

4.6 Iterative Solvers for Linear Systems

Why use iterative methods?

Virtually all direct methods for solving $Ax = b$ require $\mathcal{O}(n^3)$ floating point operations. In practical applications the matrix A often has a very special structure, and possibly contains many zero entries (is sparse). It is especially in these cases where iterative methods pay off. As an example, recall the discretized Poisson problem from the very first class meeting. Certain iterative methods were mentioned that can solve the resulting linear system in $\mathcal{O}(n)$ operations. We now study some of these methods in detail.

4.6.1 Classical Iterative Solvers

We begin with a simple example.

Example: Consider the (iterative) solution of the linear system (see also the Maple worksheet `577_Iterative_Solvers.mws`)

$$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 6 \end{bmatrix}.$$

The simplest approach is to use a fixedpoint iteration scheme. Thus, we solve the first equation for x_1 , and the second equation for x_2 , and start the iteration with some initial guess $x^{(0)} = [x_1^{(0)} x_2^{(0)}]^T$. This results in

$$\begin{aligned} x_1^{(k)} &= (6 - x_2^{(k-1)}) / 2 \\ x_2^{(k)} &= (6 - x_1^{(k-1)}) / 2, \end{aligned}$$

for $k = 1, 2, 3, \dots$. The method just described is known as *Jacobi iteration*, and a general algorithm is given by

Algorithm (Jacobi iteration)

```
Input  $A, b, x, n, M$ 
for  $k = 1$  to  $M$  do
  for  $i = 1$  to  $n$  do
     $u_i = \left( b_i - \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij} x_j \right) / a_{ii}$ 
  end
  for  $i = 1$  to  $n$  do
     $x_i = u_i$ 
  end
end
Output  $x$ 
```

An obvious improvement of the Jacobi iteration is to use the most recent estimate for x_i as soon as it becomes available. For the example above this results in

$$\begin{aligned}x_1^{(k)} &= (6 - x_2^{(k-1)})/2 \\x_2^{(k)} &= (6 - x_1^{(k)})/2,\end{aligned}$$

for $k = 1, 2, 3, \dots$. This method is known as *Gauss-Seidel iteration*, and the general algorithm is

Algorithm (Gauss-Seidel iteration)

Input A, b, x, n, M

for $k = 1$ to M do

for $i = 1$ to n do

$$x_i = \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij} x_j \right) / a_{ii}$$

end

end

Output x

Remarks:

1. Both algorithms can be made more efficient by moving the division to a preprocessing step.
2. For a generic problem Gauss-Seidel iteration converges faster than Jacobi iteration. However, there are certain problems for which Jacobi iteration may converge faster.
3. Jacobi iteration has the advantage that it can be implemented on a parallel computer in a straightforward manner.

In a more abstract setting we want to consider the solution of the linear system $Ax = b$ by an iterative algorithm of the form

$$x^{(k)} = Gx^{(k-1)} + c, \quad k = 1, 2, 3, \dots, \quad (14)$$

where $x^{(0)}$ is our initial guess, G is a (constant) iteration matrix, and c is a (constant) vector. All of the classical solvers discussed here are based on a *splitting* of A of the form

$$A = A - Q + Q$$

with splitting matrix Q . Thus, provided Q^{-1} exists,

$$\begin{aligned}Ax = b &\Leftrightarrow Qx + (A - Q)x = b \\ &\Leftrightarrow Qx = (Q - A)x + b\end{aligned} \quad (15)$$

$$\Leftrightarrow x = \underbrace{(I - Q^{-1}A)}_{=G} x + \underbrace{Q^{-1}b}_{=c}. \quad (16)$$

Our goals in choosing the splitting matrix Q are

1. Q should be invertible, i.e., we want to be able to solve (15) efficiently.
2. The iteration (14) should converge quickly to a fixed point.

The two extreme choices $Q = A$ and $Q = I$ correspond to the original problem $Ax = b$ (so that nothing has been gained), and the so-called *Richardson method* $x^{(k)} = (I - A)x^{(k-1)} + b = x^{(k-1)} + r^{(k-1)}$ (for which no system needs to be solved, but which – depending on A – may not converge). Therefore, one usually chooses Q somewhere between these two extremes.

We can see that the Jacobi iteration as defined above corresponds to the choice $Q = \text{diag}(A)$. To this end,

$$Q^{-1}A = \begin{bmatrix} \frac{1}{a_{11}} & & & \\ & \frac{1}{a_{22}} & & \\ & & \ddots & \\ & & & \frac{1}{a_{nn}} \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} \frac{a_{11}}{a_{11}} & \frac{a_{12}}{a_{11}} & \cdots & \frac{a_{1n}}{a_{11}} \\ \frac{a_{21}}{a_{22}} & \frac{a_{22}}{a_{22}} & \cdots & \frac{a_{2n}}{a_{22}} \\ \vdots & & \ddots & \vdots \\ \frac{a_{n1}}{a_{nn}} & \frac{a_{n2}}{a_{nn}} & \cdots & \frac{a_{nn}}{a_{nn}} \end{bmatrix},$$

i.e., the rows of A are scaled by the reciprocals of the diagonal entries. Then we have

$$Q^{-1}Ax = \begin{bmatrix} (\sum a_{1j}x_j)/a_{11} \\ (\sum a_{2j}x_j)/a_{22} \\ \vdots \\ (\sum a_{nj}x_j)/a_{nn} \end{bmatrix} \quad \text{and} \quad Q^{-1}b = \begin{bmatrix} b_1/a_{11} \\ b_2/a_{22} \\ \vdots \\ b_n/a_{nn} \end{bmatrix},$$

so that (using (16) componentwise)

$$\begin{aligned} x_i^{(k)} &= x_i^{(k-1)} - \left(\sum_{j=1}^n a_{ij}x_j^{(k-1)} \right) / a_{ii} + b_i/a_{ii} \\ &= \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij}x_j^{(k-1)} \right) / a_{ii}, \end{aligned}$$

which is exactly the formula used in the algorithm.

The Gauss-Seidel iteration algorithm can be derived by taking Q as the lower triangle of A , i.e.,

$$Q_{ij} = \begin{cases} A_{ij}, & i \geq j \\ 0, & \text{otherwise.} \end{cases}$$

Thus (15) can be written as

$$\begin{bmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ \vdots & & \ddots & \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} x^{(k)} = - \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ & 0 & & \vdots \\ & & \ddots & a_{n-1,n} \\ & & & 0 \end{bmatrix} x^{(k-1)} + b.$$

If we solve this for $x_i^{(k)}$, $i = 1, \dots, n$, we get

$$\begin{aligned} x_1^{(k)} &= \left(-\sum_{j=2}^n a_{1j}x_j^{(k-1)} + b_1 \right) / a_{11} \\ x_2^{(k)} &= \left(-\sum_{j=3}^n a_{2j}x_j^{(k-1)} + b_2 - a_{21}x_1^{(k)} \right) / a_{22} \\ &\vdots \\ x_i^{(k)} &= \left(-\sum_{j=i+1}^n a_{ij}x_j^{(k-1)} + b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} \right) / a_{ii} \\ &\vdots \\ x_n^{(k)} &= \left(b_n - \sum_{j=1}^{n-1} a_{nj}x_j^{(k)} \right) / a_{nn}. \end{aligned}$$

This, however, is equivalent to

$$x_i \leftarrow \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij}x_j \right) / a_{ii},$$

which is the formula from the algorithm.

We now turn to a discussion of the properties required of a general iteration matrix G (or splitting matrix Q) so that the fixed point iteration (14) is guaranteed to converge.

Theorem 4.14 *If $\|G\| = \|I - Q^{-1}A\| < 1$ then (14) converges to a solution of $Ax = b$ for any starting vector $x^{(0)}$.*

Proof: It is clear that (14) is a fixed point iteration ($x = F(x)$), and the fixed point is a solution of $Ax = b$. This follows immediately from the equivalence transformations involving (15) and (16).

Now, let

$$e^{(k)} = x^{(k)} - x,$$

where x is the fixed point solution. Then


$$\begin{aligned} e^{(k)} &= Gx^{(k-1)} + c - (Gx + c) \\ &= G(x^{(k-1)} - x) = Ge^{(k-1)}. \end{aligned}$$

Taking norms, we have

$$\|e^{(k)}\| = \|Ge^{(k-1)}\| \leq \|G\| \|e^{(k-1)}\|.$$

Proceeding recursively we get

$$\|e^{(k)}\| \leq \|G\|^k \|e^{(0)}\|,$$

so that $\|e^{(k)}\| \rightarrow 0$ if $\|G\| < 1$. Of course, this ensures $x^{(k)} \rightarrow x$, i.e., convergence of the method. 

Corollary 4.15 *If A is diagonally dominant, then Jacobi iteration converges for any $x^{(0)}$.*

Proof: By the definition of diagonal dominance (Definition 4.8 in Sect. 4.3) the entries of A satisfy

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|, \quad i = 1, \dots, n. \quad (17)$$

We need to show $\|I - Q^{-1}A\| < 1$. Since the statement of Theorem 4.14 holds for any matrix norm, we can take $\|\cdot\|_\infty$. Earlier we showed that, for Jacobi iteration,

$$Q^{-1}A = \begin{bmatrix} 1 & \frac{a_{12}}{a_{11}} & \dots & \frac{a_{1n}}{a_{11}} \\ \frac{a_{21}}{a_{22}} & 1 & \dots & \frac{a_{2n}}{a_{22}} \\ \vdots & & \ddots & \vdots \\ \frac{a_{n1}}{a_{nn}} & \frac{a_{n2}}{a_{nn}} & \dots & 1 \end{bmatrix}.$$

But then

$$\begin{aligned} \|I - Q^{-1}A\|_\infty &= \max_{1 \leq i \leq n} \sum_{j=1}^n \left| (I - Q^{-1}A)_{ij} \right| \\ &= \max_{1 \leq i \leq n} \sum_{\substack{j=1 \\ j \neq i}}^n \left| \frac{a_{ij}}{a_{ii}} \right| \\ &< 1 \end{aligned}$$

by the diagonal dominance of A (17). ♠

In order to be able to formulate a necessary and sufficient condition for the convergence of (14) we need to cite a lemma which follows from the Schur factorization Theorem 5.19 in Sect. 5.6.

Lemma 4.16 *Any $n \times n$ matrix A is similar to an upper triangular matrix U whose off-diagonal elements are arbitrarily small, i.e., for any $\epsilon > 0$ there exists a nonsingular $n \times n$ matrix S such that*

$$S^{-1}AS = U = D + T,$$

where $D = \text{diag}(U)$ (containing the eigenvalues of A) and $\|T\|_\infty < \epsilon$.

Proof: See textbook p. 214 (proof of Theorem 3). ♠

Remark: Lemma 4.16 says that any $n \times n$ matrix A is *almost* similar to a diagonal matrix. This is what we saw at work when computing eigenvalues via the QR iteration algorithm in Sect. 5.7.3.

The next ingredient requires that we introduce the spectral radius of A , denoted by $\rho(A) = \max_{1 \leq i \leq n} \{|\lambda_i| : \lambda_i \text{ an eigenvalue of } A\}$.

Theorem 4.17 *The spectral radius of A is the largest lower bound for any matrix norm induced by a vector norm, i.e.,*

$$\rho(A) = \inf_{\|\cdot\|} \|A\|.$$

Proof: First we show $\rho(A) \leq \inf_{\|\cdot\|} \|A\|$. Let (λ, x) be the eigenpair of A for which $\rho(A) = |\lambda|$. Then, for any vector norm,

$$\|A\| = \sup_{\|u\|=1} \|Au\| \geq \frac{\|Ax\|}{\|x\|} = \frac{\|\lambda x\|}{\|x\|} = |\lambda| = \rho(A).$$

Taking the infimum over all induced matrix norms we get $\inf_{\|\cdot\|} \|A\| \geq \rho(A)$.

Next we show that $\rho(A) \geq \inf_{\|\cdot\|} \|A\|$. Applying the maximum norm to the factorization of Lemma 4.16 we have

$$\|S^{-1}AS\|_{\infty} = \|D + T\|_{\infty} \leq \|D\|_{\infty} + \|T\|_{\infty}.$$

Now, Lemma 4.16 ensures that for any $\epsilon > 0$ we have $\|T\|_{\infty} < \epsilon$. Moreover, $D = \text{diag}(\lambda_1, \dots, \lambda_n)$ so that

$$\|D\|_{\infty} = \rho(A).$$

Therefore,

$$\|S^{-1}AS\|_{\infty} \leq \rho(A) + \epsilon.$$

From the homework problem 4.6#6 we know that there exists another induced norm $\|\cdot\|'_{\infty}$ such that

$$\|A\|'_{\infty} = \|S^{-1}AS\|_{\infty}.$$

Therefore,

$$\|A\|'_{\infty} \leq \rho(A) + \epsilon.$$

Finally, we can take the infimum and obtain

$$\inf_{\|\cdot\|} \|A\| \leq \rho(A) + \epsilon.$$

Since this statement holds for arbitrary ϵ we have

$$\inf_{\|\cdot\|} \|A\| \leq \rho(A)$$

and are done. ♠

Now we can prove

Theorem 4.18 *The iteration $x^{(k)} = Gx^{(k-1)} + c$ (cf. (14)) converges to the solution of $Ax = b$ for any initial guess $x^{(0)}$ if and only if $\rho(G) < 1$.*

Proof: First we assume $\rho(G) < 1$ and show convergence. Theorem 4.17 tells us that there exists a matrix norm for which $\|G\| < 1$, and then Theorem 4.14 ensures convergence.

To prove the reverse implication we assume $\rho(A) > 1$ and deduce divergence of the fixed point iteration. Let (λ, u) be an eigenpair of G with $|\lambda| \geq 1$. Then we choose $x^{(0)}$ as

$$x^{(0)} = u + x,$$

where x is the fixed point of (14). Now, using (14) we have

$$x^{(k)} - x = G(x^{(k-1)} - x).$$

By applying this idea recursively we obtain

$$x^{(k)} - x = G^k(x^{(0)} - x) = G^k u = \lambda^k u.$$

By the choice of (λ, u) we see that $x^{(k)}$ diverges. ♠

A sufficient convergence criterion for Gauss-Seidel iteration can now also be formulated.

Theorem 4.19 *If A is symmetric and positive definite then Gauss-Seidel iteration converges for any initial vector $x^{(0)}$.*

Remark: The statement of the theorem can be relaxed. All that is needed is that A is symmetric positive semi-definite with positive diagonal entries, and such that a solution of $Ax = b$ exists. This slightly weaker version ensures that the normal equations can be solved using Gauss-Seidel iteration.

Proof of Theorem 4.19: According to Theorem 4.18 we need to show $\rho(G) < 1$, where, for Gauss-Seidel iteration, we have

$$G = I - Q^{-1}A = I - (L + D)^{-1}A. \quad (18)$$

Here we have decomposed A into

$$A = \underbrace{L + D}_{=Q} + \underbrace{L^T}_{=A-Q},$$

where L is the lower triangular part of A *without* the diagonal.

We begin by defining a vector norm

$$\|x\|_A = \sqrt{x^T A x}$$

which induces a matrix norm

$$\|G\|_A = \max_{x \neq 0} \frac{\|Gx\|_A}{\|x\|_A}.$$

This allows us to make use of Theorem 4.17, and we can instead show

$$\frac{\|Gx\|_A^2}{\|x\|_A^2} = \frac{(Gx)^T A Gx}{x^T A x} < 1, \quad (19)$$

since then

$$\rho(G) \leq \|G\|_A < 1.$$

Now, (19) is equivalent to

$$x^T (A - G^T A G) x > 0. \quad (20)$$

Using the definition of G (18) some messy manipulations yield

$$\begin{aligned} A - G^T A G &= A(L^T + D)^{-1} D (L + D)^{-1} A \\ &= A(L^T + D)^{-1} \sqrt{D} \sqrt{D} (L + D)^{-1} A, \end{aligned}$$

where the positive definiteness of A ($d_i > 0$) permits the last step. Returning to (20), and using the symmetry of A , we have

$$\begin{aligned} x^T(A - G^T AG)x &= \underbrace{x^T A(L^T + D)^{-1} \sqrt{D}}_{=y^T} \underbrace{\sqrt{D}(L + D)^{-1} Ax}_{=y} \\ &= y^T y \geq 0. \end{aligned}$$

In fact, this inequality is strict for $x \neq 0$ since $\sqrt{D}(L + D)^{-1}A$ has full rank. ♠

Rate of Convergence

We want to have an estimate for how many iterations are required in the fixed point iteration

$$x^{(k)} = Gx^{(k-1)} + c$$

to reduce the initial error $e^{(0)} = x^{(0)} - x$ by a factor of $10^{-\alpha}$ for some positive exponent α .

We know (see the proof of Theorem 4.14) that the fixed point iteration scheme leads to an error estimate of the form

$$\|e^{(k)}\| \leq \|G\|^k \|e^{(0)}\|.$$

Now, using Theorem 4.17 we can conclude

$$\|e^{(k)}\| \leq [\rho(G)]^k \|e^{(0)}\|.$$

Our goal is to reduce the initial error by a factor $10^{-\alpha}$. Therefore, we want

$$\|e^{(k)}\| \leq [\rho(G)]^k \|e^{(0)}\| \leq 10^{-\alpha} \|e^{(0)}\|$$

or

$$[\rho(G)]^k \leq 10^{-\alpha}.$$

Thus

$$k \ln(\rho(G)) \leq -\alpha \ln 10.$$

Since, for a convergent fixed point iteration we have $0 < \rho(G) < 1$ we get

$$k \geq \frac{\alpha \ln 10}{-\ln(\rho(G))}.$$

If we define the *rate of convergence*

$$r = \frac{-\ln(\rho(G))}{\ln 10} = \frac{1}{\log_{10}(\rho(G))},$$

then we should have

$$k \geq \frac{\alpha}{r}$$

to ensure the desired improvement.

Successive Over Relaxation (SOR)

Another iterative solver can be obtained by computing the iterate $x^{(k)}$ as a combination of $x^{(k)}$ and $x^{(k-1)}$, i.e.,

$$x^{(k)} \leftarrow (1 - \omega)x^{(k-1)} + \omega x^{(k)},$$

with the *relaxation parameter* ω chosen appropriately. This is a generalization of the Gauss-Seidel iteration (corresponding to $\omega = 1$). The standard SOR algorithm is as follows (a more general version is discussed in the textbook):

Algorithm (SOR iteration)

Input A, b, x, ω, n, M

for $k = 1$ to M do

for $i = 1$ to n do

$$u_i = (1 - \omega)x_i + \omega \left(b_i - \sum_{j=1}^{i-1} a_{ij}u_j - \sum_{j=i+1}^n a_{ij}x_j \right) / a_{ii}$$

end

for $i = 1$ to n do

$$x_i = u_i$$

end

end

Output x

Theorem 4.20 *If A is symmetric and positive definite and $0 < \omega < 2$ then SOR iteration converges for any starting value $x^{(0)}$.*

Proof: omitted ♠

Remarks:

1. The optimal value of ω is known only for special cases. For example, if A is tridiagonal, then

$$\omega_{\text{opt}} = \frac{2}{1 + \sqrt{1 - \rho(G)}},$$

where G is the iteration matrix for the Gauss-Seidel iteration.

2. For (sparse) matrices with special structure the basic iteration algorithms can be implemented much more efficiently. For example, for the Poisson problem discussed at the beginning of the semester Jacobi iteration becomes

```

for  $k = 1$  to  $M$  do
  for  $i = 1$  to  $n - 1$  do
    for  $j = 1$  to  $n - 1$  do
       $u_{ij}^{(k)} = \left( u_{i-1,j}^{(k-1)} + u_{i,j-1}^{(k-1)} + u_{i+1,j}^{(k-1)} + u_{i,j+1}^{(k-1)} + \frac{f_{ij}}{n^2} \right) / 4$ 
    end
  end
end
end

```

Note that even though the system matrix A is of size $n^2 \times n^2$ only five entries per row are non-zero and they are hard-coded into the algorithm. Thus, the matrix A is never explicitly and completely formed. In Matlab the i - j double loops can be implemented in one single line of code.

3. A few other classical iterative methods such as symmetric successive over relaxation (SSOR) and Chebyshev iteration are discussed in the textbook.
4. Due to their rather slow convergence classical iterative solvers are no longer very popular as solvers of linear systems. However, they are frequently used as preconditioners for more sophisticated methods (e.g., conjugate gradient or multigrid).

4.7 Steepest Descent and Conjugate Gradient Methods

Even though the SOR method may converge faster than Jacobi or Gauss-Seidel iteration, it is difficult to choose a good relaxation parameter ω . Therefore, most of the time another class of iterative solvers is used for linear systems with symmetric positive definite system matrices. These methods are based on

Theorem 4.21 *If A is symmetric positive definite, then solving $Ax = b$ is equivalent to minimizing the quadratic form*

$$q(x) = \langle x, Ax \rangle - 2\langle x, b \rangle.$$

Remarks:

1. Here we have used the notation $\langle x, y \rangle = x^T y = \sum_{i=1}^n x_i y_i$ for the standard inner product in \mathbb{R}^n .
2. The family of optimization based iterative solvers is also known as *Krylov subspace iteration* in the literature.

Proof: First we consider only changes of q along the ray $x + tv$, where $t \in \mathbb{R}$, and $v \neq 0$ a fixed direction. By definition

$$\begin{aligned} q(x + tv) &= \langle x + tv, A(x + tv) \rangle - 2\langle x + tv, b \rangle \\ &= \langle x, Ax \rangle + \langle tv, Ax \rangle + \langle x, A(tv) \rangle + \langle tv, A(tv) \rangle - 2\langle x, b \rangle - 2\langle tv, b \rangle \\ &= \underbrace{\langle x, Ax \rangle - 2\langle x, b \rangle}_{=q(x)} + \langle tv, Ax \rangle + \langle x, A(tv) \rangle + \langle tv, A(tv) \rangle - 2\langle tv, b \rangle \end{aligned} \quad (21)$$

Now, using the symmetry of A ,

$$\langle x, A(tv) \rangle = tx^T(Av) = t(v^T A^T x)^T = t(v^T Ax)^T = tv^T Ax,$$

where the last equality holds because the inner product is a scalar quantity. Therefore,

$$\langle x, A(tv) \rangle = t\langle v, Ax \rangle. \quad (22)$$

Inserting (22) into (21) we get

$$\begin{aligned} q(x + tv) &= q(x) + 2t\langle v, Ax \rangle + t^2\langle v, Av \rangle - 2t\langle v, b \rangle \\ &= q(x) + 2t\langle v, Ax - b \rangle + t^2\langle v, Av \rangle. \end{aligned} \quad (23)$$

Note that here $\langle v, Av \rangle > 0$ since A is positive definite and $v \neq 0$. Therefore, as a quadratic function of t (with positive leading coefficient) $q(x + \cdot v)$ has a minimum.

A necessary condition for the minimum (along the ray $x + tv$) is

$$\frac{d}{dt}q(x + tv) = 0.$$

Thus, we calculate

$$\frac{d}{dt}q(x + tv) = 2\langle v, Ax - b \rangle + 2t\langle v, Av \rangle. \quad (24)$$

Clearly, (24) is zero for

$$\hat{t} = \frac{-\langle v, Ax - b \rangle}{\langle v, Av \rangle} = \frac{\langle v, b - Ax \rangle}{\langle v, Av \rangle}.$$

The value of the minimum is obtained from (23):

$$\begin{aligned} q(x + \hat{t}v) &= q(x) + 2\frac{\langle v, b - Ax \rangle}{\langle v, Av \rangle}\langle v, Ax - b \rangle + \left(\frac{\langle v, b - Ax \rangle}{\langle v, Av \rangle}\right)^2 \langle v, Av \rangle \\ &= q(x) - 2\frac{\langle v, b - Ax \rangle^2}{\langle v, Av \rangle} + \frac{\langle v, b - Ax \rangle^2}{\langle v, Av \rangle} \\ &= q(x) - \frac{\langle v, b - Ax \rangle^2}{\langle v, Av \rangle}. \end{aligned}$$

Now, $q(x + \hat{t}v) < q(x)$ if and only if $\langle v, b - Ax \rangle \neq 0$, i.e., the direction v is not orthogonal to the residual. Thus the value of q can be reduced by moving from x to $x + tv$ along the direction v , where v is some direction not orthogonal to the residual $b - Ax$.

To conclude the proof we have two possibilities to consider:

1. x is a solution of $Ax = b$, i.e., $b - Ax = 0$. Then $q(x)$ is actually the minimum value, and no choice of v can reduce the value of q (i.e., “ x solution of $Ax = b$ ” \implies “ x minimizer of q ”).
2. x is such that $Ax \neq b$, then there exists a direction v such that $\langle v, b - Ax \rangle \neq 0$ and so $q(x)$ is not a minimum (and thus can be improved – which we do until we have reached the minimum and case 1.). So, “ x a minimizer of q ” \implies “ x a solution of $Ax = b$ ”.

The claim is established. ♠

The argument used at the end of the proof suggests an iterative algorithm for the solution of $Ax = b$ via minimization of q .

“Algorithm”

Input A, b , initial guess $x^{(0)}$, initial search direction $v^{(0)}$

for $k = 0, 1, \dots$ do

$$\text{Calculate the “stepsize” } t = \frac{\langle v^{(k)}, b - Ax^{(k)} \rangle}{\langle v^{(k)}, Av^{(k)} \rangle}$$

$$\text{Update the solution } x^{(k+1)} = x^{(k)} + tv^{(k)}$$

Pick a new search direction $v^{(k+1)}$

Output $x^{(k+1)}$

Remark: The preceding algorithm is in quotation marks since it is not yet a complete algorithm. We have not yet answered how to choose the search directions $v^{(k+1)}$. The only constraint we have so far is that $\langle v, b - Ax \rangle \neq 0$.

Method of Steepest Descent

From Calculus we know that the gradient of q indicates the direction of greatest change. Therefore, in the so-called *method of steepest descent* we choose

$$v^{(k+1)} = -\nabla q(x^{(k+1)}).$$

In homework problem 4.7#1 you will show that

$$\nabla q(x^{(k+1)}) = 2(Ax^{(k+1)} - b),$$

and so we get

Algorithm (Steepest descent)

Input A, b , initial guess x

for $k = 0, 1, \dots$ do

$$v = b - Ax$$

$$t = \frac{\langle v, v \rangle}{\langle v, Av \rangle} \quad (\text{see the rough “algorithm” above})$$

$$x = x + tv$$

Output x

Remark: This method is not very practical, as it usually converges rather slowly. Since the level curves of q are ellipses (for problems with two unknowns), the search directions tend to zig-zag down the surface instead of going straight to the bottom of the valley (especially for elongated ellipses, i.e., $\kappa_2(A)$ large). See Figure 4.3 in the textbook.

Conjugate Search Directions

The main ingredient in deriving another one of the “top ten algorithms of the 20th century” (the *conjugate gradient method*) will be the use of so-called *conjugate search directions*. To this end we introduce

Definition 4.22 *Let A be a symmetric positive definite matrix. Then the inner product*

$$\langle x, y \rangle_A = \langle x, Ay \rangle$$

is called an A -inner product.

The A -inner product induces a vector norm

$$\|x\|_A = \sqrt{\langle x, x \rangle_A} = \sqrt{\langle x, Ax \rangle}.$$

We call a set of vectors $\{u^{(1)}, \dots, u^{(n)}\}$ *A -orthonormal* if

$$\langle u^{(i)}, u^{(j)} \rangle_A = \delta_{ij}.$$

The main theorem (which will give us the conjugate gradient method) is

Theorem 4.23 *Let $\{u^{(1)}, \dots, u^{(n)}\}$ be an A -orthonormal set, and define*

$$x^{(i)} = x^{(i-1)} + \langle b - Ax^{(i-1)}, u^{(i)} \rangle u^{(i)}, \quad i = 1, \dots, n, \quad (25)$$

with $x^{(0)}$ an arbitrary starting vector. Then

$$Ax^{(n)} = b.$$

Remarks:

1. Note that Theorem 4.23 says that the iteration (25) yields the solution of $Ax = b$ in n steps, and thus can be considered a *direct method*. However, in practice this usually does not happen (due to roundoff errors). Historically, this means that – after its introduction in 1952 by Hestenes and Stiefel – the conjugate gradient method was considered a useless (direct) method, and long forgotten.
2. In the 1970s it was observed that for certain (large, sparse, and well-conditioned) problems the conjugate gradient method converges extremely fast to an accurate approximate solution of $Ax = b$ (much faster than in n steps). And thus, it has become one of the top iterative solvers. In fact, one can show that, for well-conditioned problems, the conjugate gradient method converges in $\mathcal{O}(\sqrt{\kappa})$ iterations.

Proof of Theorem 4.23: From (25) we see that we can define the stepsize as

$$t_i = \langle b - Ax^{(i-1)}, u^{(i)} \rangle. \quad (26)$$

Then (25) becomes

$$x^{(i)} = x^{(i-1)} + t_i u^{(i)}.$$

We multiply this by A and get

$$Ax^{(i)} = Ax^{(i-1)} + t_i Au^{(i)}. \quad (27)$$

Therefore, proceeding recursively,

$$\begin{aligned} Ax^{(n)} &= Ax^{(n-1)} + t_n Au^{(n)} \\ &= Ax^{(n-2)} + t_{n-1} Au^{(n-1)} + t_n Au^{(n)} \\ &\vdots \\ &= Ax^{(0)} + t_1 Au^{(1)} + t_2 Au^{(2)} + \dots + t_n Au^{(n)}. \end{aligned}$$

Now we subtract b and take the inner product with $u^{(i)}$ (which allows us to use the A -orthonormality of $\{u^{(1)}, \dots, u^{(n)}\}$) to arrive at

$$\begin{aligned} \langle Ax^{(n)} - b, u^{(i)} \rangle &= \langle Ax^{(0)} - b, u^{(i)} \rangle + \sum_{j=1}^n t_j \underbrace{\langle Au^{(j)}, u^{(i)} \rangle}_{=\delta_{ij}} \\ &= \langle Ax^{(0)} - b, u^{(i)} \rangle + t_i. \end{aligned}$$

If we can show that

$$t_i = -\langle Ax^{(0)} - b, u^{(i)} \rangle, \quad (28)$$

then $\langle Ax^{(n)} - b, u^{(i)} \rangle = 0$, and so $Ax^{(n)} - b$ is orthogonal to all $u^{(i)}$, $i = 1, \dots, n$.

In order to see the implications of this statement we observe below that $\{u^{(1)}, \dots, u^{(n)}\}$ form a basis for \mathbb{R}^n , and thus $Ax^{(n)} - b = 0$ (which proves the statement of the theorem).

From the A -orthonormality of $\{u^{(1)}, \dots, u^{(n)}\}$ we know

$$\langle Au^{(j)}, u^{(i)} \rangle = \delta_{ij},$$

so that

$$U^T AU = I.$$

This implies that both A and U are nonsingular, and thus the columns $u^{(i)}$, $i = 1, \dots, n$, of U form a basis of \mathbb{R}^n .

It now remains to establish (28). From (26) we have

$$\begin{aligned} t_i &= \langle b - Ax^{(i-1)}, u^{(i)} \rangle \\ &= \langle b - Ax^{(0)} + Ax^{(0)} - Ax^{(1)} + Ax^{(1)} - \dots - Ax^{(i-2)} + Ax^{(i-2)} - Ax^{(i-1)}, u^{(i)} \rangle \\ &= \langle b - Ax^{(0)}, u^{(i)} \rangle + \underbrace{\langle Ax^{(0)} - Ax^{(1)}, u^{(i)} \rangle}_{=-t_1 Au^{(1)}} + \dots + \underbrace{\langle Ax^{(i-2)} - Ax^{(i-1)}, u^{(i)} \rangle}_{=-t_{i-1} Au^{(i-1)}}, \end{aligned}$$

where we have used (27). Thus,

$$t_i = \langle b - Ax^{(0)}, u^{(i)} \rangle - t_1 \underbrace{\langle Au^{(1)}, u^{(i)} \rangle}_{=0} - \dots - t_{i-1} \underbrace{\langle Au^{(i-1)}, u^{(i)} \rangle}_{=0}$$

$$= -\langle Ax^{(0)} - b, u^{(i)} \rangle,$$

which establishes (28). ♠

The main question now is

How to find $\{u^{(1)}, \dots, u^{(n)}\}$?

We can use any basis of \mathbb{R}^n , and then obtain $\{u^{(1)}, \dots, u^{(n)}\}$ by applying the Gram-Schmidt algorithm (with respect to the A -inner product) to this set. Therefore, we start with the canonical basis $\{e^{(1)}, \dots, e^{(n)}\}$. Then

$$u^{(1)} = \frac{e^{(1)}}{\|e^{(1)}\|_A} = \frac{e^{(1)}}{\sqrt{\langle e^{(1)}, Ae^{(1)} \rangle}},$$

and the general step is

$$u^{(i)} = \frac{v^{(i)}}{\|v^{(i)}\|_A},$$

where

$$v^{(i)} = e^{(i)} - \sum_{j=1}^{i-1} \langle e^{(i)}, u^{(j)} \rangle_A u^{(j)}.$$

Since $\langle e^{(i)}, u^{(j)} \rangle_A = \langle e^{(i)}, Au^{(j)} \rangle = (Au^{(j)})_i$ we have

$$u^{(1)} = \frac{e^{(1)}}{\sqrt{a_{11}}}$$

and

$$v^{(i)} = e^{(i)} - \sum_{j=1}^{i-1} (Au^{(j)})_i u^{(j)}.$$

Before stating the conjugate gradient algorithm we reformulate Theorem 4.23 for an A -orthogonal (instead of A -orthonormal) set.

Theorem 4.24 *Let $\{v^{(1)}, \dots, v^{(n)}\}$ be an A -orthogonal set, and define*

$$t_i = \frac{\langle b - Ax^{(i-1)}, v^{(i)} \rangle}{\|v^{(i)}\|_A}.$$

Then, for any starting vector $x^{(0)}$, the iteration

$$x^{(i)} = x^{(i-1)} + t_i v^{(i)}, \quad i = 1, \dots, n,$$

yields

$$Ax^{(n)} = b.$$

Proof: Obvious. ♠

Remark: Here the search directions $v^{(i)}$ are fixed before the algorithm is started. It is better to determine $v^{(i)}$ when needed during the algorithm by making sure that at each step the components in the previously used directions are removed. This is done in

Algorithm (Conjugate gradient method)

- (1) Input $A, b, x^{(0)}, M, \epsilon$
- (2) $r^{(0)} = b - Ax^{(0)}$
- (3) $v^{(0)} = r^{(0)}$
- (4) Output $0, x^{(0)}, r^{(0)}$
- (5) for $k = 0$ to $M - 1$ do
- (6) if $v^{(k)} = 0$ then stop
- (7) $t_k = \frac{\|r^{(k)}\|_2^2}{\|v^{(k)}\|_A^2}$
- (8) $x^{(k+1)} = x^{(k)} + t_k v^{(k)}$
- (9) $r^{(k+1)} = r^{(k)} - t_k A v^{(k)}$
- (10) if $\|r^{(k+1)}\|^2 < \epsilon$ then stop
- (11) $s_k = \frac{\|r^{(k+1)}\|_2^2}{\|r^{(k)}\|_2^2}$
- (12) $v^{(k+1)} = r^{(k+1)} + s_k v^{(k)}$
- (13) Output $k + 1, x^{(k+1)}, r^{(k+1)}$
- (14) end

Remarks:

1. A “computer version” (with lots of overwriting) is listed in the textbook on page 238.
2. The conjugate gradient algorithm is very efficient since the bulk of work for each iteration is the one matrix-vector multiplication $Av^{(k)}$ needed for the update of the residual. It is essential that this operation is customized for the matrix A . For many sparse matrices this can be done in $\mathcal{O}(n)$ operations, and thus the overall operations count for the conjugate gradient method for many problems is also $\mathcal{O}(n)$.
3. The Matlab script `CGDemo.m` illustrates the convergence behavior of the conjugate gradient algorithm. In this example the CG algorithm is applied to a 500×500 symmetric positive definite matrix. The parameter τ affects the number of nonzero entries as well as the 2-norm condition number of A . This is done by replacing all off-diagonal entries of A with $|a_{ij}| > \tau$ by zero, i.e., for small τ A is very sparse and well-conditioned, whereas for large τ both the number of non-zero entries as well as the condition number increase. We can see that for the ill-conditioned case no convergence at all is achieved in the allowed 20 iterations, whereas for the well-conditioned matrix the CG algorithm converges in 8 iterations to a solution with relative residual accuracy of 10^{-16} .

We now need to show that this algorithm agrees with the theory developed so far. Thus, we need to show $\{v^{(0)}, v^{(1)}, \dots, v^{(n-1)}\}$ is A -orthogonal (see c) in Theorem 4.25). Moreover, we will show that the residuals $\{r^{(0)}, r^{(1)}, \dots, r^{(n-1)}\}$ are orthogonal (cf. e)), and that they can be computed recursively as formulated in the algorithm (follows from d) below).

Theorem 4.25 *If we use the conjugate gradient algorithm to solve the linear system $Ax = b$, where A is an $n \times n$ symmetric positive matrix, then for any $m < n$*

- a) $\langle r^{(m)}, v^{(i)} \rangle = 0, \quad (0 \leq i < m),$
- b) $\langle r^{(i)}, r^{(i)} \rangle = \langle r^{(i)}, v^{(i)} \rangle, \quad (0 \leq i \leq m),$
- c) $\langle v^{(m)}, Av^{(i)} \rangle = 0, \quad (0 \leq i < m),$
- d) $r^{(i)} = b - Ax^{(i)}, \quad (0 \leq i \leq m),$
- e) $\langle r^{(m)}, r^{(i)} \rangle = 0, \quad (0 \leq i < m),$
- f) $r^{(i)} \neq 0, \quad (0 \leq i \leq m).$

Remark: Items a) and b) are not directly related to the claims formulated above, but are required to prove the other statements.

Proof: We use induction on m . The beginning, $m = 0$ is trivial. Now we assume that the statements a) – f) hold for m and prove they are true for $m + 1$.

a) Show $\langle r^{(m+1)}, v^{(i)} \rangle = 0, \quad (0 \leq i \leq m)$. First, we treat the case $i = m$. Using the recursive definition of the residuals from line (9) of the CG algorithm we have

$$\begin{aligned} \langle r^{(m+1)}, v^{(m)} \rangle &= \langle r^{(m)} - t_m Av^{(m)}, v^{(m)} \rangle \\ &= \langle r^{(m)}, v^{(m)} \rangle - t_m \langle Av^{(m)}, v^{(m)} \rangle. \end{aligned}$$

The definition of the stepsize, line (7), yields

$$\begin{aligned} \langle r^{(m+1)}, v^{(m)} \rangle &= \langle r^{(m)}, v^{(m)} \rangle - \frac{\|r^{(m)}\|_2^2}{\|v^{(m)}\|_A^2} \langle Av^{(m)}, v^{(m)} \rangle \\ &= \langle r^{(m)}, v^{(m)} \rangle - \langle r^{(m)}, r^{(m)} \rangle \\ &= 0, \end{aligned}$$

where the last simplification follows from the induction hypothesis for statement b).

Now we discuss the case $i < m$. From line (9) of the algorithm we get

$$\begin{aligned} \langle r^{(m+1)}, v^{(i)} \rangle &= \langle r^{(m)} - t_m Av^{(m)}, v^{(i)} \rangle \\ &= \underbrace{\langle r^{(m)}, v^{(i)} \rangle}_{=0} - t_m \underbrace{\langle Av^{(m)}, v^{(i)} \rangle}_{=0} \\ &= 0, \end{aligned}$$

where we have used the induction hypotheses for statements a) and c), respectively.

b) We show $\langle r^{(m+1)}, r^{(m+1)} \rangle = \langle r^{(m+1)}, v^{(m+1)} \rangle$. Using line (12) of the algorithm

$$\langle r^{(m+1)}, v^{(m+1)} \rangle = \langle r^{(m+1)}, r^{(m+1)} + s_m v^{(m)} \rangle$$

$$= \langle r^{(m+1)}, r^{(m+1)} \rangle + s_m \underbrace{\langle r^{(m+1)}, v^{(m)} \rangle}_{=0}.$$

Here we have made use of the statement just proved in a) above.

c) We need to show $\langle v^{(m+1)}, Av^{(i)} \rangle = 0$, for $0 \leq i \leq m$. First we discuss the case $0 \leq i < m$. Line (12) of the algorithm yields

$$\begin{aligned} \langle v^{(m+1)}, Av^{(i)} \rangle &= \langle r^{(m+1)} + s_m v^{(m)}, Av^{(i)} \rangle \\ &= \langle r^{(m+1)}, Av^{(i)} \rangle + s_m \langle v^{(m)}, Av^{(i)} \rangle. \end{aligned}$$

Next, line (9) of the algorithm implies

$$Av^{(k)} = \frac{1}{t_k} (r^{(k)} - r^{(k+1)}).$$

Using this formula on the first term of the right-hand side above we have

$$\begin{aligned} \langle v^{(m+1)}, Av^{(i)} \rangle &= \frac{1}{t_i} \langle r^{(m+1)}, r^{(i)} - r^{(i+1)} \rangle + s_m \langle v^{(m)}, Av^{(i)} \rangle \\ &= \frac{1}{t_i} \langle r^{(m+1)}, v^{(i)} - s_{i-1} v^{(i-1)} - v^{(i+1)} + s_i v^{(i)} \rangle + s_m \langle v^{(m)}, Av^{(i)} \rangle, \end{aligned}$$

where we have used line (12) (together with the understanding that $s_{-1} = 0$, $v^{(-1)} = 0$) to replace $r^{(i)}$ and $r^{(i+1)}$. We now expand the right-hand side, and get

$$\begin{aligned} \langle v^{(m+1)}, Av^{(i)} \rangle &= \frac{1}{t_i} \left(\underbrace{\langle r^{(m+1)}, v^{(i)} \rangle}_{\stackrel{a)}{=} 0} - s_{i-1} \underbrace{\langle r^{(m+1)}, v^{(i-1)} \rangle}_{\stackrel{a)}{=} 0} - \underbrace{\langle r^{(m+1)}, v^{(i+1)} \rangle}_{\stackrel{a)}{=} 0 \text{ for } i < m} \right. \\ &= \left. + s_i \underbrace{\langle r^{(m+1)}, v^{(i)} \rangle}_{\stackrel{a)}{=} 0} \right) + s_m \underbrace{\langle v^{(m)}, Av^{(i)} \rangle}_{=0} \\ &= 0. \end{aligned}$$

Here we have used the result already proven in a) as well as the induction hypothesis for c). Note that the third and the fifth terms above are the reason why we have to now discuss the case $i = m$ separately.

We can proceed as above, but in the end are left with

$$\langle v^{(m+1)}, Av^{(m)} \rangle = -\frac{1}{t_m} \langle r^{(m+1)}, v^{(m+1)} \rangle + s_m \langle v^{(m)}, Av^{(m)} \rangle.$$

Now, replacing the stepsizes t_m and s_m using lines (7) and (11), respectively, we get

$$\begin{aligned} \langle v^{(m+1)}, Av^{(m)} \rangle &= -\frac{\|v^{(m)}\|_A^2}{\|r^{(m)}\|_2^2} \langle r^{(m+1)}, v^{(m+1)} \rangle + \frac{\|r^{(m+1)}\|_2^2}{\|r^{(m)}\|_2^2} \langle v^{(m)}, Av^{(m)} \rangle \\ &= -\frac{\|v^{(m)}\|_A^2}{\|r^{(m)}\|_2^2} \langle r^{(m+1)}, r^{(m+1)} \rangle + \frac{\|r^{(m+1)}\|_2^2}{\|r^{(m)}\|_2^2} \|v^{(m)}\|_A^2 \\ &= 0. \end{aligned}$$

where, at the end, we have made use of statement b) proved above as well as the definition of the A -norm of a vector.

d) We need to show $r^{(m+1)} = b - Ax^{(m+1)}$. From line (8) of the algorithm we have

$$\begin{aligned} b - Ax^{(m+1)} &= b - A(x^{(m)} + t_m v^{(m)}) \\ &= b - Ax^{(m)} - t_m Av^{(m)} \\ &= r^{(m)} - t_m Av^{(m)} \end{aligned}$$

by virtue of the induction hypothesis for d). Using line (9) of the algorithm we can replace $t_m Av^{(m)}$ by a difference of residuals and have

$$b - Ax^{(m+1)} = r^{(m)} - (r^{(m)} - r^{(m+1)}) = r^{(m+1)}.$$

e) We show $\langle r^{(m+1)}, r^{(i)} \rangle = 0$ for $0 \leq i \leq m$. From line (12) of the algorithm (with $s_{-1} = v^{(-1)} = 0$) we get

$$\begin{aligned} \langle r^{(m+1)}, r^{(i)} \rangle &= \langle r^{(m+1)}, v^{(i)} - s_{i-1} v^{(i-1)} \rangle \\ &= \underbrace{\langle r^{(m+1)}, v^{(i)} \rangle}_{\stackrel{a)}{=} 0} - s_{i-1} \underbrace{\langle r^{(m+1)}, v^{(i-1)} \rangle}_{\stackrel{a)}{=} 0} = 0. \end{aligned}$$

f) We need to show $r^{(m+1)} \neq 0$. Since A is positive definite, the quadratic form for $v^{(m+1)} \neq 0$ satisfies

$$\langle v^{(m+1)}, Av^{(m+1)} \rangle > 0.$$

On the other hand, using line (12) of the algorithm,

$$\begin{aligned} \langle v^{(m+1)}, Av^{(m+1)} \rangle &= \langle r^{(m+1)} + s_m v^{(m)}, Av^{(m+1)} \rangle \\ &= \langle r^{(m+1)}, Av^{(m+1)} \rangle + s_m \underbrace{\langle v^{(m)}, Av^{(m+1)} \rangle}_{\stackrel{c)}{=} 0}. \end{aligned}$$

Thus, $r^{(m+1)}$ cannot be zero (since A is positive definite and $v^{(m+1)} \neq 0$). ♠

Remark: The algorithms of this section were designed to minimize the quadratic form $q(x) = \langle x, Ax \rangle - 2\langle x, b \rangle$. We will now show they also minimize $\|e^{(k)}\|_A$.

Let $e^{(k)} = x - x^{(k)}$, where x is the true solution of the linear system $Ax = b$. Then

$$\begin{aligned} \|e^{(k)}\|_A^2 &= \langle e^{(k)}, Ae^{(k)} \rangle \\ &= \langle x - x^{(k)}, A(x - x^{(k)}) \rangle \\ &= \langle x^{(k)}, Ax^{(k)} \rangle - 2\underbrace{\langle x^{(k)}, Ax \rangle}_{=b} + \underbrace{\langle x, Ax \rangle}_{=b} \\ &= q(x^{(k)}) + \langle x, b \rangle. \end{aligned}$$

Now, since x and b are constant, we see that minimization of q is equivalent to minimization of the A -norm of the error.

Preconditioned CG

We mentioned earlier that the number of iterations required for the conjugate gradient algorithm to converge is proportional to $\sqrt{\kappa_2(A)}$. Thus, for poorly conditioned

matrices, convergence will be very slow. A commonly used trick is therefore to *precondition* the problem.

The basic idea can be described as follows. We want to transform the linear system $Ax = b$ into an equivalent system $\hat{A}\hat{x} = \hat{b}$, where $\kappa(\hat{A}) < \kappa(A)$. According to the comments above this should result in faster convergence.

How do we find \hat{A} , \hat{x} , and \hat{b} ? Consider a nonsingular $n \times n$ matrix S . We can rewrite

$$\begin{aligned} Ax = b &\iff S^T Ax = S^T b \\ &\iff \underbrace{S^T A S}_{=\hat{A}} \underbrace{S^{-1} x}_{=\hat{x}} = \underbrace{S^T b}_{=\hat{b}}. \end{aligned}$$

Our choice of S should be such that

$$SS^T = Q^{-1} \tag{29}$$

with Q a symmetric positive definite matrix called *splitting matrix* or *preconditioner*. This will preserve the symmetry and positive definiteness of the system.

Before discussing any specific preconditioners we consider how this framework affects the CG algorithm. Formally, the conjugate gradient algorithm for the problem $\hat{A}\hat{x} = \hat{b}$ is

- (1) Input \hat{A} , \hat{b} , $\hat{x}^{(0)}$, M , ϵ
- (2) $\hat{r}^{(0)} = \hat{b} - \hat{A}\hat{x}^{(0)}$
- (3) $\hat{v}^{(0)} = \hat{r}^{(0)}$
- (4) Output 0 , $\hat{x}^{(0)}$, $\hat{r}^{(0)}$
- (5) for $k = 0$ to $M - 1$ do
- (6) if $\hat{v}^{(k)} = 0$ then stop
- (7) $\hat{t}_k = \frac{\|\hat{r}^{(k)}\|_2^2}{\|\hat{v}^{(k)}\|_{\hat{A}}^2}$
- (8) $\hat{x}^{(k+1)} = \hat{x}^{(k)} + \hat{t}_k \hat{v}^{(k)}$
- (9) $\hat{r}^{(k+1)} = \hat{r}^{(k)} - \hat{t}_k \hat{A} \hat{v}^{(k)}$
- (10) if $\|\hat{r}^{(k+1)}\|^2 < \epsilon$ then stop
- (11) $\hat{s}_k = \frac{\|\hat{r}^{(k+1)}\|_2^2}{\|\hat{r}^{(k)}\|_2^2}$
- (12) $\hat{v}^{(k+1)} = \hat{r}^{(k+1)} + \hat{s}_k \hat{v}^{(k)}$
- (13) Output $k + 1$, $\hat{x}^{(k+1)}$, $\hat{r}^{(k+1)}$
- (14) end

Remark: The algorithm is made more efficient if the preconditioning operations are included directly in the algorithm.

We therefore make the following substitutions.

$$\begin{aligned}\hat{x}^{(k)} &= S^{-1}x^{(k)}, \\ \hat{r}^{(k)} &= \hat{b} - \hat{A}\hat{x}^{(k)} = S^T b(S^T A S)(S^{-1}x^{(k)}) = S^T b - S^T A x^{(k)} = S^T r^{(k)},\end{aligned}$$

and also define

$$\begin{aligned}\hat{v}^{(k)} &= S^{-1}v^{(k)}, \\ \hat{r}^{(k)} &= Q^{-1}r^{(k)}.\end{aligned}$$

Now we can consider how this transforms the algorithm above. Line (2) becomes

$$\begin{aligned}\hat{r}^{(0)} = \hat{b} - \hat{A}\hat{x}^{(0)} &\iff S^T r^{(0)} = S^T b - S^T A S S^{-1}x^{(0)} \\ &\iff r^{(0)} = b - A x^{(0)},\end{aligned}$$

where we have multiplied by S^{-T} in the last step. For line (3) we have

$$\begin{aligned}\hat{v}^{(0)} = \hat{r}^{(0)} &\iff S^{-1}v^{(0)} = S^T r^{(0)} \\ &\iff v^{(0)} = Q^{-1}r^{(0)},\end{aligned}$$

where we have multiplied by S and used the definition (29) of the preconditioner Q . Line (7) transforms as follows:

$$\begin{aligned}\hat{t}_k &= \|\hat{r}^{(k)}\|_2^2 / \|\hat{v}^{(k)}\|_{\hat{A}}^2 \\ &= \|S^T r^{(k)}\|_2^2 / \|S^{-1}v^{(k)}\|_{S^T A S}^2 \\ &= \langle S^T r^{(k)}, S^T r^{(k)} \rangle / \langle S^{-1}v^{(k)}, S^T A S S^{-1}v^{(k)} \rangle \\ &= \langle r^{(k)}, \underbrace{S S^T}_{=Q^{-1}} r^{(k)} \rangle / \langle v^{(k)}, A v^{(k)} \rangle \\ &= \langle r^{(k)}, \tilde{r}^{(k)} \rangle / \|v^{(k)}\|_A^2.\end{aligned}$$

Here we have made use of the easily established fact that for any $n \times n$ matrix B $\langle x, B y \rangle = \langle B^T x, y \rangle$. Line (8) becomes

$$\begin{aligned}\hat{x}^{(k+1)} = \hat{x}^{(k)} + \hat{t}_k \hat{v}^{(k)} &\iff S^{-1}x^{(k+1)} = S^{-1}x^{(k)} + \hat{t}_k S^{-1}v^{(k)} \\ &\iff x^{(k+1)} = x^{(k)} + \hat{t}_k v^{(k)},\end{aligned}$$

where we have multiplied by S in the last step. For line (9) we have

$$\begin{aligned}\hat{r}^{(k+1)} = \hat{r}^{(k)} - \hat{t}_k \hat{A} \hat{v}^{(k)} &\iff S^T r^{(k+1)} = S^T r^{(k)} - \hat{t}_k (S^T A S) S^{-1}v^{(k)} \\ &\iff r^{(k+1)} = r^{(k)} - \hat{t}_k A v^{(k)},\end{aligned}$$

where we have multiplied by S^{-T} . Line (11) transforms as follows:

$$\hat{s}_k = \|\hat{r}^{(k+1)}\|_2^2 / \|\hat{r}^{(k)}\|_2^2$$

$$\begin{aligned}
&= \|S^T r^{(k+1)}\|_2^2 / \|S^T r^{(k)}\|_2^2 \\
&= \langle r^{(k+1)}, \tilde{r}^{(k+1)} \rangle / \langle r^{(k)}, \tilde{r}^{(k)} \rangle.
\end{aligned}$$

The last step here requires some of the same manipulations as those used for line (7) above. Finally, for line (12) we have

$$\begin{aligned}
\hat{v}^{(k+1)} = \hat{r}^{(k+1)} + \hat{s}_k \hat{v}^{(k)} &\iff S^{-1} v^{(k+1)} = S^T r^{(k+1)} + \hat{s}_k S^{-1} v^{(k)} \\
&\iff v^{(k+1)} = Q^{-1} r^{(k+1)} + \hat{s}_k v^{(k)} \\
&\iff v^{(k+1)} = \tilde{r}^{(k+1)} + \hat{s}_k v^{(k)},
\end{aligned}$$

where we have multiplied by S and used the definition of Q in the penultimate step.

Therefore, the optimized algorithm for the preconditioned conjugate gradient algorithm is

Algorithm (PCG)

Input $A, b, x^{(0)}, Q, M, \epsilon$

$r^{(0)} = b - Ax^{(0)}$

Solve $Q\tilde{r}^{(0)} = r^{(0)}$ for $\tilde{r}^{(0)}$

$v^{(0)} = \tilde{r}^{(0)}$

Output $0, x^{(0)}, r^{(0)}$

for $k = 0$ to $M - 1$ do

if $v^{(k)} = 0$ then stop

$$\hat{t}_k = \frac{\langle \tilde{r}^{(k)}, r^{(k)} \rangle}{\|v^{(k)}\|_A^2}$$

$$x^{(k+1)} = x^{(k)} + \hat{t}_k v^{(k)}$$

$$r^{(k+1)} = r^{(k)} - \hat{t}_k A v^{(k)}$$

Solve $Q\tilde{r}^{(k+1)} = r^{(k+1)}$ for $\tilde{r}^{(k+1)}$

if $\langle \tilde{r}^{(k+1)}, r^{(k+1)} \rangle < \epsilon$ then

if $\|r^{(k+1)}\|_2^2 < \epsilon$ then stop

end

$$\hat{s}_k = \frac{\langle \tilde{r}^{(k+1)}, r^{(k+1)} \rangle}{\langle \tilde{r}^{(k)}, r^{(k)} \rangle}$$

$$v^{(k+1)} = \tilde{r}^{(k+1)} + \hat{s}_k v^{(k)}$$

Output $k + 1, x^{(k+1)}, r^{(k+1)}$

end

Remark: This algorithm requires the additional work need to solve the linear system $Q\tilde{r} = r$ once per iteration. Therefore we will want to choose Q so that this can be done easily and efficiently.

The two extreme cases $Q = I$ and $Q = A$ are of no interest. $Q = I$ gives us the ordinary CG algorithm, whereas $Q = A$ (with $(S = A^{-1/2})$ leads to the trivial preconditioned system $\hat{A}\hat{x} = \hat{b} \iff \hat{x} = \hat{b}$ since (using the symmetry of A)

$$\hat{A} = S^T A S = A^{-T/2} A^{T/2} A^{1/2} A^{-1/2} = I.$$

This may seem useful at first, but to get the solution x we need

$$\begin{aligned} x &= S\hat{x} = A^{-1/2}\hat{x} = A^{-1/2}\hat{b} \\ &= A^{-1/2}S^T b = A^{-1/2}A^{-T/2}b = A^{-1}b, \end{aligned}$$

which is just as complicated as the original problem.

Therefore, Q should be chosen somewhere “in between”. Moreover, we want

1. Q should be symmetric and positive definite.
2. Q should be such that $Q\tilde{r} = r$ can be solved efficiently.
3. Q should approximate A^{-1} in the sense that $\|I - Q^{-1}A\| < 1$.

Possible choices for Q

If we use the decomposition $A = L + D + L^T$ of the symmetric positive definite matrix A then

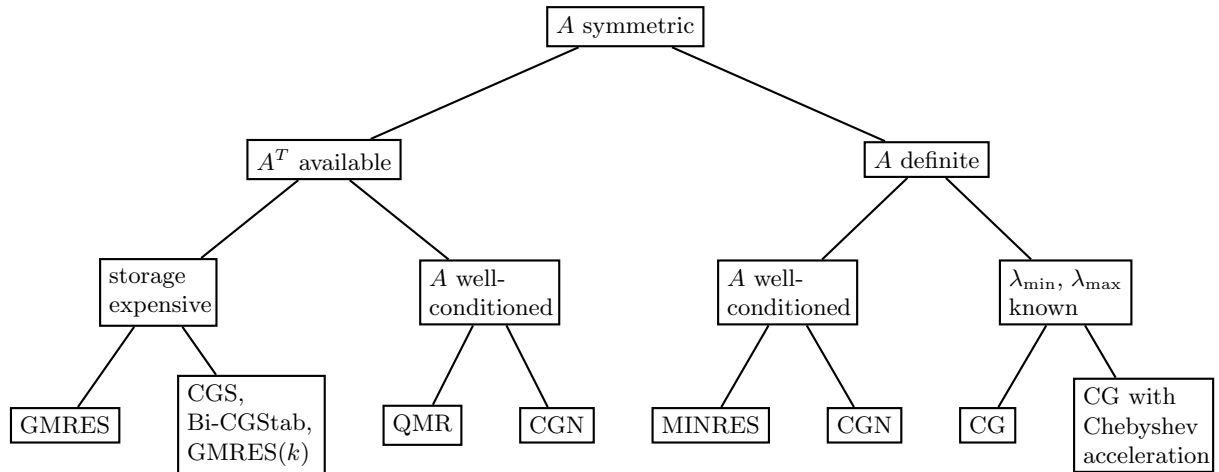
- $Q = D$ leads to Jacobi preconditioning,
- $Q = L + D$ leads to Gauss-Seidel preconditioning,
- $Q = \frac{1}{\omega}(D + \omega L)$ leads to SOR preconditioning.

Another popular preconditioner is $Q = HH^T$, where H is “close” to L . This method is referred to as incomplete Cholesky factorization (see the book by Golub and van Loan for more details).

Remark: The Matlab script `PCGDemo.m` illustrates the convergence behavior of the preconditioned conjugate gradient algorithm. The matrix A here is a 1000×1000 symmetric positive definite matrix with all zeros except $a_{ii} = 0.5 + \sqrt{i}$ on the diagonal, $a_{ij} = 1$ on the sub- and superdiagonal, and $a_{ij} = 1$ on the 100th sub- and superdiagonals, i.e., for $|i - j| = 100$. The right-hand side vector is $b = [1, \dots, 1]^T$. We observe that the basic CG algorithm converges very slowly, whereas the Jacobi-preconditioned method converges much faster.

An Overview of Iterative Solvers

We end with a decision tree (taken from the book by Demmel) containing rough guidelines for the use of most of the presently available iterative solvers. Branches to the left correspond to “No”, to the right to “Yes”. All methods mentioned belong to the family of Krylov subspace iterations.



The abbreviations used are

GMRES Generalized Minimum RESidual (1986),

CGS Conjugate Gradient Squared (1989),

Bi-CGStab Stabilized Bi-Conjugate Gradient (1992),

QMR Quasi-Minimal Residual (1991),

CGN Conjugate Gradient applied to the Normal equations,

MINRES MINimum RESidual.

Remark: A lot more details on iterative solvers can be found in the books by Demmel, Greenbaum, Golub and van Loan, and Trefethen and Bau.