Numerical Methods 3. Approximation of linear algebraic problems

Linear systems of equations

Direct methods

Iterative methods

Eigenvalue problems

Generalized inverse

Linear systems of equations

Let $A \in \mathbf{M}_{N\times N}$ be a *regular* matrix, $b \in \mathbf{R}^N$ is a vector. Solve the following equation:

$$Ax = b$$

It the right-hand side is perturbed and has the form: $b + \Delta b$, this causes an error Δx in the solution: $A(x + \Delta x) = b + \Delta b$. Hence: $\Delta x = A^{-1}\Delta b$. The **absolute error**:

$$\| \Delta x \| \le \| A^{-1} \| \cdot \| \Delta b \|$$

The **relative error**:

$$\frac{\| \Delta x \|}{\| x \|} \le \| A^{-1} \| \cdot \frac{\| \Delta b \|}{\| x \|} = \| A \| \cdot \| A^{-1} \| \cdot \frac{\| \Delta b \|}{\| A \| \cdot \| x \|} \le \| A \| \cdot \| A^{-1} \| \cdot \frac{\| \Delta b \|}{\| Ax \|} \le \| A \| \cdot \| A^{-1} \| \cdot \frac{\| \Delta b \|}{\| b \|}$$

$$\frac{\| \Delta x \|}{\| x \|} \le cond(A) \cdot \frac{\| \Delta b \|}{\| b \|}, \text{ where } cond(A) := \| A \| \cdot \| A^{-1} \| \text{ (condition number)}$$

Ill-conditioned equations

If $A \in \mathbf{M}_{N\times N}$ is a self-adjoint, positive definite matrix, then

(with respect to the matrix norm induced by the Euclidean norm): $cond(A) = \dfrac{\lambda_{\max}}{\lambda_{\min}}$

since in this case $\| A \| = \lambda_{\max}$ and $\| A^{-1} \| = \lambda(A^{-1})_{\max} = \dfrac{1}{\lambda_{\min}}$.

Example for ill-conditioned system of equations:

$$\begin{aligned} 1000x + 999y &= 1 \\ 999x + 998y &= 1 \end{aligned}$$

Solution: $x = 1, \quad y = -1$

$$\begin{aligned} 1000x + 999y &= 1 \\ 999x + 998y &= 0.999 \end{aligned}$$

Solution: $x = 0.001, \quad y = 0$

The condition number of the matrix of the system: 3.9920E+6.

Symmetrization

Let $A \in \mathbf{M}_{N\times N}$ be a given *regular* matrix, and let $b \in \mathbf{R}^N$ be a given vector. Then A^* is also regular, thus, the equation $Ax = b$ is equivalent to the *Gauss' normal equation*

$$A^* Ax = A^* b,$$

the matrix of which is self-adjoint, positive definite. However, its condition number may be *much* greater than that of the original equation.

Example: if A itself is self-adjoint, positive definite, then

$$cond(A^* A) = cond(A^2) = \frac{\lambda_{\max}^2}{\lambda_{\min}^2} = cond(A)^2$$

Problems leading to ill-conditioned equations

Approximation of functions by polynomials: Let $f:[0,1]\to\mathbf{R}$ be a continuous functions. Find the polynomial of degree at most $(N-1)$: $a_0+a_1x+a_2x^2+\ldots+a_{N-1}x^{N-1}$, which is the best approximation of the function f with respect to the $L_2(a,b)$-norm, i.e. for which the error

$$E(a_0,a_1,\ldots,a_{N-1}):=\int_0^1\left(f(x)-\sum_{j=0}^{N-1}a_jx^j\right)^2dx$$

is minimal. Obviously:

$$\frac{\partial E}{\partial a_k}=-2\int_0^1\left(f(x)-\sum_{j=0}^{N-1}a_jx^j\right)x^k\,dx \qquad \Rightarrow \qquad \sum_{j=0}^{N-1}A_{kj}a_j=b_k \quad (k=0,1,\ldots,N-1)$$

where $A_{kj}=\int_0^1 x^{j+k}dx=\frac{1}{j+k+1}$ (Hilbert matrix), and $b_k=\int_0^1 f(x)\cdot x^k dx$.

The Hilbert matrices are extremely ill-conditioned. (However, using trigonometric polynomials, the problem is well-conditioned.)

Some *inverse problems* may lead also to highly ill-conditioned equations as well.

Numerical Methods 3. Approximation of linear algebraic problems

Linear systems of equations

Direct methods

Iterative methods

Eigenvalue problems

Generalized inverse

The Gaussian elimination

The problem to be solved:

$$
\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1N}x_N = b_1 \\
&a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2N}x_N = b_2 \\
&\ldots\ldots\ldots\ldots \\
&a_{N1}x_1 + a_{N2}x_2 + a_{N3}x_3 + \ldots + a_{NN}x_N = b_N
\end{aligned}
$$

Dividing the first equation by a_{11}:

$$
\begin{aligned}
&x_1 + a'_{12}x_2 + a'_{13}x_3 + \ldots + a'_{1N}x_N = b'_1 \\
&a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \ldots + a_{2N}x_N = b_2 \\
&\ldots\ldots\ldots\ldots \\
&a_{N1}x_1 + a_{N2}x_2 + a_{N3}x_3 + \ldots + a_{NN}x_N = b_N
\end{aligned}
$$

The Gaussian elimination

Multiplying the 1st row by a_{k1} and subtracting it from the kth row (k=2,3,...,N)

$$\begin{aligned} x_1 + a'_{12}x_2 + a'_{13}x_3 + \ldots + a'_{1N}x_N &= b'_1 \\ a'_{22}x_2 + a'_{23}x_3 + \ldots + a'_{2N}x_N &= b'_2 \\ \ldots\ldots\ldots\ldots & \\ a'_{N2}x_2 + a'_{N3}x_3 + \ldots + a'_{NN}x_N &= b'_N \end{aligned}$$

The procedure is repeated for the equations 2.,...,N. (elimination).
At the end of the elimination, the system has the form::

$$\begin{aligned} x_1 + c_{12}x_2 + c_{13}x_3 + \ldots + c_{1N}x_N &= d_1 \\ x_2 + c_{23}x_3 + \ldots + c_{2N}x_N &= d_2 \\ x_3 + \ldots + c_{3N}x_N &= d_3 \\ \ldots\ldots\ldots\ldots & \\ x_N &= d_N \end{aligned}$$

The Gaussian elimination

Substitutions:

Nth equation: x_N

$(N-1)$th equation: x_{N-1}

$(N-2)$th equation: x_{N-2}

.....

1st equation: x_1

Total number of the necessary arithmetic operations: $O(N^3)$. (very high!)

Remark: Sometimes the actual row has to be swapped with a later row to avoid a division by a zero or an approximately zero entry.

The Gaussian elimination, example:

$$\begin{aligned} 2x &- 6y &+ 10z &= -12 \\ 2x &- 5y &+ 3z &= -4 \\ 3x &- 2y &+ z &= 3 \end{aligned}$$

$$\left(\begin{array}{ccc|c} 2 & -6 & 10 & -12 \\ 2 & -5 & 3 & -4 \\ 3 & -2 & 1 & 3 \end{array}\right)$$

The Gaussian elimination, example:

$$\begin{aligned} 2x &- 6y + 10z = -12 \\ 2x &- 5y + 3z = -4 \\ 3x &- 2y + z = 3 \end{aligned}$$

$$\left(\begin{array}{ccc|c} 1 & -3 & 5 & -6 \\ 2 & -5 & 3 & -4 \\ 3 & -2 & 1 & 3 \end{array}\right)$$

The Gaussian elimination, example:

$$\begin{aligned} 2x &- 6y &+ 10z &= -12 \\ 2x &- 5y &+ 3z &= -4 \\ 3x &- 2y &+ z &= 3 \end{aligned}$$

$$\left(\begin{array}{ccc|c} 1 & -3 & 5 & -6 \\ 0 & 1 & -7 & 8 \\ 3 & -2 & 1 & 3 \end{array}\right)$$

The Gaussian elimination, example:

$$\begin{aligned} 2x &- 6y &+ 10z &= -12 \\ 2x &- 5y &+ 3z &= -4 \\ 3x &- 2y &+ z &= 3 \end{aligned}$$

$$\left(\begin{array}{ccc|c} 1 & -3 & 5 & -6 \\ 0 & 1 & -7 & 8 \\ 0 & 7 & -14 & 21 \end{array}\right)$$

The Gaussian elimination, example:

$$\begin{aligned} 2x &- 6y &+ 10z &= -12 \\ 2x &- 5y &+ 3z &= -4 \\ 3x &- 2y &+ z &= 3 \end{aligned}$$

$$\left(\begin{array}{ccc|c} 1 & -3 & 5 & -6 \\ 0 & 1 & -7 & 8 \\ 0 & 0 & 35 & -35 \end{array}\right)$$

The Gaussian elimination, example:

$$\begin{aligned} 2x &- 6y &+10z &= -12 \\ 2x &- 5y &+3z &= -4 \\ 3x &- 2y &+z &= 3 \end{aligned}$$

$$\left(\begin{array}{ccc|c} 1 & -3 & 5 & -6 \\ 0 & 1 & -7 & 8 \\ 0 & 0 & 1 & -1 \end{array}\right)$$

The Gaussian elimination, example:

$$\begin{aligned} 2x &- 6y &+10z &= -12 \\ 2x &- 5y &+3z &= -4 \\ 3x &- 2y &+z &= 3 \end{aligned}$$

$$\left(\begin{array}{ccc|c} 1 & -3 & 5 & -6 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -1 \end{array}\right)$$

The Gaussian elimination, example:

$$\begin{aligned} 2x &- 6y &+ 10z &= -12 \\ 2x &- 5y &+ 3z &= -4 \\ 3x &- 2y &+ z &= 3 \end{aligned}$$

$$\left(\begin{array}{ccc|c} 1 & -3 & 0 & -1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -1 \end{array}\right)$$

The Gaussian elimination, example:

$$\begin{aligned} 2x &- 6y &+ 10z &= -12 \\ 2x &- 5y &+ 3z &= -4 \\ 3x &- 2y &+ z &= 3 \end{aligned}$$

$$\left(\begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -1 \end{array}\right)$$

$$\begin{aligned} x &= 2 \\ y &= 1 \\ z &= -1 \end{aligned}$$

Matrix inversion by Gaussian elimination

$$AA^{-1} = I$$

Split A^{-1} and I into column vectors:

$$A^{-1} = \left(\begin{array}{c|c|c|c|c} & & & & \\ a_1 & a_2 & a_3 & \ldots & a_N \\ & & & & \end{array} \right)$$

$$I = \left(\begin{array}{c|c|c|c|c} 1 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & \ldots & 1 \end{array} \right) = \left(\begin{array}{c|c|c|c|c} & & & & \\ e_1 & e_2 & e_3 & \ldots & e_N \\ & & & & \end{array} \right)$$

Now solve the systems of equations: $Aa_k = e_k \qquad (k = 1,2,\ldots,N)$

Matrix inversion by Gaussian elimination, example

$$A := \begin{pmatrix} -3 & 2 & 0 \\ 0 & 3 & 2 \\ -2 & 0 & 1 \end{pmatrix}, \quad A^{-1} = ?$$

$$\left(\begin{array}{ccc|ccc} -3 & -2 & 0 & 1 & 0 & 0 \\ 0 & 3 & 2 & 0 & 1 & 0 \\ -2 & 0 & 1 & 0 & 0 & 1 \end{array}\right)$$

Matrix inversion by Gaussian elimination, example

$$A := \begin{pmatrix} -3 & 2 & 0 \\ 0 & 3 & 2 \\ -2 & 0 & 1 \end{pmatrix}, \quad A^{-1} = ?$$

$$\left(\begin{array}{ccc|ccc} 1 & 2/3 & 0 & -1/3 & 0 & 0 \\ 0 & 3 & 2 & 0 & 1 & 0 \\ -2 & 0 & 1 & 0 & 0 & 1 \end{array}\right)$$

Matrix inversion by Gaussian elimination, example

$$A := \begin{pmatrix} -3 & 2 & 0 \\ 0 & 3 & 2 \\ -2 & 0 & 1 \end{pmatrix}, \quad A^{-1} = ?$$

$$\left(\begin{array}{ccc|ccc} 1 & 2/3 & 0 & -1/3 & 0 & 0 \\ 0 & 3 & 2 & 0 & 1 & 0 \\ 0 & 4/3 & 1 & -2/3 & 0 & 1 \end{array}\right)$$

Matrix inversion by Gaussian elimination, example

$$A := \begin{pmatrix} -3 & 2 & 0 \\ 0 & 3 & 2 \\ -2 & 0 & 1 \end{pmatrix}, \quad A^{-1} = ?$$

$$\left(\begin{array}{ccc|ccc} 1 & 2/3 & 0 & -1/3 & 0 & 0 \\ 0 & 1 & 2/3 & 0 & 1/3 & 0 \\ 0 & 4/3 & 1 & -2/3 & 0 & 1 \end{array}\right)$$

Matrix inversion by Gaussian elimination, example

$$A := \begin{pmatrix} -3 & 2 & 0 \\ 0 & 3 & 2 \\ -2 & 0 & 1 \end{pmatrix}, \quad A^{-1} = ?$$

$$\left(\begin{array}{ccc|ccc} 1 & 2/3 & 0 & -1/3 & 0 & 0 \\ 0 & 1 & 2/3 & 0 & 1/3 & 0 \\ 0 & 0 & 1/9 & -2/3 & -4/9 & 1 \end{array}\right)$$

Matrix inversion by Gaussian elimination, example

$$A := \begin{pmatrix} -3 & 2 & 0 \\ 0 & 3 & 2 \\ -2 & 0 & 1 \end{pmatrix}, \quad A^{-1} = ?$$

$$\left(\begin{array}{ccc|ccc} 1 & 2/3 & 0 & -1/3 & 0 & 0 \\ 0 & 1 & 2/3 & 0 & 1/3 & 0 \\ 0 & 0 & 1 & -6 & -4 & 9 \end{array}\right)$$

Matrix inversion by Gaussian elimination, example

$$A := \begin{pmatrix} -3 & 2 & 0 \\ 0 & 3 & 2 \\ -2 & 0 & 1 \end{pmatrix}, \quad A^{-1} = ?$$

$$\left(\begin{array}{ccc|ccc} 1 & 2/3 & 0 & -1/3 & 0 & 0 \\ 0 & 1 & 0 & 4 & 3 & -6 \\ 0 & 0 & 1 & -6 & -4 & 9 \end{array}\right)$$

Matrix inversion by Gaussian elimination, example

$$A := \begin{pmatrix} -3 & 2 & 0 \\ 0 & 3 & 2 \\ -2 & 0 & 1 \end{pmatrix}, \quad A^{-1} = ?$$

$$\left(\begin{array}{ccc|ccc} 1 & 0 & 0 & -3 & -2 & 4 \\ 0 & 1 & 0 & 4 & 3 & -6 \\ 0 & 0 & 1 & -6 & -4 & 9 \end{array}\right)$$

$$A^{-1} = \begin{bmatrix} -3 & -2 & 4 \\ 4 & 3 & -6 \\ -6 & -4 & 9 \end{bmatrix}$$

The Gauss-Jordan-elimination

The essential difference between Gaussian and Gauss-Jordan elimination is that when eliminating with the *k*th equation, the elimination of the *k*th unknown is performed not only for the latter equations but also for the previous equations at the same time. Thus, there is no need for the substitution steps.

The Gauss-Jordan-elimination, example

$$\begin{aligned} 2x &- 6y &+ 10z &= -12 \\ 2x &- 5y &+ 3z &= -4 \\ 3x &- 2y &+ z &= 3 \end{aligned}$$

$$\left(\begin{array}{ccc|c} 2 & -6 & 10 & -12 \\ 2 & -5 & 3 & -4 \\ 3 & -2 & 1 & 3 \end{array}\right)$$

The Gauss-Jordan-elimination, example

$$\begin{aligned} 2x &- 6y &+ 10z &= -12 \\ 2x &- 5y &+ 3z &= -4 \\ 3x &- 2y &+ z &= 3 \end{aligned}$$

$$\left(\begin{array}{ccc|c} 1 & -3 & 5 & -6 \\ 2 & -5 & 3 & -4 \\ 3 & -2 & 1 & 3 \end{array}\right)$$

The Gauss-Jordan-elimination, example

$$\begin{aligned} 2x &- 6y &+10z &= -12 \\ 2x &- 5y &+3z &= -4 \\ 3x &- 2y &+z &= 3 \end{aligned}$$

$$\left(\begin{array}{ccc|c} 1 & -3 & 5 & -6 \\ 0 & 1 & -7 & 8 \\ 3 & -2 & 1 & 3 \end{array}\right)$$

The Gauss-Jordan-elimination, example

$$\begin{aligned} 2x &- 6y &+ 10z &= -12 \\ 2x &- 5y &+ 3z &= -4 \\ 3x &- 2y &+ z &= 3 \end{aligned}$$

$$\left(\begin{array}{ccc|c} 1 & -3 & 5 & -6 \\ 0 & 1 & -7 & 8 \\ 0 & 7 & -14 & 21 \end{array}\right)$$

The Gauss-Jordan-elimination, example

$$\begin{aligned} 2x &- 6y &+ 10z &= -12 \\ 2x &- 5y &+ 3z &= -4 \\ 3x &- 2y &+ z &= 3 \end{aligned}$$

$$\left(\begin{array}{ccc|c} 1 & -3 & 5 & -6 \\ 0 & 1 & -7 & 8 \\ 0 & 0 & 35 & -35 \end{array}\right)$$

The Gauss-Jordan-elimination, example

$$\begin{aligned} 2x &- 6y &+ 10z &= -12 \\ 2x &- 5y &+ 3z &= -4 \\ 3x &- 2y &+ z &= 3 \end{aligned}$$

$$\left(\begin{array}{ccc|c} 1 & 0 & -16 & 18 \\ 0 & 1 & -7 & 8 \\ 0 & 0 & 35 & -35 \end{array}\right)$$

The Gauss-Jordan-elimination, example

$$\begin{aligned} 2x &- 6y &+ 10z &= -12 \\ 2x &- 5y &+ 3z &= -4 \\ 3x &- 2y &+ z &= 3 \end{aligned}$$

$$\left(\begin{array}{ccc|c} 1 & 0 & -16 & 18 \\ 0 & 1 & -7 & 8 \\ 0 & 0 & 1 & -1 \end{array}\right)$$

The Gauss-Jordan-elimination, example

$$\begin{aligned} 2x &- 6y &+ 10z &= -12 \\ 2x &- 5y &+ 3z &= -4 \\ 3x &- 2y &+ z &= 3 \end{aligned}$$

$$\left(\begin{array}{ccc|c} 1 & 0 & -16 & 18 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -1 \end{array}\right)$$

The Gauss-Jordan-elimination, example

$$\begin{aligned} 2x &- 6y &+ 10z &= -12 \\ 2x &- 5y &+ 3z &= -4 \\ 3x &- 2y &+ z &= 3 \end{aligned}$$

$$\left(\begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & -1 \end{array}\right)$$

$$\begin{aligned} x &= 2 \\ y &= 1 \\ z &= -1 \end{aligned}$$

The *LU* decomposition

If the Gaussian elimination can be performed without swapping rows (no pivot elements are 0), then A can uniquely be decomposed in the form $A = LU$, where L is a lower triangular matrix (with diagonal elements 1), U is an upper triangular matrix.

After performing the LU-decomposition, the system of equations $Ax = b$ is equivalent to the system $LUx = b$, i.e.

$$Ly = b, \quad Ux = y$$

Both equation require low computational cost (number of operations: $O(N^2)$), since:

$$\begin{aligned}
y_1 &= b_1 \\
l_{21}y_1 + y_2 &= b_2 \\
l_{31}y_1 + l_{32}y_2 + y_3 &= b_3 \\
&\dots
\end{aligned}
\quad , \quad
\begin{aligned}
&\dots \\
u_{(N-2),(N-2)}x_{N-2} + u_{(N-2),(N-1)}x_{N-1} + u_{(N-2),N}x_N &= y_{N-2} \\
u_{(N-1),(N-1)}x_{N-1} + u_{(N-1),N}x_N &= y_{N-1} \\
u_{NN}x_N &= y_N
\end{aligned}$$

If there are *a lot of right-hand sides* (but the matrix remains unchanged), then the LU decomposition has to be performed *only once*.

The *LU* decomposition, example

$$\begin{pmatrix} 2 & -6 & 10 \\ 2 & -5 & 3 \\ 3 & -2 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & & \\ & 1 & \\ & & 1 \end{pmatrix}$$

The *LU* decomposition, example

$$\begin{pmatrix} 2 & -6 & 10 \\ 0 & 1 & -7 \\ 3 & -2 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & & \\ 1 & 1 & \\ & & 1 \end{pmatrix}$$

The *LU* decomposition, example

$$\begin{pmatrix} 2 & -6 & 10 \\ 0 & 1 & -7 \\ 0 & 7 & -14 \end{pmatrix} \qquad \begin{pmatrix} 1 & & \\ 1 & 1 & \\ \frac{3}{2} & & 1 \end{pmatrix}$$

The LU decomposition, example

$$\begin{pmatrix} 2 & -6 & 10 \\ 0 & 1 & -7 \\ 0 & 0 & 35 \end{pmatrix} \qquad \begin{pmatrix} 1 & & \\ 1 & 1 & \\ \frac{3}{2} & 7 & 1 \end{pmatrix}$$

U L

The method of orthogonalization

Gram-Schmidt-orthogonalization of vector systems:

Let $a_1, a_2, \ldots, a_n$ be linearly independent vectors in an Euclidean space. Define

$$\tilde{e}_1 := a_1, \quad e_1 := \frac{\tilde{e}_1}{\| \tilde{e}_1 \|}, \text{ and for } 1 < k \leq n:$$

$$\tilde{e}_k := a_k - \sum_{j=1}^{k-1} \langle a_k, e_j \rangle \cdot e_j, \qquad e_k := \frac{\tilde{e}_k}{\| \tilde{e}_k \|}$$

Then the obtained vector system $e_1, e_2, \ldots, e_n$ is *orthonormal*, and for any $1 \leq k \leq n$, the subspaces generated by the first k vectors of e's and a's coincide:

$$[e_1, e_2, \ldots, e_k] = [a_1, a_2, \ldots, a_k]$$

The method of orthogonalization

Denote by $a_1, a_2, \ldots, a_N$ the row vectors of the matrix A. Then the equation $Ax = b$ is equivalent to this system:

$$\langle x, a_k \rangle = b_k \qquad (k = 1,2,\ldots,N)$$

Denote by $e_1, e_2, \ldots, e_N$ the orthonormal basis obtained by Gram-Schmidt orthogonalization from the vectors $a_1, a_2, \ldots, a_N$. Then the numbers $\langle x, e_k \rangle$ can be calculated by the following recursion:

$$\langle x, e_1 \rangle = \frac{\langle x, a_1 \rangle}{\| a_1 \|}$$

$$\langle x, e_k \rangle = \frac{\langle x, \tilde{e}_k \rangle}{\| \tilde{e}_k \|} = \frac{\langle x, a_k \rangle - \sum_{j=1}^{k-1} \langle a_k, e_j \rangle \cdot \langle x, e_j \rangle}{\| \tilde{e}_k \|} = \frac{b_k - \sum_{j=1}^{k-1} \langle a_k, e_j \rangle \cdot \langle x, e_j \rangle}{\| \tilde{e}_k \|} \qquad (k = 2,3,\ldots,N)$$

Thus, the solution can be expressed in the form of finite Fourier series:

$$\boxed{x = \sum_{k=1}^{N} \langle x, e_k \rangle \cdot e_k}$$

Number of operations: $O(N^3)$

Solution of systems of equations by spectral decomposition

Let $A \in \mathbf{M}_{N\times N}$ be *self-adjoint*, regular matrix. Denote by $\lambda_1, \lambda_2, ..., \lambda_N$ the eigenvalues and $s_1, s_2, ..., s_N$ the orthonormal eigenvectors. Express the right-hand vector *b* in terms of finite Fourier series:

$$b = \sum_{j=1}^{N} \langle b, s_j \rangle \cdot s_j$$

Them the solution of the equation $Ax = b$:

$$x = \sum_{j=1}^{N} \frac{\langle b, s_j \rangle}{\lambda_j} \cdot s_j$$

since $Ax = \sum_{j=1}^{N} \frac{\langle b, s_j \rangle}{\lambda_j} \cdot As_j = \sum_{j=1}^{N} \frac{\langle b, s_j \rangle}{\lambda_j} \cdot \lambda_j s_j = \sum_{j=1}^{N} \langle b, s_j \rangle \cdot s_j = b$.

That is, the solution can be expressed in an *explicit form*. Computational cost: $O(N^2)$.
Drawback: all of the eigenvalues (and a system of eigenvectors) should be explicitly known.

Solution of three-diagonal system of equations by recursion

$$\begin{pmatrix} B_1 & C_1 & 0 & 0 & \dots & 0 \\ A_2 & B_2 & C_2 & 0 & \dots & 0 \\ 0 & A_3 & B_3 & C_3 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & A_{N-1} & B_{N-1} & C_{N-1} \\ 0 & \dots & 0 & 0 & A_N & B_N \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \dots \\ b_N \end{pmatrix}$$

Try to find the solution in the form $x_k := m_{k+1}x_{k+1} + n_{k+1}$ ('backward' recursion).

Then $x_{k-1} := m_k x_k + n_k = m_k(m_{k+1}x_{k+1} + n_{n+1}) + n_k = m_k m_{k+1}x_{k+1} + (m_k n_{n+1} + n_k)$
Substituting into the *k*th equation $(k = 2,3,\dots,N-1)$:

$$A_k x_{k-1} + B_k x_k + C_k x_{k+1} = A_k[m_k m_{k+1}x_{k+1} + (m_k n_{n+1} + n_k)] + B_k[m_{k+1}x_{k+1} + n_{k+1}] + C_k x_{k+1} =$$
$$= (A_k m_k m_{k+1} + B_k m_{k+1} + C_k)x_{k+1} + (A_k m_k n_{n+1} + A_k n_k + B_k n_{k+1}) = b_k$$

Solution of three-diagonal system of equations by recursion

The equality is clearly valid if

$$A_k m_k m_{k+1} + B_k m_{k+1} + C_k = 0 \quad \text{és} \quad A_k m_k n_{n+1} + A_k n_k + B_k n_{k+1} = b_k,$$

that is, if the numbers m_k, n_k satisfy the 'forward' recursions:

$$m_{k+1} = -\frac{C_k}{A_k m_k + B_k}, \qquad n_{k+1} = \frac{b_k - A_k n_k}{A_k m_k + B_k}$$

Define $m_1 := 0$, $n_1 := 0$, then $m_2 = -\frac{C_1}{B_1}$, $n_2 = \frac{b_1}{B_1}$, and $x_1 = m_2 x_2 + n_2 = -\frac{C_1}{B_1} x_2 + \frac{b_1}{B_1}$,

whence the 1st equation: $B_1 x_1 + C_1 x_2 = -C_1 x_2 + b_1 + C_1 x_2 = b_1$

In the backward recursion, define $x_N := n_{N+1}$, then $x_{N-1} = m_N x_N + n_N$, thus, the *N*th equation:

$$A_N x_{N-1} + B_N x_N = A_N (m_N x_N + n_N) + B_N x_N = (A_N m_N + B_N) x_N + A_N n_N =$$
$$= (A_N m_N + B_N) n_{N+1} + A_N n_N = b_N - A_N n_N + A_N n_N = b_N$$

Solution of three-diagonal system of equations by recursion

The complete algorithm:
Forward step: 2 recursions:

$$m_1 := 0, \qquad n_1 := 0$$

$$m_{k+1} := -\frac{C_k}{A_k m_k + B_k}, \qquad n_{k+1} := \frac{b_k - A_k n_k}{A_k m_k + B_k}, \qquad (k = 1,\ldots,N)$$

Backward step: 1 recursion:

$$x_N := n_{N+1}$$

$$x_{k-1} := m_k x_k + n_k \qquad (k = N, N-1,\ldots,2)$$

Computational cost: $O(N)$ only!

Numerical Methods 3. Approximation of linear algebraic problems

Linear systems of equations

Direct methods

Iterative methods

Eigenvalue problems

Generalized inverse

Converting to a fixed point iteration

Transform the original equation $Ax = b$ to the following form (there are lots of possibilities):

$$x = Bx + f\,,$$

and, for an arbitrary starting approximation $x_0 \in \mathbf{R}^N$, consider the following iteration:

$$x_{n+1} := Bx_n + f \qquad (n = 0,1,2,\ldots)$$

If $\| B \| < 1$, then the mapping $F(x) := Bx + f$ is a contraction, since

$$\| F(x) - F(y) \| = \| Bx + f - By - f \| \le \| B \| \cdot \| x - y \|$$

Therefore a unique fixed point exists, and the recursively defined sequence $x_{n+1} := Bx_n + f$ converges to this vector.

The smaller the matrix norm $\| B \|$, the faster the convergence.

Converting to a fixed point iteration

Theorem: If the absolute values of all the eigenvalues of B are less than 1, then the iteration is convergent.

However, the condition of convergence is not always sufficient… look at the following example:

$$B := \begin{pmatrix} \alpha & \beta & & & \\ & \alpha & \beta & & \\ & & \dots & \dots & \\ & & & \alpha & \beta \\ & & & & \alpha \end{pmatrix} \in \mathbf{M}_{N\times N}, \qquad f := \mathbf{0}, \qquad x_0 := \begin{pmatrix} 0 \\ 0 \\ \dots \\ 0 \\ 1 \end{pmatrix} \in \mathbf{R}^N$$

All the eigenvalues of B are equal to α. The exact solution is the zero vector.

The $(N-j)$th component of the nth approximation (if $n < N$): $x_n^{(N-j)} = \binom{n}{j} \alpha^{n-j} \beta^j$

The computation may be broken down before the convergence, due to **overflow**.
(For instance, $\alpha := 1/2$, $\beta := 2$, $n := 200$, $j := 100$, $N > 200$).

The simple (Richardson) iteration

Let $A \in \mathbf{M}_{N\times N}$ be a self-adjoint, positive definite matrix. The equation $Ax = b$ is equivalent to the equation

$$x = x - \omega \cdot (Ax - b) = (I - \omega A)x + \omega b$$

where $\omega > 0$ is an iteration parameter. This results in the fixed point iteration:

$$x_{n+1} := (I - \omega A)x_n + \omega b$$

The above iteration is convergent for any sufficiently small parameter $\omega > 0$.

The iteration is the fastest, when $\| I - \omega A \|$ is the least, i.e. when $\omega = \dfrac{2}{\lambda_{\min} + \lambda_{\max}}$.

In this case: $\| I - \omega A \| = \dfrac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}}$

That is, for the proper definition of the optimal iteration parameter, the greatest and the least eigenvalues should be known.

The Jacobi iteration

Let us decompose the matrix of the equation $Ax = b$ into the sum of a **diagonal matrix**, and a **lower** and an **upper triangular matrices**: $A = L + D + U$. Then $Dx = -(L+U)x + b$, i.e. $x = -D^{-1}(L+U)x + D^{-1}b$.

The Jacobi iteration: $x_{n+1} = -D^{-1}(L+U)x_n + D^{-1}b$. Componentwise:

$$x_{n+1}^{(k)} = -\frac{1}{a_{kk}} \sum_{j=1}^{k-1} a_{kj} x_n^{(j)} - \frac{1}{a_{kk}} \sum_{j=k+1}^{N} a_{kj} x_n^{(j)} + \frac{1}{a_{kk}} \cdot b_k \qquad (k = 1,2,\ldots,N)$$

If A is *diagonally dominant*, i.e. $\sum_{j \neq k} |a_{kj}| < |a_{kk}|$, then the Jacobi iteration is convergent.

Indeed, in this case the row norm of the matrix $B := -D^{-1}(L+U)$ is less than 1, since

$$\| B \| = \max_k \sum_{j \neq k} \frac{|a_{kj}|}{|a_{kk}|} = \max_k \frac{1}{|a_{kk}|} \sum_{j \neq k} |a_{kj}| < 1.$$

The Seidel iteration

The crucial difference between the Jacobi and Seidel iteration is as follows:
At the update of the components of the approximate solution, the components which have been just updated, will be *immediately* utilized at the update of the next components.

$$x_{n+1}^{(k)} = -\frac{1}{a_{kk}} \sum_{j=1}^{k-1} a_{kj} x_{n+1}^{(j)} - \frac{1}{a_{kk}} \sum_{j=k+1}^{N} a_{kj} x_n^{(j)} + \frac{1}{a_{kk}} \cdot b_k \qquad (k = 1,2,\ldots,N)$$

If A is diagonally dominant, or self-adjoint and positive definite,
then the Seidel iteration is convergent.

Variational methods

Let A be a **self-adjoint, positive definite matrix**, and consider the equation $Ax = b$. Denote by x^* the exact solution.

Introduce the inner product $\langle x, y \rangle_A := \langle Ax, y \rangle$ (**energetic scalar product**, ***A*-scalar product**).

The norm induced by this inner product is the **energetic norm** or ***A*-norma**: $\| x \|_A^2 = \langle Ax, x \rangle$.

Define the **energetic functional** in the following way: $F(x) := \langle Ax, x \rangle - 2\langle x, b \rangle$.

Obviously: $F(x) = \langle x, x \rangle_A - 2\langle x, x^* x \rangle_A = \left|| x - x^* || \right|_A^2 - ||x^*||_A^2$, therefore:

The energetic functional has a unique minimal value,
and this is reached at the exact solution x^* of the equation $Ax = b$.

Thus, the original problem (the solution of a system of equations) is converted to a minimization problem. The approximate solution techniques based on this minimization problem are called **variational methods**.

Minimization along a direction

Let $e \in \mathbf{R}^N$ be a given (direction) vector, and let $x \in \mathbf{R}^N$ be a given approximate solution. Minimize the univariate function $f(t) := F(x + t \cdot e)$. First, calculate the gradient of F:

$$F(x+h) = \langle Ax + Ah, x + h \rangle - 2\langle x + h, b \rangle =$$

$$= \langle Ax, x \rangle + \langle Ax, h \rangle + \langle Ah, x \rangle + \langle Ah, h \rangle - 2\langle x, b \rangle - 2\langle h, b \rangle = F(x) + 2\langle Ax - b, h \rangle + \langle Ah, h \rangle$$

which implies that $DF(x)h = 2\langle Ax - b, h \rangle$, i.e. $DF(x) = 2(Ax - b)$

Now the minimization of f can be performed in a standard way:

$$f'(t) = \langle Df(x + t \cdot e), e \rangle = 2\langle A(x + t \cdot e) - b, e \rangle = 2\langle Ax - b, e \rangle - 2t\langle Ae, e \rangle = 0,$$

i.e. the derivative vanishes at $t = -\dfrac{\langle Ax - b, e \rangle}{\langle Ae, e \rangle}$. Consequently, the improved approximation is:

$$\tilde{x} = x + t \cdot e = x - \frac{\langle Ax - b, e \rangle}{\langle Ae, e \rangle} \cdot e$$

The gradient method

Main idea: the energetic functional F should always be minimized along the *steepest descent direction* i.e. along the direction of the negative gradient vector.

Let x_n be an arbitrary approximate solution of the equation $Ax = b$, and denote by $r_n := Ax_n - b$ the **residual vector**. Then the improved approximation after minimizing the functional **F** along the direction r_n:

$$x_{n+1} = x_n + t \cdot r_n = x_n - \frac{\langle Ax_n - b, r_n \rangle}{\langle Ar_n, r_n \rangle} \cdot r_n = x_n - \frac{\| r_n \|^2}{\langle Ar_n, r_n \rangle} \cdot r_n$$

Theorem: The error after the nth step can be estimated as:

$$\| x_n - x^* \|_A^2 \leq \left(1 - \frac{\lambda_{\min}}{\lambda_{\max}} \right)^n \| x_0 - x^* \|_A^2$$

However, to apply the method, it is *not* necessary to know the extremal eigenvalues!

Computational cost: $O(N^2)$ in each iteration step.

The conjugate gradient method

Let A be a **self-adjoint, positive definite matrix**, and consider the equation $Ax = b$.
Let $x_0 \in \mathbf{R}^N$ be an arbitrary starting approximation and set $r_0 := Ax_0 - b$, $d_0 := -r_0$. For every $n = 0,1,2,\ldots$, define:

$$r_n := Ax_n - b$$

$$x_{n+1} := x_n - \frac{\langle r_n, d_n \rangle}{\langle Ad_n, d_n \rangle} \cdot d_n$$

$$r_{n+1} := Ax_{n+1} - b$$

$$d_{n+1} := -r_{n+1} + \frac{\langle Ar_{n+1}, d_n \rangle}{\langle Ad_n, d_n \rangle} \cdot d_n$$

Theorem: Without rounding errors, the conjugate gradient method results in the exact solution within at most N iteration steps.

That is, in principle, the conjugate gradient method can be classified as a direct method.

Numerical Methods 3. Approximation of linear algebraic problems

Linear systems of equations

Direct methods

Iterative methods

Eigenvalue problems

Generalized inverse

Eigenvalue problems

An eigenvalue problem is always equivalent to the solution of an equation of higher degree (characteristic equation):

$$As = \lambda s \qquad (s \neq \mathbf{0}) \qquad \Leftrightarrow \qquad \det(A - \lambda I) = 0$$

Gershgorin's theorem: For an arbitrary matrix $A \in \mathbf{M}_{N\times N}$, the eigenvalues of A are located in the union of the closed circles of the complex plane centered at a_{kk}, with radius $r_k := \sum_{j\neq k} |a_{kj}|$.

Indeed, let λ be an eigenvalue with eigenvector $s \neq \mathbf{0}$. Denote by k the index, for which $|s_k|$ is maximal, i.e. $|s_k| = \|s\|_{\max}$. For this index k:

$$(As)_k = \sum_{j=1}^{N} a_{kj}s_j = a_{kk}s_k + \sum_{j\neq k} a_{kj}s_j = \lambda s_k \qquad \Rightarrow \qquad a_{kk} - \lambda = -\sum_{j\neq k} a_{kj}\frac{s_j}{s_k} \qquad \Rightarrow$$

$$|a_{kk} - \lambda| \leq \sum_{j\neq k} |a_{kj}| \cdot \frac{|s_j|}{|s_k|} \leq \sum_{j\neq k} |a_{kj}| = r_k$$

Eigenvalue problems

Let $A \in \mathbf{M}_{N\times N}$ be self-adjoint with eigenvalues $0 \le |\lambda_1| \le |\lambda_2| \le \cdots \le |\lambda_{N-1}| < |\lambda_N|$ and with orthonormal eigenvectors $s_1, s_2, \ldots, s_N$.

The power method: Let x_0 be a starting vector which is *not* orthogonal to s_N.
For $n = 0,1,2, \ldots$, define $x_{n+1} := Ax_n$.

Then the sequence of the quotients $\dfrac{\langle Ax_n, x_n \rangle}{\| x_n \|^2}$ (*Rayleigh quotients*) converges to λ_N.

Indeed, let $x_0 := \sum_{j=1}^{N} \alpha_j s_j$ $(\alpha_N \neq 0)$, then $x_n = A^n x_0 = \sum_{j=1}^{N} \alpha_j \lambda_j^n s_j$. Hence

$$\langle Ax_n, x_n \rangle = \left\langle \sum_{j=1}^{N} \alpha_j \lambda_j^{n+1} s_j, \sum_{k=1}^{N} \alpha_k \lambda_k^n s_k \right\rangle = \sum_{j=1}^{N} |\alpha_j|^2 \cdot \lambda_j \cdot |\lambda_j|^{2n},$$ and, at the same time:

$$\| x_n \|^2 = \left\langle \sum_{j=1}^{N} \alpha_j \lambda_j^n s_j, \sum_{k=1}^{N} \alpha_k \lambda_k^n s_k \right\rangle = \sum_{j=1}^{N} |\alpha_j|^2 \cdot |\lambda_j|^{2n},$$ which implies the theorem.

Eigenvalue problems

Let $A \in \mathbf{M}_{N\times N}$ be self-adjoint with eigenvalues $0 \le |\lambda_1| \le |\lambda_2| \le \cdots \le |\lambda_{N-1}| < |\lambda_N|$ and with orthonormal eigenvectors $s_1, s_2, ..., s_N$. Applying the power method to the inverse matrix A^{-1}:

The inverse iteration: Let x_0 be a starting vector which is *not* orthogonal to s_1.

For $n = 0,1,2, ...$, define $x_{n+1} := A^{-1}x_n$.

Then the sequence of quotients $\dfrac{\| x_n \|^2}{\left\langle A^{-1}x_n, x_n \right\rangle}$ converges to λ_1.

At each step of iteration, one has to solve a system of equations $Ax_{n+1} = x_n$.
(From computational point of view, the use of the *LU* decomposition may be advantageous).

Eigenvalue problems

Jacobi's method: Let $A \in \mathbf{M}_{N\times N}$ self-adjoint. Define the pair of indices (p,q) $(p<q)$, for which $|a_{pq}|$ is maximal outside the main diagonal. Define $\operatorname{ctg} 2t := \dfrac{a_{qq}-a_{pp}}{2a_{pq}}$, and

$$Q := \begin{pmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & \cos t & \dots & \dots & \dots & \sin t & \\ & & \dots & 1 & & & \dots & \\ & & \dots & & \dots & & \dots & \\ & & \dots & & & 1 & \dots & \\ & & -\sin t & \dots & \dots & \dots & \cos t & \\ & & & & & & & 1 \\ & & & & & & & & 1 \end{pmatrix}$$

Now update the matrix A:

$$A := Q^{*}AQ,$$

and repeat the procedure. If the eigenvalues of A are distinct, then the matrix sequence defined above tends to a diagonal matrix, the main diagonal of which contains the eigenvalues of A.

Numerical Methods 3. Approximation of linear algebraic problems

Linear systems of equations

Direct methods

Iterative methods

Eigenvalue problems

Generalized inverse

The Singular Value Decomposition (SVD)

The Singular Value Decomposition (SVD): Every matrix $A \in \mathbf{M}_{m\times n}$ can be (non-uniquely) decomposed in the form

$$A = USV^*, \quad \text{where:}$$

$U \in \mathbf{M}_{m\times m}$, $V \in \mathbf{M}_{n\times n}$ are orthogonal matrices;
$S \in \mathbf{M}_{m\times n}$ is a diagonal matrix;
the non-zero diagonal elements of S (the *singular values* of A) are the positive square roots of the matrix A^*A.

Generalized inverses of matrices

Let the singular value decomposition of the matrix $A \in \mathbf{M}_{m\times n}$ be: $A = USV^*$. Then the matrix

$$\boxed{A^+ = VS^+U^* \in \mathbf{M}_{n\times m}}$$

is said to be the *generalized inverse* (Moore-Penrose pseudoinverse) of the matrix *A*, where if

$$S = \begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \sigma_3 & \\ & & & \ddots \end{pmatrix}, \quad \text{then} \quad S^+ := \begin{pmatrix} 1/\sigma_1 & & & \\ & 1/\sigma_2 & & \\ & & 1/\sigma_3 & \\ & & & \ddots \end{pmatrix}$$

The generalized inverse is uniquely determined, and, if $A \in \mathbf{M}_{n\times n}$ is regular, then $A^+ = A^{-1}$.

Generalized solutions of systems of equations

The generalized solution of the equation $Ax = b$ is: $x^+ := A^+ b$ (always exists and uniquely determined).

Theorem: The generalized solution of the equation $Ax = b$ minimizes the functional

$$F(x) := \| Ax - b \|^2$$

Moreover, if there are several minimizing vectors, then the generalized solution has the least Euclidean norm (the solution *in the sense of least squares*).

The vectors w which minimize the functional $F(x) := \| Ax - b \|^2$, satisfy the **Gaussian normal equations**:

$$A^* Aw = A^* b$$

Thus, the generalized solution can be approximated by some variational (iterative) method.