \(A = QAQ^T\), we can write

\[
A^{-1} = QA^{-1}Q^T.
\]

Multiplication by \(Q^T\) and \(Q\) each require \(n^2\) multiplies normally, but for special \(Q\), the cost is only \(O(n \log n)\) using the FFT (assuming \(n\) has a factorization into small primes). The FFT is based on a factorization of \(Q\) into \(O(\log n)\) sparse matrices.

**fast direct methods**

- block Cyclic Reduction  \(O(n \log_2 n)\) flops
- Fourier Analysis  \(O(n \log_2 n)\) flops
- combination, FACR(l)  \(O(n \log \log n)\) flops

These are applicable to discretized *separable* BVP’s, i.e., separation of variables. Software: FISHPACK.

**Review questions**

1. What is the idea of odd/even reduction for a tridiagonal matrix?
2. What do we call the generalization of odd/even reduction to a block tridiagonal matrix?
3. What is the operation count of a fast direct method for solving systems of linear equations? Name two examples of fast direct methods.
Chapter 6

ITERATIVE METHODS

6.1 Stationary Methods
6.2 Convergence
6.3 Gradient Methods
6.4 GMRES
6.5 Multigrid Methods

We seek to solve $Ax = b$ where $A$ large and sparse. $x^{(0)}, x^{(1)}, x^{(2)}, \ldots \rightarrow x$. These methods are very economical with storage. They are the only way to solve discretized PDEs in 3 space variable, e.g.,

Poisson’s equation $-\frac{\partial^2 y}{\partial x^2} - \frac{\partial^2 y}{\partial y^2} - \frac{\partial^2 y}{\partial z^2} = f(x, y, z)$.

6.1 Stationary methods

Example

```
0 0 0
| | |
0----0
| | |
0----0
| | |
0----0
| | |
0----0
| | |
0----0
| | |
1 1 1
```

143
Away from a boundary

\[
\frac{1}{h^2} \begin{pmatrix} - u_{i+3} \\ - u_{i-1} + 4u_i - u_{i+1} \\ - u_{i-3} \end{pmatrix} = 0
\]

*Note.* This is not a matrix but a two dimensional arrangement of an expression. Matrices are enclosed in brackets.

\[
u_i = \frac{1}{4} \begin{pmatrix} u_{i+3} \\ u_{i-1} + u_{i+1} \\ u_{i-3} \end{pmatrix}
\]

Jacobi’s method \( u_i^{\text{new}} = \frac{1}{4} \begin{pmatrix} u_{i+3} \\ u_{i-1} + u_{i+1} \\ u_{i-3} \end{pmatrix}^{\text{old}} \)

Initial guess

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1
\end{array}
\]

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
1 & 1 & 1 & 1
\end{array}
\]

\[
\begin{array}{cccc}
0.0625 & 0.625 & 0.625 & 0 \\
0.312 & 0.375 & 0.312 & 0 \\
1 & 1 & 1 & 1
\end{array}
\]

This is also called the method of simultaneous displacements. (Note \(0.375 \div 0.0625 \div 0.0625 = 0.5\) for round-to-even but 0.501 for rounding.)

Gauss-Seidel method \( u_i^{\text{new}} = \frac{1}{4} \begin{pmatrix} u_{i-1} \\ u_{i-3} \end{pmatrix}^{\text{new}} + \frac{1}{4} \begin{pmatrix} u_{i+3} \\ u_{i+1} \end{pmatrix}^{\text{old}} \)

Assuming we sweep from the lower left to the upper right (↗),

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0.0039 & 0.00778 & 0.0102 & 0 \\
0.0156 & 0.0272 & 0.0332 & 0 \\
0.0625 & 0.0935 & 0.106 & 0 \\
0.25 & 0.312 & 0.328 & 0 \\
1 & 1 & 1 & 1
\end{array}
\]

etc.

144
This is also called the method of successive displacements. Gauss-Seidel requires only half as much memory because $u_{\text{new}}^h$ can overwrite $u_{\text{old}}^h$. For this problem Gauss-Seidel converges twice as fast as Jacobi. We call these relaxation methods because at each point we relax one variable so as to satisfy an equation.

I recommend this method to you for imitation. You will hardly ever again eliminate directly, at least not when you have more than 2 unknowns. The indirect procedure can be done while half asleep, or while thinking about other things.

Gauss, December 26, 1823

In general for $\sum_j a_{ij}x_j = b_i$, solve for $x_i$ in $i$th equation:

$$x_i = \frac{1}{a_{ii}} \left( b_i - \sum_{j \neq i} a_{ij}x_j \right).$$

Iterate

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j \neq i} a_{ij}x_j^{(k)} \right).$$

This can be expressed in matrix notation if we write $A = L + D + U$ where $L$ is strictly lower triangular, $D$ is diagonal, and $U$ is strictly upper triangular:

$$Ax = b, \quad (L + D + U)x = b, \quad x = D^{-1}(b - (L + U)x),$$

or

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

Gauss-Seidel is

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left( b_i - \sum_{j < i} a_{ij}x_j^{(k+1)} - \sum_{j > i} a_{ij}x_j^{(k)} \right)$$

or

$$x^{(k+1)} = D^{-1}(b - Lx^{(k+1)} - Ux^{(k)}).$$

**residual correction paradigm** Suppose we have $C \approx A^{-1}$. We can get $x^{(1)} = C \cdot b$. We have

remainder = $A^{-1} \cdot$ residual.

Therefore

$$x = x^{(1)} + A^{-1}(b - Ax^{(1)}).$$

How about $x^{(2)} = x^{(1)} + C \cdot (b - Ax^{(1)})$? Consider as an example $3x = 1$, $C = 0.3$. The iteration is

$$x^{(k+1)} = x^{(k)} + 0.3(1 - 3x^{(k)}),$$

145
and the iterates are \( x^{(0)} = 0, \ x^{(1)} = 0.3, \ x^{(2)} = 0.33, \ x^{(3)} = 0.333, \ldots \) In this framework

\[
\text{Jacobi} : \quad x^{(k+1)} = x^{(k)} + D^{-1}(b - Ax^{(k)}) \\
\text{Gauss-Seidel} : \quad x^{(k+1)} = x^{(k)} + (L + D)^{-1}(b - Ax^{(k)})
\]

Viewed this way, we see that the correction is some approximation to \( A^{-1} \) times the residual. Also iterative improvement fits in this framework.

For further insight consider the simple example

\[
-\frac{d^2}{dx^2} u = 0, \quad 0 < x < 1, \quad u(0) = 0, \quad \frac{d}{dx} u(1) = 0
\]

A numerical solution \( u_j \approx u(jh) \) is obtained by solving

\[
-\frac{1}{h} \left( \frac{u_{j+1} - u_j}{h} - \frac{u_j - u_{j-1}}{h} \right) = 0, \quad 1 \leq j \leq J - 1, \quad u_0 = 0, \quad \frac{u_J - u_{J-1}}{h} = 0.
\]

The value of \( u_0 \) is obviously known:

\( u_0^{\text{new}} = u_0^{\text{old}} = 0 \)

Solving \( j \)th equation for \( j \)th unknown:

Jacobi

\[
\begin{align*}
\ u_j^{\text{new}} &= \frac{1}{2} (u_{j-1}^{\text{old}} + u_{j+1}^{\text{old}}) \\
\ u_j^{\text{new}} &= u_{j-1}^{\text{old}}
\end{align*}
\]

Gauss-Seidel

\[
\begin{align*}
\ u_j^{\text{new}} &= \frac{1}{2} (u_{j-1}^{\text{new}} + u_{j+1}^{\text{old}}) \\
\ u_j^{\text{new}} &= u_{j-1}^{\text{new}}
\end{align*}
\]

overdo it by 50 %

\[
\begin{align*}
\ u_j^{\text{new}} &= u_j^{\text{old}} + 1.5 \left( \frac{1}{2} u_{j-1}^{\text{new}} + \frac{1}{2} u_{j+1}^{\text{old}} - u_j^{\text{old}} \right) \\
\ u_j^{\text{new}} &= u_j^{\text{old}} + 1.5 \left( u_{j-1}^{\text{old}} - u_j^{\text{old}} \right)
\end{align*}
\]

Note: 1.5 → 1 gives Gauss-Seidel.
successive over-relaxation (SOR)  Gauss-Seidel:

\[
\begin{align*}
    x^{(k+1)} &= D^{-1}(b - Lx^{(k+1)} - Ux^{(k)}) \\
    x^{(k+1)} &= x^{(k)} + D^{-1} \left( b - Lx^{(k+1)} - (D + U)x^{(k)} \right)
\end{align*}
\]

Introduce relaxation parameter \( \omega \) to hasten convergence. \( \omega > 1 \iff \text{over-relaxation} \)

\[
\begin{align*}
    x^{(k+1)} &= x^{(k)} + \omega D^{-1}(b - Lx^{(k+1)} - (D + U)x^{(k)}) \quad \text{computational form} \\
    (I + \omega D^{-1}L)x^{(k+1)} &= (I + \omega D^{-1}L)x^{(k)} + \omega D^{-1}(b - Ax^{(k)}) \\
    x^{(k+1)} &= x^{(k)} + (L + \frac{1}{\omega}D)^{-1}(b - Ax^{(k)}) \quad \text{theoretical form}
\end{align*}
\]

Other methods

- symmetric SOR—SSOR: reverse the ordering every iteration
- block Jacobi, block SOR

Review questions

1. What is another name for the method of simultaneous displacements? successive displacements?

2. What is the Jacobi method for \((L + D + U)x = b\) where \(L\) is strictly lower triangular, \(D\) is diagonal, and \(U\) is strictly upper triangular? Gauss-Seidel? SOR? Write each method as a matrix iteration which mirrors its implementation.

3. Apply Jacobi, Gauss-Seidel, and SOR to specific examples.

4. What does SOR abbreviate? What do we call the parameter that it uses?
5. Given the computational form of a stationary iterative method, rewrite it in the form of iterative refinement in which some matrix $M$ is approximating the original matrix $A$.

6. What is SSOR?

7. What is the form of a stationary iterative method?

**Exercises**

1. The following algorithm is supposed to do one iteration of SOR for $Ax = b$. Diagonal elements of $A$ are stored in $d$ and the $i$th component of $b - Ax$ is computed by the function $\text{residual}(i, x)$.

   ```
   for $i = 1, 2, \ldots, n$ do
     $\Delta x_i = \text{residual}(i, x)/d_i$;
     $x = x + \omega \ast \Delta x_i$;
   
   This is not SOR. Rewrite the algorithm correctly (assuming $A$ and $b$ are inaccessible except through $\text{residual}$).
   
   2. Consider the system of equations

   \[
   \begin{align*}
   5x + 2y + 2z &= 10, \\
   x + 2y + z &= 10, \\
   x + 5y + 10z &= 10.
   \end{align*}
   \]

   Suppose we are solving this with SOR with relaxation parameter $\omega = 1.5$ and have already computed the iterates

   \[
   \begin{array}{c|ccc}
   k & x_k & y_k & z_k \\
   \hline
   0 & 1 & 5 & -2 \\
   1 & 0.7 & & \\
   \end{array}
   \]

   Determine $y_1$.

**6.2 Convergence**

**6.2.1 Conditions for convergence**

For

\[
\begin{align*}
\text{Jacobi} & \quad x^{(k+1)} = (I - D^{-1}A)x^{(k)} + D^{-1}b, \\
\text{SOR} & \quad x^{(k+1)} = (I - (L + \frac{1}{\omega}D)^{-1}A)x^{(k)} + (L + \frac{1}{\omega}D)^{-1}b.
\end{align*}
\]

148
Both are stationary iterative methods of the form 

\[ x^{(k+1)} = B x^{(k)} + c \]

(stationary because \( B \) and \( c \) do not change) where

\[
B_J = -D^{-1}(L + U),
\]
\[
B_\omega = (D + \omega L)^{-1}((1 - \omega)D - \omega U).
\]

In general

\[ x^{(k)} \to x \implies x = B x + c. \]

This latter condition is satisfied for \( B_J \) and \( B_\omega \). (Why?) Hence successive errors satisfy

\[ x^{(k+1)} - x = B (x^{(k)} - x), \]

so

\[ x^{(k)} - x = B^k (x^{(0)} - x). \]

What is \( B^k \)? If \( B \) is diagonalizable

\[ B = X \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n) X^{-1} \]

and

\[ B^k = X \text{diag}(\lambda_1^k, \lambda_2^k, \ldots, \lambda_n^k) X^{-1}. \]

\[
\lim_{k \to \infty} B^k = 0 \iff \lim_{k \to \infty} X^{-1} B^k X = 0
\]
\[
\iff \lim_{k \to \infty} \lambda_i^k = 0 \quad \text{for } i = 1, 2, \ldots, n
\]
\[
\iff |\lambda_i| < 1 \quad \text{for } i = 1, 2, \ldots, n
\]
\[
\iff \rho(B) < 1.
\]

THEOREM A stationary iterative method \( x^{(k+1)} = B x^{(k)} + c \) converges for an arbitrary initial approximation \( x^{(0)} \) if and only if \( \rho(B) < 1 \).

Proof. Recall \( x^{(k)} - x = B^k (x^{(0)} - x) \).

\( \iff \rho(B) < 1 \Rightarrow B^k \to 0 \text{ as } k \to \infty \Rightarrow x^{(k)} \to x \text{ as } k \to \infty \)

\( \Rightarrow \) Assume \( \rho(B) \geq 1 \). There exists \( v \neq 0 \) and \( |\lambda| \geq 1 \) such that \( B v = \lambda v \). Let \( x^{(0)} = x + v \). Then \( x^{(k)} - x = B^k v = \lambda^k v \not\to 0 \text{ as } k \to \infty. \square \)

Jacobi and Gauss-Seidel converge if \( A \) is strictly diagonally dominant. If \( n = 3 \)

\[
B_J = \\
\begin{bmatrix}
0 & -\frac{a_{12}}{a_{11}} & -\frac{a_{13}}{a_{11}} \\
\frac{a_{21}}{a_{11}} & 0 & -\frac{a_{13}}{a_{23}} \\
-\frac{a_{31}}{a_{33}} & -\frac{a_{32}}{a_{33}} & 0
\end{bmatrix}
\]

149
\[ \|B_J\|_\infty = \max \left\{ \frac{|a_{12}| + |a_{13}|}{|a_{11}|}, \frac{|a_{21}| + |a_{23}|}{|a_{22}|}, \frac{|a_{31}| + |a_{32}|}{|a_{33}|} \right\} < 1 \]

\[ \Rightarrow \rho(B_J) < 1 \]

For SOR, \( \rho(B_\omega) \geq |\omega - 1| \) regardless of \( A \). Convergence implies \( 0 < \omega < 2 \). Thus only these values of \( \omega \) are of interest. Conversely, if \( A \) is symmetric positive definite, \( 0 < \omega < 2 \) implies convergence.

### 6.2.2 Rate of convergence

Convergence is not enough; we need a reasonable rate of convergence. We want to know the number of iterations \( k \) required to reduce error by a factor \( \varepsilon \):

\[ \varepsilon = \max_{x^{(0)} \neq x} \frac{\|B^k(x^{(0)} - x)\|}{\|x^{(0)} - x\|} = \|B^k\|. \]

It can be shown \( \lim_{k \to \infty} \|B^k\|^{1/k} = \rho(B) \) where \( \|B^k\|^{1/k} \) represents the average error reduction per iteration. Therefore

\[ \varepsilon^{1/k} \sim \rho(B) \text{ as } k \to \infty, \]

\[ k \sim \frac{\log \varepsilon}{\log \rho(B)} \text{ as } k \to \infty, \]

or

\[ k(\varepsilon) \sim \frac{\log \varepsilon}{\log \rho(B)} \text{ as } \varepsilon \to 0. \]

E.g., for \( \varepsilon = 10^{-d} \) and \( \rho(B) = 0.99 \) the number of iterations = 229d.

*Example* Five-point difference operator on an \( N + 1 \) by \( N + 1 \) grid:

\[ \rho(B_J) = \cos \frac{\pi}{N} = 1 - \frac{\pi^2}{2N^2} + O\left(\frac{1}{N^4}\right) \]

\[ \log \rho(B_J) = -\frac{\pi^2}{2N^2} + O\left(\frac{1}{N^4}\right) \]

\[ k(\varepsilon) \approx \frac{1}{\pi^2} \log \frac{1}{\varepsilon} \]

\[ \rho(B_{GS}) = \cos^2 \frac{\pi}{N} = 1 - \frac{\pi^2}{2N^2} + O\left(\frac{1}{N^4}\right) \]

\[ \log \rho(B_{GS}) = -\frac{\pi^2}{2N^2} + O\left(\frac{1}{N^4}\right) \]

\[ k(\varepsilon) \approx \frac{1}{\pi^2} \log \frac{1}{\varepsilon} \]
Let \( n = (N - 1)^2 \). The cost per iteration is \( 2n \) “multiplications.” The cost for Gauss-Seidel is \( 2n \cdot \frac{n}{\pi^2} \log \frac{1}{\varepsilon} \). If we want to reduce the iteration error \( x^{(k)} - x \) to the level of discretization error \( O\left(\frac{1}{N^2}\right) \), then \( \varepsilon = O\left(\frac{1}{N^2}\right) \) and cost = \( O(n^2 \log n) \).

For SOR we want to choose \( \omega \) to minimize \( \rho(B, \omega) \).

“SOR theory”

For the example one should choose

\[
\bar{\omega} = \frac{2}{1 + \sin \frac{\pi}{N}},
\]

for which

\[
\rho(B, \omega) = \frac{1 - \sin \frac{\pi}{N}}{1 + \sin \frac{\pi}{N}} = \frac{1 - \frac{\pi}{N} + O\left(\frac{1}{N^2}\right)}{1 + \frac{\pi}{N} + O\left(\frac{1}{N^2}\right)} = 1 - \frac{2\pi}{N} + O\left(\frac{1}{N^2}\right).
\]

The number of iterations to reduce error by \( \varepsilon \) is

\[
k(\varepsilon) \sim \frac{\log \varepsilon}{\log \rho(B, \bar{\omega})} \sim \frac{N}{2\pi} \log \frac{1}{\varepsilon},
\]

and the computation is

\[
\begin{align*}
u_i^{\text{new}} &= u_i^{\text{old}} + \frac{\omega}{4} \begin{pmatrix}
-u_{i-1} & -u_{i-N+1} \\
-4u_i & -u_{i+1}
\end{pmatrix}^{\text{new}} - \begin{pmatrix}
-u_{i+n-1} & -u_{i+1} \\
+4u_i & -u_{i+1}
\end{pmatrix}^{\text{old}} \\
&= (1 - \omega)u_i^{\text{old}} + \frac{\omega}{4} \begin{pmatrix}
u_{i-1} & u_{i-N+1} \\
+u_{i-1} & +u_{i+1}
\end{pmatrix}^{\text{new}} + \begin{pmatrix}
u_{i+n-1} & +u_{i+1} \\
+u_{i-1} & +u_{i+1}
\end{pmatrix}^{\text{old}},
\end{align*}
\]

which is 3 “multiplications.” The cost of reducing the error by \( \varepsilon \) is

\[
\frac{3}{2\pi} n^{3/2} \log \frac{1}{\varepsilon} \text{ “multiplications.”}
\]

**Review questions**

1. What is a necessary and sufficient condition for convergence of a stationary iterative method \( x^{(k+1)} = Bx^{(k)} + c \) for an arbitrary initial approximation?

2. What condition on the relaxation parameter \( \omega \) of SOR is necessary for convergence? What condition is sufficient for convergence in the case of a symmetric positive definite matrix?

3. Asymptotically how many iterations are required for a stationary iterative method \( x^{(k+1)} = Bx^{(k)} + c \) to reduce the error by a factor \( \varepsilon \)?
Exercises

1. For the problem
\[
\begin{bmatrix}
2 & 2 \\
1 & 2
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix}
= 
\begin{bmatrix}
-3 \\
5
\end{bmatrix}
\]
determine the Jacobi iteration matrix and its spectral radius. Use the latter to estimate how many Jacobi iterations would be needed to reduce by a factor of 1000 the error in some approximate solution.

2. Estimate how many Jacobi iterations would be needed to reduce the error by a factor of \(10^{-6}\) for \(Ax = b\) where
\[
A = \begin{bmatrix}
4 & 1 & 1 & 1 \\
1 & 4 & \ddots & \\
& \ddots & \ddots & 1 \\
1 & 1 & 1 & 4
\end{bmatrix}
\]

3. For SOR applied to \(\sum_{j=1}^{n} a_{ij} x_j = b_i\) write down the equation which defines \(x_i^{(k+1)}\) in terms of previously computed values.

4. Show that for any 2 by 2 system, the Jacobi method converges if and only if Gauss-Seidel converges.

5. Prove that SOR with \(0 < \omega \leq 1\) converges for strictly diagonally dominant matrices.

6. Show that for any \(\omega\) between 1 and 2 there exists a strictly diagonally dominant matrix (depending on \(\omega\)) such that SOR does not converge for that matrix.

7. (a) Let \(A\) be the \((N-1)^2 \times (N-1)^2\) matrix defined as follows: For any vector \(u = [u_{1,1}; u_{1,2}, u_{2,1}; \ldots; u_{1,N-1}, u_{N-2}, \ldots; u_{N-1,1}; u_{2,N-1}, u_{3,N-1}, \ldots; u_{N-1,2}; \ldots; u_{N-1,N-1}]^T\) define
\[
(Au)_{i,j} = 4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1}
\]
where
\[
u_{0,j} = u_{N,j} = u_{i,0} = u_{i,N} = 0.
\]
The eigenvectors of \(A\) are \(v^{k,l}, 1 \leq k, l \leq N - 1\), where
\[
v_{i,j}^{k,l} = \sin\left(\frac{k\pi i}{N}\right) \sin\left(\frac{l\pi j}{N}\right).
\]
Determine the eigenvalues of \(A\).

(b) Express \(B_J\) as a polynomial in \(A\), and thus determine the eigenvalues and the spectral radius of \(B_J\).

(c) Write \(A\) as a block tridiagonal matrix and separately define each of the blocks.
6.3 Gradient Methods

6.3.1 Methods based on approximate minimization

6.3.2 Steepest descent

6.3.3 Conjugate gradient method

6.3.4 Preconditioned conjugate gradient method

We assume that $A$ is symmetric positive definite:

$$A = Q\Lambda Q^T, \quad \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n), \quad \lambda_i > 0.$$ 

Unlike SOR, there are no parameters to choose.

6.3.1 Methods based on approximate minimization

Minimization methods minimize some objective function $\phi(x) = \phi(x_1, x_2, \ldots, x_n)$:

Possible objective functions:

(i) norm of error $\|x - A^{-1}b\|_2$,

(ii) norm of residual $\|A(x - A^{-1}b)\|_2 = \|Ax - b\|_2$,

(iii) energy norm of error $\|A^{1/2}(x - A^{-1}b)\|_2 = \sqrt{(x - A^{-1}b)^T(A(x - A^{-1}b))}$

where $A^{1/2} = QA^{1/2}Q^T$ is the unique symmetric positive definite square root of $A$. Each of these is a weighted 2-norm of the rotated error:

$$\|A(x - A^{-1}b)\|_2 = \|\Lambda Q^T(x - A^{-1}b)\|_2$$
$$\|A^{1/2}(x - A^{-1}b)\|_2 = \|A^{1/2}Q^T(x - A^{-1}b)\|_2$$
$$\|x - A^{-1}b\|_2 = \|Q^T(x - A^{-1}b)\|_2$$

least desirable weights $\lambda_1, \lambda_2, \ldots, \lambda_n$

weights $\lambda_1^{1/2}, \lambda_2^{1/2}, \ldots, \lambda_n^{1/2}$

most desirable weights 1, 1, \ldots, 1
We denote the energy norm by $|||w||| := \sqrt{w^T A w}$. In some applications it measures the square root of energy or power. We would like to minimize $\|x - A^{-1}b\|_2$ but any minimization method would need to know $A^{-1}b$. However,

$$|||x - A^{-1}b|||^2 = (x - A^{-1}b)^T A (x - A^{-1}b) = x^T A x - 2x^T b + b^T A^{-1}b,$$

which attains its minimum at the same value of $x$ as does

$$\phi(x) = \frac{1}{2} x^T A x - x^T b,$$

and this is cheap to compute for given $x$.

*Note*

unit circle in energy norm

= $\{w \mid |||w||| = 1\}$

= $\{w \mid \|\Lambda^{1/2} Q^T w\|_2 = 1\}$ change variable $\Lambda^{1/2} Q^T w := u$

= $\{Q \Lambda^{-1/2} u \mid \|u\|_2 = 1\}$

= $Q \Lambda^{-1/2} \cdot (\text{unit circle})$

$\text{elongation} = \frac{\lambda^{-1/2}_{\min}}{\lambda^{-1/2}_{\max}} = \sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}} = \sqrt{\|A\|_2 \|A^{-1}\|_2} = \sqrt{\kappa_2(A)}$
The contours of $|||x - A^{-1}b|||$ are curves of equal error in energy norm.

For the 5-point difference operator on unit square

$$\frac{1}{h^2} \begin{pmatrix} -u_{i-1} & -u_{i+N-1} \\ -u_{i-N+1} & 4u_i & -u_{i+1} \end{pmatrix}, \quad h = \frac{1}{N}.$$  

The eigenvalues are $(2N \sin \frac{p\pi}{2N})^2 + (2N \sin \frac{q\pi}{2N})^2$, $p, q = 1, 2, \ldots, N - 1$.

$$\lambda_{\min} \approx 2\pi^2 \quad \lambda_{\max} \approx 8N^2 \quad \sqrt{\kappa_2(A)} \approx \frac{2}{\pi}N$$

generic minimization method

$x_0 =$ initial guess; \{subscripts index iterates not components\}

for $i = 0, 1, 2, \ldots$ do {
  choose a direction $p_i$; \{search direction\}
  choose a distance $\alpha_i$;
  $x_{i+1} = x_i + \alpha_i p_i$;
}

One could choose $\alpha_i$ to minimize $\phi(x_i + \alpha p_i)$. We have

$$\frac{d}{d\alpha} \phi(x_i + \alpha p_i) = \frac{d}{d\alpha} \left( \frac{\alpha^2}{2} p_i^T A p_i + \alpha (p_i^T Ax_i - p_i^T b) + \frac{1}{2} x_i^T Ax_i - x_i^T b \right)$$

$$= \alpha p_i^T A p_i - p_i^T r_i$$
where \( r_i = b - Ax_i \), the residual. Hence

\[
\alpha_i = \frac{p_i^T r_i}{p_i^T A p_i}.
\]

Note that

\[
r_{i+1} = r_i - \alpha_i A p_i,
\]

which is very cheap because we need to compute \( A p_i \) anyway to get \( \alpha_i \). Also note

\[
x_{i+1} = x_i + \frac{p_i p_i^T}{p_i^T A p_i} \cdot r_i.
\]

The matrix with the underbrace is a rank one approximation to \( A^{-1} \).

**cyclic coordinate descent**  Use search directions \( e_1, e_2, \ldots, e_n, e_1, e_2, \ldots, e_n, \ldots \):

\[
x_1 = x_0 + e_1 \frac{e_1^T r_0}{a_{11}},
\]

\[
x_2 = x_1 + e_2 \frac{e_2^T r_1}{a_{22}},
\]

\[
\vdots
\]

\[
x_{\text{new}} = x_{\text{old}} + e_i \frac{e_i^T r_{\text{old}}}{a_{ii}}
\]

For descent in the \( i \)th direction only the \( i \)th component of residual is computed and only the \( i \)th component is updated. This is part of a Gauss-Seidel sweep—the relaxation of the \( i \)th unknown.

\( n \) steps of cyclic coordinate descent \( \equiv 1 \) sweep of Gauss-Seidel

You can see why GS always converges if \( A \) is symmetric positive definite. Also, you can see the reason for overrelaxation, i.e., for multiplying the correction by \( \omega > 1 \). The graph below
shows that

the objective function is reduced if we use $\omega \alpha_k$, $0 < \omega < 2$.

### 6.3.2 Steepest descent

What is the direction of steepest descent? Consider

$$\phi(x_{\text{old}} + \varepsilon p)$$

where $\varepsilon > 0$ is infinitesimal and $\|p\|_2$ is fixed. We have

$$\phi(x_{\text{old}} + \varepsilon p) = \frac{1}{2} (x_{\text{old}} + \varepsilon p)^T A (x_{\text{old}} + \varepsilon p) - (x_{\text{old}} + \varepsilon p)^T b$$

$$= \frac{1}{2} x_{\text{old}}^T A x_{\text{old}} - x_{\text{old}}^T b - \varepsilon p^T (b - A x_{\text{old}}) + O(\varepsilon^2).$$

We can get greatest decrease in $p^T (b - A x_{\text{old}}) = p^T r$ for fixed $\|p\|_2$

by choosing $p = b - A x_{\text{old}} = r_{\text{old}}$. I.e., the residual points in the direction of steepest descent:

- $x_0 =$ initial guess; $r_0 = b - A x_0$;
- for $i = 0, 1, 2, \ldots$ do {
  - $\alpha_i = \frac{r_i^T r_i}{r_i^T A r_i}$;
  - $x_{i+1} = x_i + \alpha_i r_i$;
  - $r_{i+1} = r_i - \alpha_i A r_i$;
}

157
Convergence can be slow if \( \kappa_2(A) \) is large.

The trouble with these methods is that when we minimize in direction \( p_i \), we lose the minimization in directions \( p_1, p_2, \ldots, p_{i-1} \). This can be avoided if we use conjugate directions

\[ p_i^T A p_j = 0 \text{ if } i \neq j. \]

The conjugate gradient method constructs conjugate directions\(^1\) from the gradients \( r_0, r_1, r_2, \ldots \). We will skip the details and simply state the

### 6.3.3 Conjugate gradient method

\[
x_0 = \text{initial guess;}
\]
\[
r_0 = b - Ax_0;
\]
\[
p_0 = r_0;
\]

\[
\text{for } i = 0, 1, 2, \ldots \text{ do } \{
\]
\[
\alpha_i = \frac{r_i^T r_i}{p_i^T A p_i};
\]
\[
x_{i+1} = x_i + \alpha_i p_i;
\]
\[
r_{i+1} = r_i - \alpha_i A p_i;
\]
\[
p_{i+1} = r_{i+1} + \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i} p_i;
\]
\[
\}
\]

The cost per iteration is

1. matrix \cdot vector \quad A p_i
2. vector \cdot vector \quad r_{i+1}^T r_{i+1}, \quad p_i^T (A p_i)
3. vector + scalar \cdot vector

\(^1\)Often \((x, y)_A = x^T A y\) is called the energy inner product.
for a total cost of 1 matrix–vector product and 5n multiplications. For
\[
\begin{bmatrix}
-1 \\ -1 \\ 4 \\ -1
\end{bmatrix}
\]
the cost of matrix–vector is 2.5n “multiplications.”

We note that
\[
\begin{align*}
\mathbf{r}_i &\in \text{span} \{ \mathbf{r}_0, A\mathbf{r}_0, \ldots, A^i \mathbf{r}_0 \}, \\
\mathbf{p}_i &\in \text{span} \{ \mathbf{r}_0, A\mathbf{r}_0, \ldots, A^i \mathbf{r}_0 \}, \\
\mathbf{x}_i - \mathbf{x}_0 &\in \text{span} \{ \mathbf{r}_0, A\mathbf{r}_0, \ldots, A^{i-1} \mathbf{r}_0 \}.
\end{align*}
\]
We call this a Krylov subspace. In fact \( \mathbf{x}_i \) is that element of \( \mathbf{x}_0 + \text{span} \{ \mathbf{r}_0, A\mathbf{r}_0, \ldots, A^{i-1} \mathbf{r}_0 \} \) which is closest to \( A^{-1} \mathbf{b} \) in energy norm.

**Optimality property**
\[
||| \mathbf{x}_i - A^{-1} \mathbf{b} ||| = \min \{|||| \mathbf{x} - A^{-1} \mathbf{b} ||| : \mathbf{x} \in \mathbf{x}_0 + \text{span} \{ \mathbf{r}_0, A\mathbf{r}_0, \ldots, A^{i-1} \mathbf{r}_0 \} \}.
\]
Moreover, it can be shown that the gradients \( \{ \mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{i-1} \} \) constitute an orthogonal basis for the Krylov subspace.

The conjugate gradient method is an incremental construction of \( \mathbf{x}_i \).

The exact solution \( A^{-1} \mathbf{b} = \mathbf{x}_0 + A^{-1} \mathbf{r}_0 \). If \( A \) has \( m \) distinct eigenvalues, \( A^{-1} \) is a linear combination of \( I, A, \ldots, A^{m-1}, \)
\[
e.g., \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}^{-1} = \frac{3}{2} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} - \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}.
\]
To see this, let \( A \) have distinct eigenvalues \( \lambda_1, \lambda_2, \ldots, \lambda_m \), and define
\[
p(\lambda) = (\lambda - \lambda_1)(\lambda - \lambda_2) \cdots (\lambda - \lambda_m) = c_m \lambda^m + \cdots + c_1 \lambda + c_0.
\]
Writing \( A = QAQ^T \), we have
\[
p(A) = Q(\Lambda - \lambda_1 I)(\Lambda - \lambda_2 I) \cdots (\Lambda - \lambda_m I)Q^T = 0,
\]
so
\[
c_m A^m + \cdots + c_1 A + c_0 = 0,
\]
and
\[
c_m A^{m-1} + \cdots + c_1 I + c_0 A^{-1} = 0.
\]
Hence \( A^{-1} \mathbf{r}_0 \in \text{span} \{ \mathbf{r}_0, A\mathbf{r}_0, \ldots, A^{m-1} \mathbf{r}_0 \} \), \( A^{-1} \mathbf{b} \in \mathbf{x}_0 + \text{span} \{ \mathbf{r}_0, A\mathbf{r}_0, \ldots, A^{m-1} \mathbf{r}_0 \} \), and \( x_m = A^{-1} \mathbf{b} \). This is the finite termination property. (In practice we do not go this far.)

**Convergence rate**
\[
||| \mathbf{x}_k - A^{-1} \mathbf{b} ||| \leq \left( \frac{\sqrt{\kappa_2(A)} - 1}{\sqrt{\kappa_2(A)} + 1} \right)^k ||| \mathbf{x}_0 - A^{-1} \mathbf{b} |||
\]
To reduce energy norm of error by \( \varepsilon \) requires \( \approx \frac{\sqrt{\kappa_2(A)}}{2} \log \frac{1}{\varepsilon} \) iterations, e.g., \( \frac{n^{1/2}}{\pi} \log \frac{1}{\varepsilon} \).
6.3.4 Preconditioned conjugate gradient method

To reduce $\kappa_2(A)$, use $G^{-1}AG^{-T}$ instead where $M = GG^T \approx A$ and $M^{-1}$ itself is cheap to perform. Examples of preconditioners are

- SSOR,
- incomplete Cholesky.

Review questions

1. Define the energy norm associated with a symmetric positive definite matrix $A$.
2. Solving $Ax = b$ where $A$ is symmetric positive definite is equivalent to minimizing what readily computable scalar function of $x$? Relate this to the energy norm of the error.
3. What is the spectral condition number $\kappa_2(A)$ of a symmetric positive definite matrix $A$?
4. Given a search direction $p_i$ and an approximation $x_i$ to the value $x^*$ that minimizes $\phi(x) = \frac{1}{2} x^T Ax - b^T x$ with $A$ symmetric positive definite, what is the best choice for $x_{i+1}$?
5. How are search directions chosen in cyclic coordinate descent?
6. What known method do we get if we apply cyclic coordinate descent to objective function $\phi(x) = \frac{1}{2} x^T Ax - b^T x$ where $A$ is symmetric positive definite?
7. For objective function $\phi(x) = \frac{1}{2} x^T Ax - b^T x$ where $A$ is symmetric positive definite, what is the direction of steepest descent?
8. On what property of the matrix does the rate of convergence of steepest descent depend?
9. What does it mean for the directions of the conjugate gradient method to be conjugate?
10. What is a Krylov subspace?
11. Which of the following methods choose their search directions from a Krylov subspace: cyclic coordinate descent? steepest descent? conjugate gradient? For each that does, what is its Krylov subspace?
12. What is the optimality property of the conjugate gradient method?
13. What is the finite termination property of the conjugate gradient method?
14. On what property of the matrix does the rate of convergence of conjugate gradient depend?
Exercises

1. Let

\[ A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}. \]

Plotted below are the level curves (contours) of the function \( \phi(x) = \frac{1}{2}x^T Ax - b^T x \) where \( x = [x_1 \quad x_2]^T \) is variable. Draw the straight line

\[ a_{11}x_1 + a_{12}x_2 = b_1 \]

and the straight line

\[ a_{21}x_1 + a_{22}x_2 = b_2. \]

Explain in words how you constructed these two lines. Recall that relaxing the \( i \)th variable so as to satisfy the \( i \)th equation is equivalent to minimizing \( \phi(x) \) in the direction of the \( i \)th coordinate axis.

2. Draw level curves for \( ||x - A^{-1}b|| \) where

\[ A = \begin{bmatrix} 4 & -2 \\ -2 & 4 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 3 \\ 0 \end{bmatrix}. \]

Model your solution after page 154 of notes. Label various key quantities and do a reasonably careful drawing.

3. Given \( x_i, r_i, p_i \), one conjugate gradient iteration is given by

\[
\begin{align*}
\alpha_i &= r_i^T r_i / (p_i^T A p_i) \\
r_{i+1} &= r_i - \alpha_i A p_i \\
x_{i+1} &= x_i + \alpha_i p_i \\
p_{i+1} &= r_{i+1} + \frac{r_{i+1}^T r_{i+1}}{r_i^T r_i} p_i 
\end{align*}
\]
where $A$ is assumed to be symmetric positive definite. Suppose you are given

```java
static void ax(float[] x, float[] y) {
    int n = x.length;
    // Given x this method returns y where
    // y[i] = a[i][0]*x[0] +...+ a[i][n-1]*x[n-1]
    ...
}
```

You should complete the following method which performs one conjugate gradient iteration, overwriting the old values of $x$, $r_i$, $r_i^T$, and $p_i$ with the new values. Your method must use no temporary arrays except for one float array of dimension $n$ and it must do a minimum amount of floating-point arithmetic.

```java
static void cg(float[] x, float[] r, float rTr, float[] p)
```

4. Show that in the generic minimization method if $\alpha$ is chosen to minimize $\phi(x_i + \alpha p_i)$, then $r_{i+1}$ is orthogonal to $p_i$. Interpret this geometrically.

5. Solve the following system using the conjugate gradient method with $x_1 = 0$.

\[
\begin{bmatrix}
4 & -1 & 0 \\
-1 & 4 & -1 \\
0 & -1 & 4
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix} =
\begin{bmatrix}
2 \\
6 \\
2
\end{bmatrix}
\]

### 6.4 GMRES

GMRES, “generalized minimal residuals,” is a popular iterative method of recent origin for nonsymmetric matrices $A$. It is based on the principle of minimizing the norm of the residual $\|b - Ax\|_2$, since the energy norm is available only for an s.p.d. matrix. It is, however, not a gradient method; rather it chooses for the correction $x_k - x_0$ that element of the Krylov subspace $\text{span}\{r_0, Ar_0, \ldots, A^{k-1}r_0\}$ which minimizes the 2-norm of the residual.

For numerical stability we construct orthonormal bases $\{q_1, q_2, \ldots, q_k\}$ for the Krylov subspaces incrementally. However, rather than applying the Gram-Schmidt process to the sequence $r_0, Ar_0, \ldots, A^{k-1}r_0$, we use what is known as the Arnoldi process. It is based on the fact that each Krylov subspace can be obtained from the orthonormal basis of the Krylov subspace of one dimension less using the spanning set $q_1, q_2, \ldots, q_k, Aq_k$. In other words a new direction in the expanded Krylov subspace can be created by multiplying the most recent basis vector $q_k$ by $A$ rather than by multiplying $A^{k-1}r_0$ by $A$. We remove from this new direction $Aq_k$ its orthogonal projection onto $q_1, q_2, \ldots, q_k$ obtaining the direction

$$v_{k+1} = Aq_k - q_1h_{1k} - q_2h_{2k} - \cdots - q_1h_{kk}$$
where the coefficients $h_{i,k}$ are determined by the orthogonality conditions. Normalization produces

$$q_{k+1} = v_{k+1}/h_{k+1,k}.$$  

The coefficients $h_{ij}$ have been labeled so that we can write

$$AQ_k = Q_{k+1}H_{(k+1)\times k}$$  \hspace{1cm} (6.1)

where $Q_k := [q_1, q_2, \ldots, q_k]$ and $H_{(k+1)\times k}$ is a $k + 1$ by $k$ upper Hessenberg matrix whose $(i,j)$th element for $j \geq i - 1$ is $h_{ij}$.  

Given the basis $q_1, q_2, \ldots, q_k$, we can express any element of the $k$th dimensional Krylov subspace as $Q_k y$ for some $k$-vector $y$. Thus, the minimization problem becomes

$$\min_y \|b - AQ_k y\|_2.$$  

This is a linear least squares problem involving $k + 1$ unknowns and $n$ equations. The number of equations can be reduced to $k + 1$ by using eq. (6.1) to get

$$\|b - AQ_k y\|_2 = \|b - Q_{k+1}H_{(k+1)\times k} y\|_2$$

and then multiplying by an orthogonal matrix $[Q_{k+1}, Q'_{k+1}]^T$ to get

$$\|b - Q_{k+1}H_{(k+1)\times k} y\|_2^2 = \|Q_{k+1}^T b - H_{(k+1)\times k} y\|_2^2 + \text{constant}.$$

### 6.5 Multigrid Methods

If $A$ arises from PDEs, the multigrid method is applicable.

<table>
<thead>
<tr>
<th>continuum problem</th>
<th>(linear operator) $u = \text{inhomog term}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISCRETIZATION</td>
<td>$A_h u_h = b_h$</td>
</tr>
<tr>
<td>COARSENING</td>
<td>$A_{2h} u_{2h} = b_{2h}$</td>
</tr>
</tbody>
</table>

We use relaxation methods, e.g., Gauss-Seidel, to obtain $\tilde{u}_h \approx u_h$ having a smooth remainder (negative error). The remainder $v_h := u_h - \tilde{u}_h$ satisfies

$$A_h v_h = b_h - A_h \tilde{u}_h,$$

and because $v_h$ is smooth, it can be well approximated on a coarser grid:

$$A_{2h} v_{2h} = R_{2h}^h (b_h - A_h \tilde{u}_h)$$

where $R_{2h}^h$ is a restriction operator. This is solved to get a correction $\tilde{v}_{2h}$ either directly or by a recursive call to the multigrid method. Then the approximate solution is improved after interpolating the correction $v_{2h}$ back to the fine grid:

$$\tilde{u}_h = \tilde{u}_h + I_{2h}^h v_{2h}.$$  

This is an optimal method: $O(n)$ operations for $n$ unknowns.