TMA4215, fall semester 2011, Numerical methods for linear algebra

Elena Celledoni

1 Introduction

We consider the approximation of the solution of linear systems of algebraic equations with n equations and n unknowns:

$$\begin{array}{rcl} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n & = & b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n & = & b_2 \\ \vdots & \vdots & \vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,n}x_n & = & b_n \end{array} \tag{1}$$

where $(x_1, \cdots, x_n)$ are the unknowns. Given $(a_{i,j})_{1\leq i,j\leq n}$, and b_i, $i = 1, \cdots, n$.

We can rewrite (1) in a matrix form by defining:

$$A := \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & \cdots & a_{2,n} \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & \cdots & a_{n,n} \end{bmatrix}, \quad x := \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad b := \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

so we have that (1) becomes

$$A \cdot x = b. \tag{2}$$

Linear systems can be easily solved when A has a special structure, for example when A is a diagonal matrix or triangular matrix and $a_{i,i} \neq 0$ for $i = 1, \ldots, n$:

- A diagonal matrix

$$A = \begin{bmatrix} a_{1,1} & 0 & \cdots & \cdots & 0 \\ 0 & a_{2,2} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \cdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & a_{n,n} \end{bmatrix} \Rightarrow \begin{array}{rcl|rcl} a_{1,1}x_1 & = & b_1 & x_1 & = & \frac{b_1}{a_{1,1}} \\ a_{2,2}x_2 & = & b_2 & x_2 & = & \frac{b_2}{a_{2,2}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n,n}x_n & = & b_n & x_n & = & \frac{b_n}{a_{n,n}} \end{array}$$

- A triangular matrix

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & \cdots & a_{1,n} \\ 0 & a_{2,2} & a_{2,3} & \cdots & a_{2,n} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & 0 & a_{n-1,n-1} & a_{n-1,n} \\ 0 & 0 & \cdots & 0 & a_{n,n} \end{bmatrix} \Rightarrow$$

$$\begin{array}{rcl} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + \cdots + a_{1,n}x_n & = & b_1 \\ a_{2,2}x_2 + \cdots + \cdots + a_{2,n}x_n & = & b_2 \\ \cdots \quad \vdots \quad \vdots & & \Rightarrow \\ a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n & = & b_{n-1} \\ a_{n,n}x_n & = & b_n \end{array}$$

$$\begin{array}{rcl} x_1 & = & \frac{b_1 - \sum_{j=2}^{n} a_{1,j}x_j}{a_{1,1}} \\ x_2 & = & \frac{b_2 - \sum_{j=3}^{n} a_{2,j}x_j}{a_{2,2}} \\ \vdots & \vdots & \vdots \\ x_{n-1} & = & \frac{b_{n-1} - a_{n-1,n}x_n}{a_{n-1,n-1}} \\ x_n & = & \frac{b_n}{a_{n,n}} \end{array} \qquad (3)$$

The formula (3) is called back substitution algorithm. For general maurices other techniques based on matrix factorizations are in general used. If the matrix is of large size and *sparse* (i.e. it has a relatively small number of elements different from zero), then it is more convenient to use iterative techniques. We will describe here briefly both approaches. In the next section we consider the stability of linear systems, that is the sensitivity of the output with respect to perturbations in the input data.

2 Stability of linear systems

2.1 Preliminaries

A vector norm is a function $\|\cdot\| : \mathbf{R}^n \to \mathbf{R}$ satisfying three axioms:

1. $\|u\| \geq 0$ for all $u \in \mathbf{R}^n$ and $\|u\| = 0$ if and only if $u = 0$.
2. $\|\lambda u\| = |\lambda|\|u\|$for all vectors $u \in \mathbf{R}^n$ and scalars $\lambda \in \mathbf{R}$.
3. $\|u + v\| \leq \|u\| + \|v\|$ for all $u \in \mathbf{R}^n$ and $v \in \mathbf{R}^n$.

If u is a vector with n components, examples of norms are:

- $\|u\|_1 := |u_1| + \ldots + |u_n|$, norm-1;
- $\|u\|_\infty := \max_{1 \leq i \leq n} |u_i|$, max-norm;
- $\|u\|_2 := (|u_1|^2 + \ldots + |u_n|^2)^{\frac{1}{2}} = \left(u^T u\right)^{\frac{1}{2}}$, norm-2.

We will also make use of matrix-norms. Matrix norms satisfy an extra axiom compared to vector norms; this axiom has to do with the product of matrices, so if A and B are $n \times n$ matrices, for a vector- norm according to this axiom we have that

$$\|AB\| \leq \|A\|\|B\|.$$

Assume U is a matrix with elements $u_{i,j}$, examples of matrix-norms are:

- $\|U\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^n u_{i,j}^2}$, Frobenius norm;
- $\|U\|_\infty := \max_i \sum_{j=1}^n |u_{i,j}|$ max-norm;
- $\|U\|_1 := \max_j \sum_{i=1}^n |u_{i,j}|$ norm-1.

For example for

$$U := \begin{bmatrix} 3 & 2 \\ -1 & 0 \end{bmatrix},$$

we get

$$\begin{aligned} \|U\|_F &= \sqrt{3^2 + 2^2 + (-1)^2} = \sqrt{14} = 3.74165738677394, \\ \|U\|_\infty &= \max\{|3| + |2|, |-1| + |0|\} = 5, \\ \|U\|_1 &= \max\{|3| + |-1|, |2| + |0|\} = 4. \end{aligned}$$

2.2 Stability analysis

Consider $Ax = b$, A invertible. Let $x(\varepsilon)$ be the solution of the linear system

$$(A + \varepsilon F)x(\varepsilon) = (b + \varepsilon f).$$

We are intersted in the case when ε tends to zero. Here $f \in \mathbf{R}^n$ and F is $n \times n$ matrix.

We seek for bounds for the relative error

$$\frac{\|x - x(\varepsilon)\|}{\|x\|},$$

for a chosen vector norm $\|\cdot\|$. We will also make use of the so called subordinate matrix norm deduced from $\|\cdot\|$, that is for a given $n \times n$ matrix B:

$$\|B\| := \max_{x \neq 0,\, x \in \mathbf{R}^n} \frac{\|Bx\|}{\|x\|}.$$

Proposition 2.1 *Assume A is invertible. For all ε such that $\varepsilon \leq C_\varepsilon$ then $A + \varepsilon F$ is also invertible.*

Proposition 2.2 *For all ε such that $\varepsilon \leq \tilde{C}_\varepsilon$ thent here exists a unique $x(\varepsilon)$ and its components, $x_i(\varepsilon)$ are continuous functions of ε.*

Lemma 2.3

$$\left.\frac{d}{d\varepsilon} x(\varepsilon)\right|_{\varepsilon=0} = A^{-1}(f - Fx).$$

Proof. By differentiation.

We now use Taylor theorem and obtain

$$x(\varepsilon) = x(0) + \varepsilon x'(0) + \mathcal{O}(\varepsilon^2),$$

where $x(0) = x$. So

$$\|x(\varepsilon) - x\| \leq \varepsilon \|x'(0)\| + \mathcal{O}(\varepsilon^2),$$

and using the lemma we obtain

$$\|x(\varepsilon) - x\| \leq \varepsilon \|A^{-1}(f - Fx)\| + \mathcal{O}(\varepsilon^2).$$

This leads to

$$\frac{\|x(\varepsilon) - x\|}{\|x\|} \leq \varepsilon \frac{\|A^{-1}(f - Fx)\|}{\|x\|} + \mathcal{O}(\varepsilon^2) \leq \varepsilon \|A^{-1}\| \left(\frac{\|f\|}{\|x\|} + \frac{\|Fx\|}{\|x\|} \right) + \mathcal{O}(\varepsilon^2).$$

Now $Ax = b$ implies $\|b\| \leq \|A\|\|x\|$, and this is equivalent to

$$\frac{1}{\|x\|} \leq \frac{\|A\|}{\|b\|}.$$

We also have

$$\frac{\|Fx\|}{\|x\|} \leq \|F\| = \frac{\|F\|\|A\|}{\|A\|}.$$

So proceeding in the estimation of the relative error we get

$$\frac{\|x(\varepsilon) - x\|}{\|x\|} \leq \varepsilon \|A^{-1}\| \left(\frac{\|f\|\|A\|}{\|b\|} + \frac{\|F\|\|A\|}{\|A\|} \right) + \mathcal{O}(\varepsilon^2),$$

and we obtain

$$\frac{\|x(\varepsilon) - x\|}{\|x\|} \leq \varepsilon \|A^{-1}\|\|A\| \left(\frac{\|f\|}{\|b\|} + \frac{\|F\|}{\|b\|} \right) + \mathcal{O}(\varepsilon^2).$$

The real number

$$\mathcal{K}(A) := \|A^{-1}\| \, \|A\|$$

is called **condition number** of A. The condition number depends on A and on the matrix-norm used to measure the relative error. The final bound of the relative error is then

$$\frac{\|x(\varepsilon) - x\|}{\|x\|} \leq \varepsilon \, \mathcal{K}(A) \left(\frac{\|f\|}{\|b\|} + \frac{\|F\|}{\|b\|} \right) + \mathcal{O}(\varepsilon^2),$$

and we clearly see that the condition number gives a bound of the relative error in the output data by means of the relative error in the input data.

We observe that since $I = A^{-1}A$, then

$$\|I\| \leq \|A^{-1}\| \, \|A\| = \mathcal{K}(A),$$

and for all subordinate matrix-norms

$$\|I\| = \max_{x \neq 0, \, x \in \mathbf{R}^n} \frac{\|I \, x\|}{\|x\|} = 1,$$

so

$$1 \leq \mathcal{K}(A).$$

3 Gaussian elimination

Definition. Two linear systems of equations are said to be equivalent if they have the same solution.

Gaussian elimination is an algorithm where in $n-1$ steps, $Ax = b$ is transformed in an equivalent system $Ux = f$ and U is an upper triangular matrix. The advantage is that triangular systems can be easily solved using the formula (3).

In particular we have

$$Ax = b \to A^{(1)}x = b^{(1)} \to \cdots \to A^{(k)}x = b^{(k)} \to \cdots \to A^{(n-1)}x = b^{(n-1)}$$

all intermediate linear systems $A^{(k)}x = b^{(k)}$ $k = 1, \ldots, n-1$ are equivalent to $Ax = b$. The last system $A^{(n-1)}x = b^{(n-1)}$, is in upper triangular form, and

system number k is

$$A^{(k)} = \begin{bmatrix} \tilde{a}_{1,1} & \cdots & \tilde{a}_{1,k} & \tilde{a}_{1,k+1} & \cdots & \tilde{a}_{1,n} \\ 0 & \ddots & \vdots & \vdots & \cdots & \vdots \\ \vdots & \ddots & \tilde{a}_{k,k} & \tilde{a}_{k,k+1} & \cdots & \tilde{a}_{k,n} \\ \vdots & \vdots & 0 & \tilde{a}_{k+1,k+1} & \cdots & \tilde{a}_{k+1,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & \tilde{a}_{n,k+1} & \cdots & \tilde{a}_{n,n} \end{bmatrix}.$$

We obtain $A^{(1)}x = b^{(1)}$ from $Ax = b$ by replacing the den 2nd, 3rd, .., nth equation in $Ax = b$ with corresponding linear combinations of the first equation with the 2nd, 3rd, .., nth equation. To obtain $A^{(2)}x = b^{(2)}$ (and the subsequent linear systems) the same process is repeated considering only prosessen er repetert med å ta hensyn til kolonnene fra the columns form the 2nd to the nth and the rows from the 2nd to the nth.

3.1 Example

Given:

$$\begin{array}{rcl} x_1 + 4x_2 + x_3 &=& 6 \\ 2x_1 - x_2 - 2x_3 &=& 3 \\ x_1 + 3x_2 + 2x_3 &=& 5 \end{array} \qquad A = \begin{bmatrix} 1 & 4 & 1 \\ 2 & -1 & -2 \\ 1 & 3 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 3 \\ 5 \end{bmatrix} \tag{4}$$

we want to reduce it t a triangular form.

We start with replacing the second equation with a linear combination of the first two equations, that is we replace
$2x_1 - x_2 - 2x_3 = 3$ with

$$(2x_1 - x_2 - 2x_3) + (-2) \cdot (x_1 + 4x_2 + x_3) = 3 + (-2) \cdot 6, \Rightarrow -9x_2 - 4x_3 = -9.$$

We then replace the 3rd equation with a linear combination of the 3rd and 1st equation:

$$(x_1 + 3x_2 + 2x_3) + (-1) \cdot (x_1 + 4x_2 - x_3) = 5 + (-1) \cdot 6, \Rightarrow -x_2 + x_3 = -1.$$

This way we get the following new system

$$\begin{array}{rcl} x_1 + 4x_2 + x_3 &=& 6 \\ -9x_2 - 4x_3 &=& -9 \\ -x_2 + x_3 &=& -1 \end{array} \qquad A^{(1)} = \begin{bmatrix} 1 & 4 & 1 \\ 0 & -9 & -4 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ -9 \\ -1 \end{bmatrix} \tag{5}$$

The coefficients used in the linear combination of the equations are $\begin{bmatrix} 2 \\ 1 \end{bmatrix}$, and they are chosen so that in the new system we get that the second and third element of the first column vanish, this way we *eliminate* to coefficients in the linear system.

Now we work only with the two last equations:

$$\begin{aligned} -9x_2 - 4x_3 &= -9 \\ -x_2 + x_3 &= -1 \end{aligned} \tag{6}$$

We replace the last equaltion with

$$(-x_2 + x_3) + (-\frac{1}{9}) \cdot (-9x_2 - 4x_3) = -1 + \left(-\frac{1}{9}\right) \cdot (-9) \Rightarrow \frac{13}{9} x_3 = 0.$$

The coefficient used for the linear combination is $\frac{1}{9}$.

In the end we get the linear system

$$\begin{aligned} x_1 + 4x_2 + x_3 &= 6 \\ -9x_2 - 4x_3 &= -9 \\ \tfrac{13}{9} x_3 &= 0 \end{aligned} \qquad A^{(2)} = \begin{bmatrix} 1 & 4 & 1 \\ 0 & -9 & -4 \\ 0 & 0 & \frac{13}{9} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ -9 \\ 0 \end{bmatrix} \tag{7}$$

and the solution can be computed by the backward substitution algorithm. Starting form the last equation $x_3 = 0$, and proceeding upwards to solve $-9x_2 = -9 \Rightarrow x_2 = 1$ og $x_1 + 4 = 6 \Rightarrow x_1 = 2$, we get

$$x = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}.$$

Note now that when we *eliminated* the first column we used the two coefficients:

$$\begin{bmatrix} 2 \\ 1 \end{bmatrix},$$

and for the second we used

$$\frac{1}{9}.$$

By using these coefficients we construct a triangular matrix L:

$$L := \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & \frac{1}{9} & 1 \end{bmatrix}.$$

Note also that if from (7) we define now $U := A^{(2)}$ and we compute $L \cdot U$, we get A back.

In general when we perform the Gaussian elimination, we compute simultaneously a factorization of the matrix $A = LU$, where L and U are two triangular matrices.

This is very handy if we want to compute solution of two or more systems with the same coefficient matrix A but different right hand sides, b, $\hat{b}$ and so on. In this case one can use the same factorization $LU = A$ two (or more) times ganger and compute to different backward substitutions, one with b and one with $\hat{b}$.

Such factorization is also used for computing the determinant of A, because $\det(A) = \prod_{i=1}^{n} u_{i,i}$. Analogously one can use the factorization to find the inverse A^{-1}. In our case we have

$$\det(A) = 1 \cdot (-9) \cdot \frac{13}{9} = -13$$

and

$$A^{-1} = \begin{bmatrix} 1 & 4 & 1 \\ 0 & -9 & -4 \\ 0 & 0 & \frac{13}{9} \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & \frac{1}{9} & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & \frac{4}{9} & \frac{7}{13} \\ 0 & -\frac{1}{9} & -\frac{4}{13} \\ 0 & 0 & \frac{9}{13} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -\frac{7}{9} & -\frac{1}{9} & 1 \end{bmatrix}$$

Note that it is much easier to compute the inverse of a triangular matrix than the inverse of a general invertible but unstructured matrix.

3.2 Gauss elimination: general algorithm

For the general matrix (1) we have:

$$Ax = b \to A^{(1)}x = b^{(1)}$$

if $a_{1,1} \neq 0$

$$\left.\begin{array}{lllllllll} l_{2,1} & := & \frac{a_{2,1}}{a_{1,1}} & a_{2,p}^{(1)} & := & a_{2,p} - l_{2,1}a_{1,p} & b_2^{(1)} & := & b_2 - l_{2,1}b_1 \\ l_{3,1} & := & \frac{a_{3,1}}{a_{1,1}} & a_{3,p}^{(1)} & := & a_{3,p} - l_{3,1}a_{1,p} & b_3^{(1)} & := & b_3 - l_{3,1}b_1 \\ \vdots & & & & & & & & \\ l_{n,1} & := & \frac{a_{n,1}}{a_{1,1}} & a_{n,p}^{(1)} & := & a_{n,p} - l_{n,1}a_{1,p} & b_n^{(1)} & := & b_n - l_{n,1}b_1 \end{array}\right\} p = 2, \ldots, n.$$

Now we define

$$A := A^{(1)}, b := b^{(1)}$$

and continue with

$$Ax = b \to A^{(2)}x = b^{(2)}.$$

If $a_{2,2} \neq 0$, we obtain, for $j = 3, \ldots, n$,

$$l_{j,2} := \frac{a_{j,2}}{a_{2,2}} \quad a_{j,p}^{(2)} := a_{j,p} - l_{j,2}a_{2,p} \quad b_j^{(2)} := b_j - l_{j,2}b_2 \quad p = 3, \ldots, n,$$

and then we define

$$A := A^{(2)}, b := b^{(2)}.$$

In general for $A^{(k)}$ we have,

$$A := A^{(k-1)}, b := b^{(k-1)}$$

and

$$Ax = b \to A^{(k)}x = b^{(k)}$$

with

$$l_{j,k} := \frac{a_{j,k}}{a_{k,k}} \quad a_{j,p}^{(k)} := a_{j,p} - l_{j,k}a_{k,p} \quad b_j^{(k)} := b_j - l_{j,k}b_k \quad p = k+1, \ldots, n.$$

for $j = k+1, \ldots n$, and assuming $a_{k,k} \neq 0$. In the end we obtain the following algorithm:

Gaussian elimination

For $k = 1, \ldots, n-1$

For $j = k+1, \ldots, n$

$l_{j,k} := \frac{a_{j,k}}{a_{k,k}}$

For $p = k+1, \ldots, n+1$,

$a_{j,p} := a_{j,p} - l_{j,k} a_{k,p}$

End

End

End

And with $U := A^{(n-1)}$ we also obtain the following, so called LU**-factorization** for A,

$$A = LU.$$

3.3 Gaussian elimination with partial pivoting

In Gaussisan elimination (G-E), as described on the general algorithm, we divide always by $a_{k,k}$ (the so called pivot element). Obviously we might get problems when such value is zero or very small. To avoid such problems we can perform systematic permutationns of the rows of the linear system. This procedure is called partial pivoting.

3.4 Two exmples: Gaussian elimination with partial pivoting

Consider the linear system

$$\begin{array}{rcl} x_1 + x_2 + x_3 + x_4 &=& 1 \\ x_1 + x_2 + 2x_3 - x_4 &=& 1 \\ x_1 + 2x_2 - x_3 - x_4 &=& 1 \\ x_1 - x_2 + x_3 - x_4 &=& 1 \end{array} \qquad \left[\begin{array}{cccc} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & -1 \\ 1 & 2 & -1 & -1 \\ 1 & -1 & 1 & -1 \end{array}\right] \tag{8}$$

We eliminate the first column by subtracting the first row from the other three. We obtain

$$\begin{array}{rcl} x_1 + x_2 + x_3 + x_4 &=& 1 \\ x_3 - 2x_4 &=& 0 \\ x_2 - 2x_3 - 2x_4 &=& 0 \\ -2x_2 - 2x_4 &=& 0 \end{array} \qquad \left[\begin{array}{cccc} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & -2 \\ 0 & 1 & -2 & -2 \\ 0 & -2 & 0 & -2 \end{array}\right] \tag{9}$$

now we can not proceede, because there is no α which could be used to eliminate x_2

$$(x_2 - 2x_3 - 2x_4) - \alpha \cdot (x_3 - 2x_4) = 0.$$

The only option is to permute the rows. To minimize roundoff error propagation it pays off to exchange the second row with the row having the biggest coefficient in absolute value for x_2, that is the third. Then we get

$$\begin{array}{rcl} x_1 + x_2 + x_3 + x_4 &=& 1 \\ -2x_2 - 2x_4 &=& 0 \\ x_2 - 2x_3 - 2x_4 &=& 0 \\ x_3 - 2x_4 &=& 0 \end{array} \qquad \left[\begin{array}{cccc} 1 & 1 & 1 & 1 \\ 0 & -2 & 0 & -2 \\ 0 & 1 & -2 & -2 \\ 0 & 0 & 1 & -2 \end{array}\right] \tag{10}$$

and now we continue the Gaussian elimination as usual. We obtain:

$$\begin{array}{rcl} x_1 + x_2 + x_3 + x_4 &=& 1 \\ -2x_2 - 2x_4 &=& 0 \\ -2x_3 - 3x_4 &=& 0 \\ x_3 - 2x_4 &=& 0 \end{array} \qquad \left[\begin{array}{cccc} 1 & 1 & 1 & 1 \\ 0 & -2 & 0 & -2 \\ 0 & 0 & -2 & -3 \\ 0 & 0 & 1 & -2 \end{array}\right] \tag{11}$$

and in the end:

$$\begin{array}{rcl} x_1 + x_2 + x_3 + x_4 &=& 1 \\ -2x_2 - 2x_4 &=& 0 \\ -2x_3 - 3x_4 &=& 0 \\ -\frac{7}{2}x_4 &=& 0 \end{array} \qquad \left[\begin{array}{cccc} 1 & 1 & 1 & 1 \\ 0 & -2 & 0 & -2 \\ 0 & 0 & -2 & -3 \\ 0 & 0 & 0 & -\frac{7}{2} \end{array}\right] \tag{12}$$

which, by the backward substitution algorithm, gives the following solution

$$x = \left[\begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \end{array}\right].$$

In the second example given:

$$\begin{array}{rcl} x_1 + 3x_2 + 2x_3 &=& 5 \\ 2x_1 - x_2 - 2x_3 &=& 3 \\ x_1 + 4x_2 + x_3 &=& 6 \end{array} \qquad A = \begin{bmatrix} 1 & 3 & 2 \\ 2 & -1 & -2 \\ 1 & 4 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 5 \\ 3 \\ 6 \end{bmatrix} \tag{13}$$

we want to reduce the system to triangular form using Gaussian elimination with partial pivoting.

We exchange rows:

$$\begin{array}{rcl} 2x_1 - x_2 - 2x_3 &=& 3 \\ x_1 + 3x_2 + 2x_3 &=& 5 \\ x_1 + 4x_2 + x_3 &=& 6 \end{array} \tag{14}$$

then we eliminate x_1,

$$\begin{array}{rcl} 2x_1 - x_2 - 2x_3 &=& 3 \\ 3.5x_2 + 3x_3 &=& 3.5 \\ 4.5x_2 + 2x_3 &=& 4.5 \end{array} \tag{15}$$

we exchange rows once more:

$$\begin{array}{rcl} 2x_1 - x_2 - 2x_3 &=& 3 \\ 4.5x_2 + 2x_3 &=& 4.5 \\ 3.5x_2 + 3x_3 &=& 3.5 \end{array} \tag{16}$$

and eliminate x_2 from the last equation,

$$\begin{array}{rcl} 2x_1 - x_2 - 2x_3 &=& 3 \\ 4.5x_2 + 2x_3 &=& 4.5 \\ 1.4444x_3 &=& 0 \end{array} \tag{17}$$

with the backward substitution algorithm we obtain the solution $x_3 = 0$, $x_2 = 1$, $x_1 = 2$.

3.5 G-E with partial pivoting: general algorithm

In general we get the following Gaussian elimination algorithm with partial pivoting: first one initializes the vector π (pivot vektor) so that $\pi_i := i$ for $i = 1, \ldots n$,

Gaussian elimination with partial pivoting

For $k = 1, \ldots, n-1$

$a := |a_{k,k}|$

  For $j = k+1, \ldots, n$

    if ($a < |a_{j,k}|$)

      $a := |a_{j,k}|$

      $\pi_k := j$

    End

  End

  if ($\pi_k \neq k$)

    $s := \pi_k$

    For $p = k, \ldots, n$

      $r := a_{k,p}$

      $a_{k,p} := a_{s,p}$

      $a_{s,p} := r$

    End

  End

  For $j = k+1, \ldots, n$

    $l_{j,k} := \frac{a_{j,k}}{a_{k,k}}$

    For $p = k+1, \ldots, n+1,$

      $a_{j,p} := a_{j,p} - l_{j,k} a_{k,p}$

    End

  End

End

3.6 Complexity of Gaussian elimination

One can prove that for a $n \times n$ matrix the complexity of Gaussian elimination is of

$$\frac{1}{3}n^3 - \frac{1}{3}n,$$

operations (additions and multiplications). For big n $\frac{1}{3}n^3$ dominates the cost. We say that Gaussian elimination has complexity $\mathcal{O}(n^3)$. The backward substitution algorithm, (3)) has lower cost, that is $\mathcal{O}(n^2)$.

4 Other matrix factorizations

Bisides the LU-factorizations there are many other important matrix factorizations which is useful to know about.

Recall that an eigenvalue of A is a real or complex value such that, there is u vector such that

$$Au = \lambda u.$$

u is called eigenvector of A.

1. QR**- factorization**: any real matrix A $n \times p$ can be factorized in the form

   $$A = QR$$

   where Q is $n \times n$ orthogonal (i.e. $Q^TQ = QQ^T = I$ where I is the identity matrix) and R is $n \times p$ is upper triangular.

2. **Polar decomposition**: any matrix A $n \times n$ can be factorized in the form

   $$A = QS$$

   where Q is $n \times n$ orthogonal and S is $n \times n$ is symmetric.

3. **Schur canonical form**: for any matrix A $n \times n$ there exists a matrix P (complex) unitary (i.e. $P^HP = PP^H = I$ and P^H the transpose-conjugate of P) such that

   $$P^HAP = T$$

where T is upper triangular.

As a consequence, if A is Hermitian (i.e. $A = A^H$) then

$$P^H A P = P^H A^H P = T$$

and T is Hermitian and triangular and therefore is diagonal.

4. **Singular value decomposition**. For all A $n \times n$ real matrices, A can be factorized as
$$A = U \Sigma V^T$$
where Σ is diagonal, U and V are $n \times n$ orthogonal matrices. The diagonal elements of Σ are the singular values of A, i.e. the square roots of the eigenvalues of $A^T A$. This factorization has analogs for $n \times p$ matrices and for matrices with complex entries.

5. **Jordan canonical form**. For any A real $n \times n$ (or $A \in C^{n \times n}$) it exists a matrix $M \in C^{n \times n}$ invertible, such that
$$M^{-1} A M = J = \begin{bmatrix} J_1 & & & \\ & J_2 & & \\ & & \ddots & \\ & & & J_k \end{bmatrix}, \qquad \text{(block-diagonal)}. \tag{18}$$
Here J_i is a $m_i \times m_i$-matrix, and $\sum_{i=1}^{k} m_i = n$. The *Jordan-blocks* J_i have the form
$$J_i = \begin{bmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{bmatrix}, \quad \text{if } m_i \geq 2$$
and $J_i = [\lambda_i]$ if $m_i = 1$. If all $m_i = 1$, then $k = n$ and the matrix is diagonalizable. If A has n distinct egenvalues, it is always diagonalizable. The converse is not true, that is a matrix can be diagonalizable even if it has multiple eigenvalues.

5 Symmetric matrices

When we talk about symmetric matrices, we mean normally *real* symmetric matrices. The *transpose* A^T of a $m \times n$-matrix A, is a $n \times m$-matrix with a_{ji} as the (ij)-element (a matrix whose columns are the rows of A). A $n \times n$ matrix is symmetric if $A^T = A$.

A symmetric $n \times n$ matrix has real eigenvalues $\lambda_1, \ldots, \lambda_n$ and a set of real orthonormal eigenvectors $x_1, \ldots, x_n$. Let $\langle \cdot, \cdot \rangle$ denote the standard inner-product on C^n, then $\langle x_i, x_j \rangle = \delta_{ij}$ (Kronecker-delta).

A consequence of this is that the matrix of eigenvectors $X = [x_1, \ldots, x_n]$ is real and orthogonal and its inverse is therefore the transpose

$$X^{-1} = X^T.$$

The diagonalization of A is given by

$$\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n), \quad X = [x_1, \ldots, x_n], \quad X^T X = I, \quad X^T A X = \Lambda \Leftrightarrow A = X \Lambda X^T$$

5.1 Positive definite matrices

If A is symmetric and $\langle x, Ax \rangle = x^T A x > 0$ for all $0 \neq x \in R^n$ A is called *positive definite*. Here we denote with $\langle \cdot, \cdot \rangle$ the Euclidean inner product.

A (symmetric) is positive semi-definite if $\langle x, Ax \rangle \geq 0$ for all $x \in R^n$ and $\langle x, Ax \rangle = 0$ for at least a $x \neq 0$.

A positive definite $\Leftrightarrow$ A has only positive eigenvalues.

A positive semi-definite $\Leftrightarrow$ A has only non-negative eigenvalues, and at least a 0-eigenvalue.

6 Gershgorin's theorem

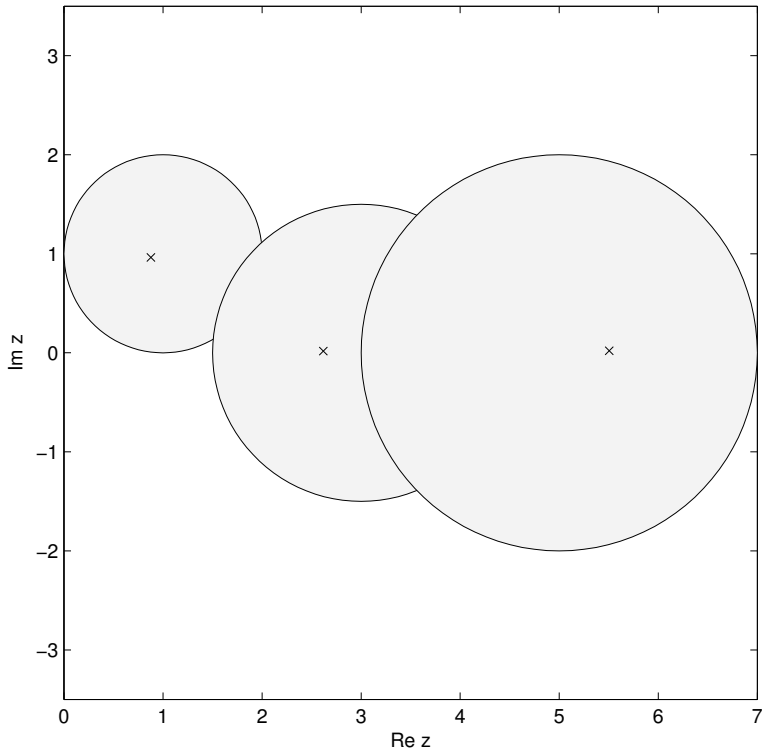
Gershgorin's theorem. Is given $A = (a_{ik}) \in C^{n \times n}$. Define n disks S_j in the complex plane by

$$S_j = \left\{ z \in C : |z - a_{jj}| \leq \sum_{k \neq j} |a_{jk}| \right\}.$$

The union $S = \bigcup_{j=1}^{n} S_j$ contains all the eigenvalues of A. For every eigenvalue λ of A there is a j such that $\lambda \in S_j$.

Example.

$$A = \begin{bmatrix} 1+i & 1 & 0 \\ 0.5 & 3 & 1 \\ 1 & 1 & 5 \end{bmatrix}.$$



Proof of Gershgorin's theorem: Let λ be a eigenvalue with associate eigenvector $x = [\xi_1, \ldots, \xi_n]^T \neq 0$. Choose ℓ among the indexes $1, \ldots, n$ such that $|\xi_\ell| \geq |\xi_k|,\ k = 1, \ldots, n$, and so $|\xi_\ell| > 0$. The equation $Ax = \lambda x$ has component ℓ:

$$\sum_{k=1}^{n} a_{\ell k}\xi_k = \lambda\,\xi_\ell \ \Rightarrow\ (\lambda - a_{\ell\ell})\xi_\ell = \sum_{k\neq\ell} a_{\ell k}\xi_k$$

Divide by $|\xi_\ell|$ on each side and take the absolute value

$$|\lambda - a_{\ell\ell}| \quad = \quad \left| \sum_{k \neq \ell} a_{\ell k} \frac{\xi_k}{\xi_\ell} \right| \quad \leq \quad \sum_{k \neq \ell} |a_{\ell k}| \frac{|\xi_k|}{|\xi_\ell|} \quad \leq \quad \sum_{k \neq \ell} |a_{\ell k}|$$

Then we get $\lambda \in S_\ell$.

Example. Diagonally dominant matrices with positive diagonal elements are positive definite. Why?

7 Solution of linear systems by iteration

To approximate x in the numerical solution of $Ax = b$, given $x^{(0)}$, we construct a sequence of vectors $x^{(1)}, \ldots, x^{(n)}, \ldots$. A way to do this is by fixed-point iteration.

We consider an equivalent formulation of $Ax = b$ as fix-point equaltion. For example:

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

where I is the $n \times n$ identity matrix. Given x_0, we then obtain the iteration:

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

In general one can obtain an iteration as follows:

- write A as a sum of two terms: $A = M - N$, choose M invertible;
- from $(M - N)x = b$ one gets $Mx = Nx + b$ and

$$x = M^{-1}Nx + M^{-1}b;$$

- so given x^0 one builds the iteration:

$$x^{(n+1)} = M^{-1}Nx^{(n)} + M^{-1}b.$$

Typically M is chosen such that M^{-1} is easy to compute, for example M can be diagonal or triangular.

7.1 Example

We want to solve with fix-point-iteration the system

$$\begin{array}{rcl} x_1 - x_2 & = & 1 \\ -x_1 + 2x_2 & = & -1 \end{array} \quad A := \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}, \quad \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \tag{19}$$

and start with

$$x^{(0)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The solution is

$$x = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

We take

$$M := \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \quad N = M - A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

We have $Mx = Nx + b$, and $x = M^{-1}Nx + M^{-1}b$, i.e.

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

By using $x^{(0)}$ we compute

$$x^{(1)} = M^{-1}Nx^{(0)} + M^{-1}b,$$

$$\begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ -\frac{1}{2} \end{bmatrix}.$$

We continue $x^{(2)} = M^{-1}Nx^{(1)} + M^{-1}b$:

$$\begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \end{bmatrix} + \begin{bmatrix} 1 \\ -\frac{1}{2} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ 0 \end{bmatrix}$$

and more

$$\begin{bmatrix} x_1^{(3)} \\ x_2^{(3)} \end{bmatrix} = \begin{bmatrix} 1 \\ -0.2500 \end{bmatrix}, \quad \begin{bmatrix} x_1^{(4)} \\ x_2^{(4)} \end{bmatrix} = \begin{bmatrix} 0.7500 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} x_1^{(5)} \\ x_2^{(5)} \end{bmatrix} = \begin{bmatrix} 1 \\ -0.1250 \end{bmatrix},$$

$$\begin{bmatrix} x_1^{(6)} \\ x_2^{(6)} \end{bmatrix} = \begin{bmatrix} 0.8750 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} x_1^{(28)} \\ x_2^{(28)} \end{bmatrix} = \begin{bmatrix} 1.0000 \\ -0.0000 \end{bmatrix}.$$

7.2 Example

In the system (19) we take

$$M := \begin{bmatrix} 1 & 0 \\ -1 & 2 \end{bmatrix} \quad N = M - A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix},$$

and with the same $x^{(0)}$ we compute $x^{(1)} = M^{-1}Nx^{(0)} + M^{-1}b$:

$$\begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

and we get the true solution of the system already at the first iteration.

In general if M is the diagonal part of A, i.e.

$$m_{i,i} = a_{i,i}, \quad i = 1, \ldots, n, \quad m_{i,j} = 0, \quad i \neq j,$$

(where $m_{i,j}$ are the elements of M, and $a_{i,j}$ are the elements of A), then we obtain the so called **Jacobi method**.

If M is the lower triangular part of A, i.e.

$$m_{i,j} = a_{i,j}, \quad i = 1, \ldots, n, \; j = 1, \ldots, i, \quad m_{i,j} = 0, \quad i = 1, \ldots, n, \; j = i{+}1, \ldots, n,$$

then the method is called **Gauss-Seidel method**.

7.3 Convergence

To measure the extent to which x^n has converged to x we use vector norms: $\|x - x^{(n)}\|$.

We write the iteration in the following general way

$$x^{(n+1)} = M^{-1}Nx^{(n)} + M^{-1}b,$$

by defining $C := M^{-1}N$ and $g := M^{-1}b$ we can write:

$$x^{(n+1)} = Cx^{(n)} + g. \tag{20}$$

Theorem 7.1 *If there is a matrix-norm $\|\cdot\|$ such that $\|C\| < 1$ the iteration (20) converges for all $x^{(0)}$.*

7.4 Example

Consider

$$B = \begin{bmatrix} 3 & 1 \\ -1 & 2.5 \end{bmatrix}, M = \begin{bmatrix} 3 & 0 \\ 0 & 2.5 \end{bmatrix}, N = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

then

$$C_J = M^{-1}N = \begin{bmatrix} 0 & -0.3333 \\ 0.4000 & 0 \end{bmatrix}$$

and $\|C_J\|_F = 0.5207$, $\|C_J\|_1 = \|C_J\|_{\max} = 0.4000$, so given f, the Jacobi method converges for $Bx = f$ for any $x^{(0)}$.

For the Gauss-Seidel method we have

$$C_{GS} = \begin{bmatrix} 3 & 0 \\ -1 & 2.5 \end{bmatrix}^{-1} \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -0.3333 \\ 0 & -0.1333 \end{bmatrix}$$

and $\|C_{GS}\|_F = 0.3590$, $\|C_{GS}\|_1 = 0.4667$ and $\|C_{GS}\|_{\max} = 0.3333$, so Gauss-Seidel method converges for $Bx = f$ and for any $x^{(0)}$.

7.5 Example

Consider the matrix from the example (19):

$$A := \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix},$$

Both Jacobi and Gauss-Seidel converge. For the Jacobi method we have:

$$C_J = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{1}{2} & 0 \end{bmatrix}$$

with norms $\|C_J\|_F \geq 1$ $\|C_J\|_{\max} \geq 1$ $\|C_J\|_1 \geq 1$. For Gauss-Seidel

$$C_{GS} = \begin{bmatrix} 1 & 0 \\ -1 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & \frac{1}{2} \end{bmatrix}$$

and $\|C_{GS}\|_F \geq 1$ $\|C_{GS}\|_{\max} \geq 1$ $\|C_{GS}\|_1 \geq 1$.

Here the hypothesis of the previous theorem are not satisfied, even if in our numerical experiments the iteration converges. We should use another

technique to prove convergence. Let $\sigma(C)$ be the set of eigenvalues of C, then we can define

$$\rho(C) := \max_{\lambda \in \sigma(C)} |\lambda|$$

$\rho(C)$ is named spectral radius of C, and it is a positive number.

One can show that

Theorem 7.2 *$\rho(C) < 1$ if and only if the iterative method (20) converges for all $x^{(0)}$.*

In our example we must look at the eigenvalues of C_J and C_{GS}. Note that

$$Cu = \lambda u \Leftrightarrow (C - \lambda I)u = 0 \Leftrightarrow \det(C - \lambda I) = 0,$$

where I is the identity matrix and

$$\det(U) = \det\left(\begin{bmatrix} u_{1,1} & u_{1,2} \\ u_{2,1} & u_{2,2} \end{bmatrix}\right) = u_{1,1} \cdot u_{2,2} - u_{1,2} \cdot u_{2,1}.$$

So

$$\det(C_J - \lambda I) = \det\left(\begin{bmatrix} 0 & 1 \\ \frac{1}{2} & 0 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}\right) = \det\left(\begin{bmatrix} -\lambda & 1 \\ \frac{1}{2} & -\lambda \end{bmatrix}\right) = \lambda^2 - \frac{1}{2},$$

and $\lambda_{1,2}^J = \pm\sqrt{\frac{1}{2}} \approx \pm 0.7071$. So $\rho(C) = 0.7071$ and $\rho(C) < 1$.

For C_{GS} we have

$$\det(C_J - \lambda I) = \det\left(\begin{bmatrix} -\lambda & 1 \\ 0 & \frac{1}{2} - \lambda \end{bmatrix}\right) = \lambda\left(\frac{1}{2} - \lambda\right),$$

and $\lambda_1^{GS} = 0$ and $\lambda_2^{GS} = \frac{1}{2}$ and $\rho(C_{GS}) = \frac{1}{2}$.

The reason why Gauss-Seidel converges faster than Jacobi is that $\rho(C_{GS}) \leq \rho(C_J)$.

8 Krylov subspace methods

Given x^0 an initial guess for the solution x of the linear system $Ax = b$, A $n \times n$, consider the residual vector

$$r^0 := b - Ax^0.$$

The Krylov subspace generated by A and r^0 is the subspace of $\mathbf{R}^n$ defined by

$$\mathcal{K}^m(A, r^0) := \text{span}\{r^0, Ar^0, \dots A^{m-1}r^0\}.$$

This subspace has dimension less than or equal to m.

A **Krylov subspace method** is as method where the m-th approximation is such that

$$x^m = x^0 + v, \quad v \in \mathcal{K}^m(A, r^0)$$

and the residual $r^m := b - Ax^m$ satisfies

$$r^m \perp w, \quad \forall w \in \mathcal{L}^m,$$

where $\mathcal{L}^m$ is a subspace of $\mathbf{R}^n$ of dimension less than or equal to m.

Typical choices for $\mathcal{L}^m$ are:

$$\mathcal{L}^m = \begin{cases} \mathcal{K}^m(A, r^0) \\ A\mathcal{K}^m(A, r^0) \end{cases}$$

The first choice leads to the so called Full Orthogonalization Method (FOM). When A is symmetric and positive definite, this method is equivalent to the famous Conjugate Gradient method (CG) . The second choice gives rise to the so called Generalized Minimal Residual method (GMRES). For more details on the implementation of these iterative techniques please refer to numerical linear algebra courses.

9 Preconditioning

Given

$$Ax = b$$

we want to find $M \approx A$ and M easy to "invert", such that

$$M^{-1}Ax = M^{-1}b$$

is "easier" to solve with an iterative method, i.e. the iterative methods converge faster. This is typically achieved if the choice of M leads to

$$\mathcal{K}(M^{-1}A) \leq \mathcal{K}(A),$$

where $\mathcal{K}(A)$ denotes the condition number of A.

To get a satisfactory preconditioner we need:

- $M \approx A$ must be such that linear systems of the type

$$Mz = w$$

  are easy to solve;

- $M \approx A$ must be a good approximation of A such that the product $M^{-1}A$ is close to the identity matrix, this implies that $\mathcal{K}(M^{-1}A)$ is small compared to $\mathcal{K}(A)$.

If one sapiently constructs M following the above guidelines, the iteration will converge significantly faster to the solution of the linear system. In the implementation one should carefully handle the extra operations involved in applying a Krylov subspace method to the preconditioned system instead of to the original system, in order not to loose all the computational gain given by using a preconditioner.