## Triangular Decomposition

Let $L$ denote a lower-triangular matrix (where the elements $l_{ij}$ satisfy $l_{ij} = 0$ for $i < j$) and $R$ an upper-triangular matrix ($r_{ij} = 0$ for $i > j$); the diagonal elements of $L$ satisfy $l_{11} = ... = l_{nn} = 1$. Matrices $A$, $L$, $R$ are supposed to be of size $n \times n$ and vectors $x$, $b, ...$ have $n$ components. Frequently, numerical methods must solve one or more systems of linear equations

$$Ax = b\ .$$

A well-known direct method to solve this system is Gaussian elimination. First, in a "forward"-phase, an equivalent system

$$Rx = \widehat{b}$$

is calculated. Then, in a "backward"-phase starting with the last component $x_n$, all components of $x$ are calculated one by one in the order $x_n, x_{n-1}, \ldots, x_1$. Gaussian elimination requires $\frac{2}{3}n^3 + O(n^2)$ arithmetic operations for full matrices $A$. With this count of $O(n^3)$, Gaussian elimination must be considered as an expensive endeavor, and is prohibitive for large values of $n$. (For alternatives, see iterative methods below in Appendix C2.) The forward phase of Gaussian elimination is equivalent to an *LR-decomposition*. This means the factorization into the product of two triangular matrices $L, R$ in the form

$$PA = LR\ .$$

Here $P$ is a permutation matrix arranging for the exchange of rows that corresponds to the pivoting of the Gaussian algorithm. The $LR$-decomposition exists for all nonsingular $A$. After the $LR$-decomposition is calculated, only two equations with triangular matrices need to be solved,

$$Ly = Pb \quad \text{and} \quad Rx = y\ .$$

## Tridiagonal Matrices

For tridiagonal matrices the $LR$-decomposition specializes to an algorithm that requires only $O(n)$ operations, which is inexpensive. Since several of the matrices in this book are tridiagonal, we include the algorithm. Let the tridiagonal system $Ax = b$ be in the form

$$\begin{pmatrix} \alpha_1 & \beta_1 & & & 0 \\ \gamma_2 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{n-1} & \alpha_{n-1} & \beta_{n-1} \\ 0 & & & \gamma_n & \alpha_n \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix} \tag{C1.5}$$

Starting the Gaussian elimination with the first row to produce zeros in the subdiagonal during a forward loop, the algorithm is as follows:

```
$\hat\alpha_1 := \alpha_1,\ \hat b_1 := b_1$
(forward loop)  for $i = 2, \dots, n$ :
    $\hat\alpha_i = \alpha_i - \beta_{i-1}\dfrac{\gamma_i}{\hat\alpha_{i-1}}, \quad \hat b_i = b_i - \hat b_{i-1}\dfrac{\gamma_i}{\hat\alpha_{i-1}}$
$x_n := \dfrac{\hat b_n}{\hat\alpha_n}$                                                    (C1.6)
(backward loop)  for $i = n-1, \dots, 1$ :
    $x_i = \dfrac{1}{\hat\alpha_i}(\hat b_i - \beta_i x_{i+1})$
```

Here the "new" elements of the equivalent triangular system are indicated with a "hat;" the necessary checks for nonsingularity ($\hat\alpha_{i-1} \neq 0$) are omitted. The algorithm (C1.6) needs about $8n$ operations. If one would start Gaussian elimination from the last row and produces zeros in the superdiagonal, an $RL$-decomposition results. The reader may wish to formulate the related backward/forward algorithm as an exercise.

### Cholesky Decomposition

For *positive-definite* matrices $A$ (means symmetric or Hermitian and $x^H Ax > 0$ for all $x \neq 0$) there is exactly one lower-triangular matrix $L$ with positive diagonal elements such that

$$A = LL^H \ .$$

Here the diagonal elements of $L$ need not be normalized. For real matrices $A$ also $L$ is real, hence $A = LL^{tr}$. (Hint: The Hermitian matrix $A^H$ of $A$ is defined as $\bar A^{tr}$, where $\bar A$ means elementwise complex conjugate.) For a computer program of Cholesky decomposition see [PTVF92].

## C2 Iterative Methods for $Ax = b$

The system of linear equations $Ax = b$ in $\mathbb{R}^n$ can be written

$$Mx = (M - A)x + b \ ,$$

where $M$ is a suitable matrix. For nonsingular $M$ the system $Ax = b$ is equivalent to the fixed-point equation

$$x = (I - M^{-1}A)x + M^{-1}b \ ,$$

which leads to the iteration

$$x^{(k+1)} = (\underbrace{I - M^{-1}A}_{=:B})x^{(k)} + M^{-1}b \ . \tag{C2.1}$$

The computation of $x^{(k+1)}$ is done by solving the system of equations $Mx^{(k+1)} = (M - A)x^{(k)} + b$. Subtracting the fixed-point equation and applying Lemma 4.2 shows

$$\text{convergence} \quad \Longleftrightarrow \quad \rho(B) < 1 \; ;$$

$\rho(B)$ is the spectral radius of matrix $B$. For this convergence criterion there is a sufficient criterion that is easy to check. Natural matrix norms satisfy $\|B\| \geq \rho(B)$. Hence $\|B\| < 1$ implies convergence. Application to the matrix norms

$$\|B\|_\infty = \max_i \sum_{j=1}^{n} |b_{ij}| \, ,$$

$$\|B\|_1 = \max_j \sum_{i=1}^{n} |b_{ij}| \, ,$$

produces sufficient convergence criteria: The iteration converges if

$$\sum_{j=1}^{n} |b_{ij}| < 1 \quad \text{for } 1 \leq i \leq n$$

or if

$$\sum_{i=1}^{n} |b_{ij}| < 1 \quad \text{for } 1 \leq j \leq n \; .$$

By obvious reasons these criteria are called row sum criterion and column sum criterion. The *preconditioner* matrix $M$ is constructed such that rapid convergence of (C2.1) is achieved. Further, the structure of $M$ must be simple so that the linear system is easily solved for $x^{(k+1)}$.

Simple examples are obtained by additive splitting of $A$ into the form $A = D - L - U$, with

$D$ diagonal matrix
$L$ strict lower-triangular matrix
$U$ strict upper-triangular matrix

### Jacobi's Method

Choosing $M := D$ implies $M - A = L + U$ and establishes the iteration

$$Dx^{(k+1)} = (L + U)x^{(k)} + b \; .$$

By the above convergence criteria a strict diagonal dominance of $A$ is sufficient for the convergence of Jacobi's method.

### Gauß–Seidel Method

Here the choice is $M := D - L$. This leads via $M - A = U$ to the iteration

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

## SOR (Successive Overrelaxation)

The SOR method can be seen as a modification of the Gauß-Seidel method, where a *relaxation parameter* $\omega_{\mathrm{R}}$ is introduced and chosen in a way that speeds up the convergence:

$$M := \frac{1}{\omega_{\mathrm{R}}} D - L \implies M - A = \left(\frac{1}{\omega_{\mathrm{R}}} - 1\right) D + U$$

$$\left(\frac{1}{\omega_{\mathrm{R}}} D - L\right) x^{(k+1)} = \left(\left(\frac{1}{\omega_{\mathrm{R}}} - 1\right) D + U\right) x^{(k)} + b$$

The SOR-method can be written as follows:

$$\begin{cases} B_{\mathrm{R}} := \left(\frac{1}{\omega_{\mathrm{R}}} D - L\right)^{-1} \left(\left(\frac{1}{\omega_{\mathrm{R}}} - 1\right) D + U\right) \\ x^{(k+1)} = B_{\mathrm{R}} x^{(k)} + \left(\frac{1}{\omega_{\mathrm{R}}} D - L\right)^{-1} b \end{cases}$$

The Gauß–Seidel method is obtained as special case for $\omega_{\mathrm{R}} = 1$.

### Choosing $\omega_{\mathrm{R}}$

The difference vectors $d^{(k+1)} := x^{(k+1)} - x^{(k)}$ satisfiy

$$d^{(k+1)} = B_{\mathrm{R}} d^{(k)} \,. \tag{C2.2}$$

This is the power method for eigenvalue problems. Hence the $d^{(k)}$ converge to the eigenvector of the dominant eigenvalue $\rho(B_{\mathrm{R}})$. Consequently, if (C2.2) converges then

$$d^{(k+1)} = B_{\mathrm{R}} d^{(k)} \approx \rho(B_{\mathrm{R}}) d^{(k)} \,.$$

Then $|\rho(B_{\mathrm{R}})| \approx \frac{\|d^{(k+1)}\|}{\|d^{(k)}\|}$ for arbitrary vector norms. There is a class of matrices $A$ with

$$\rho(B_{\mathrm{GS}}) = (\rho(B_{\mathrm{J}}))^2 \,,\ B_{\mathrm{J}} := D^{-1}(L+U)$$

$$\omega_{\mathrm{opt}} = \frac{2}{1 + \sqrt{1 - \rho(B_{\mathrm{J}})^2}} \,,$$

see [Va62], [SB96]. Here $B_{\mathrm{J}}$ denotes the iteration matrix of the Jacobi method and $B_{\mathrm{GS}}$ that of the Gauß-Seidel method. For matrices $A$ of that kind a few iterations with $\omega_{\mathrm{R}} = 1$ suffice to estimate the value $\rho(B_{\mathrm{GS}})$, which in turn gives an approximation to $\omega_{\mathrm{opt}}$. With our experience with Cryer's projected SOR applied to the valuation of options (Section 4.6) the simple strategy $\omega_{\mathrm{R}} = 1$ is frequently recommendable.

This appendix has merely introduced classic iterative solvers, which are stationary in the sense that the preconditioner matrix $M$ does not vary with $k$. For an overview on advanced nonstationary iterative methods see [Ba94].