Sensitivity of Linear Systems

Atkinson §8.4, NMMC[1] §1.2.7, & GvL[2] §2.6

1 Our first topic is methods for solving nonsingular linear systems of equations. The first question one usually asks about a problem, before analyzing methods to solve it, is whether a solution exists, and whether it's unique. For nonsingular linear systems the answer is yes, so we move on to analyzing how the solution depends on the data, i.e. sensitivity.

We will analyze sensitivity to errors in the inputs: the coefficient matrix and the RHS. These are forward error bounds. Later on we will see that Gaussian Elimination (one particular algorithm for finding the solution) in finite-precision floating point arithmetic finds the exact answer to a perturbed problem, with a bound on the size of the perturbations (a backwards error bound). Those results combined with these will together result in forward error bounds on the solution computed via Gaussian Elimination in finite-precision.

2 Sensitivity to RHS. (Atkinson §8.4)

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$$

where $\mathbf{A}$ is nonsingular. What happens if you perturb the RHS, eg due to roundoff, and then solve the perturbed system exactly?

$$\mathbf{A}\boldsymbol{x}_\delta = \boldsymbol{b} + \delta\boldsymbol{b}$$

$$\boldsymbol{x}_\delta = \mathbf{A}^{-1}\boldsymbol{b} + \mathbf{A}^{-1}(\delta\boldsymbol{b}) = \boldsymbol{x} + \mathbf{A}^{-1}(\delta\boldsymbol{b})$$

$$\|\boldsymbol{x}_\delta - \boldsymbol{x}\| = \|\mathbf{A}^{-1}(\delta\boldsymbol{b})\|.$$

We want the relative error; if the error is 10^{-30} it sounds good unless the norm of the solution is also 10^{-30}. Divide both sides by $\|\boldsymbol{x}\|$:

$$\frac{\|\boldsymbol{x}_\delta - \boldsymbol{x}\|}{\|\boldsymbol{x}\|} = \frac{\|\mathbf{A}\|\|\mathbf{A}^{-1}(\delta\boldsymbol{b})\|}{\|\mathbf{A}\|\|\boldsymbol{x}\|} \le \kappa(\mathbf{A})\frac{\|\delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|}.$$

The condition number is

$$\kappa(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\|.$$

Condition numbers are ≥ 1; e.g. for the 2-norm the condition number is $\sigma_{\max}/\sigma_{\min}$ where σ_i are singular values of $\mathbf{A}$. An ill-conditioned system is a system with a large condition number. If the system is ill-conditioned, then a small perturbation to the RHS can lead to large changes in the solution. For example,

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & \epsilon \end{bmatrix}, \quad \boldsymbol{b} = (1,0)^T, \quad \delta\boldsymbol{b} = (0,\delta)^T$$

The unperturbed solution is $(1,0)^T$. If δ is small then the perturbation is small, but the solution to the perturbed problem is $(1,\delta/\epsilon)^T$ so if $\epsilon \ll \delta$ then the error is large.

3 Perturbing both: Formal asymptotic analysis from GvL §2.6.2. $\mathbf{A}$ is nonsingular. Consider

$$(\mathbf{A} + \epsilon\mathbf{F})\boldsymbol{x}(\epsilon) = \boldsymbol{b} + \epsilon\boldsymbol{f}.$$

Now differentiate ('implicit differentiation')

$$\mathbf{A}\dot{\boldsymbol{x}}(\epsilon) + \mathbf{F}\boldsymbol{x}(\epsilon) + \epsilon\mathbf{F}\dot{\boldsymbol{x}}(\epsilon) = \boldsymbol{f}.$$

Now evaluate at $\epsilon = 0$, and define the notation $\boldsymbol{x}(0) = \boldsymbol{x} =$ the solution to the unperturbed problem

$$\mathbf{A}\dot{\boldsymbol{x}} + \mathbf{F}\boldsymbol{x} = \boldsymbol{f} \quad \Rightarrow \quad \dot{\boldsymbol{x}} = \mathbf{A}^{-1}(\boldsymbol{f} - \mathbf{F}\boldsymbol{x}).$$

$\boldsymbol{x}(\epsilon)$ has a Taylor series expansion of the form

$$\boldsymbol{x}(\epsilon) = \boldsymbol{x} + \epsilon\dot{\boldsymbol{x}}(0) + o(\epsilon).$$

[1]Numerical Methods in Matrix Computations by Bjork

[2]Matrix Computations by Golub and van Loan

(The 'little-o' notation $y = o(\epsilon)$ means that $\lim_{\epsilon\to 0} y/\epsilon = 0$. The 'big-o' notation $y = \mathcal{O}(\epsilon)$ means that $\lim_{\epsilon\to 0} y/\epsilon$ exists and is neither infinite nor zero.)

So we have the following

$$\frac{\|\boldsymbol{x}(\epsilon) - \boldsymbol{x}(0)\|}{\|\boldsymbol{x}(0)\|} = \epsilon\frac{\|\mathbf{A}^{-1}(\boldsymbol{f} - \mathbf{F}\boldsymbol{x})\|}{\|\boldsymbol{x}\|} + o(\epsilon).$$

We'll use the bound

$$\frac{\|\mathbf{A}^{-1}(\boldsymbol{f} - \mathbf{F}\boldsymbol{x})\|}{\|\boldsymbol{x}\|} \le \|\mathbf{A}^{-1}\|\left(\frac{\|\boldsymbol{f}\|}{\|\boldsymbol{x}\|} + \|\mathbf{F}\|\right).$$

Note that

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b} \quad \Rightarrow \quad \|\mathbf{A}\|\|\boldsymbol{x}\| \ge \|\boldsymbol{b}\| \quad \Rightarrow \quad \frac{1}{\|\boldsymbol{x}\|} \le \frac{\|\mathbf{A}\|}{\|\boldsymbol{b}\|}.$$

Using this above we obtain

$$\frac{\|\boldsymbol{x}(\epsilon) - \boldsymbol{x}\|}{\|\boldsymbol{x}\|} \le \|\mathbf{A}^{-1}\|\|\mathbf{A}\|\left(\frac{\|\epsilon\boldsymbol{f}\|}{\|\boldsymbol{b}\|} + \frac{\|\epsilon\mathbf{F}\|}{\|\mathbf{A}\|}\right) + o(\epsilon).$$

This says that the relative error is proportional to the condition number times the relative amplitude of the perturbations in the RHS and in the coefficient matrix. The above derivation (though correct) is not completely rigorous because we never justified that the derivatives exist (they do for small enough ϵ).

The above result is asymptotic, meaning that the error will be approximately equal to the formula on the right hand side *for small enough* ϵ. Unfortunately we don't know how small ϵ needs to be in order for the approximation to be accurate. We want some information about the error that will be correct for a known range of ϵ.

4 Theorem 8.4 in Atkinson is not very good. We will instead prove a bound similar to the above rigorously. To prove a rigorous bound we need a Lemma (see also Atkinson Theorems 7.10 & 7.11). Note that $\|\mathbf{F}\| < 1$ (any operator norm) implies that all eigenvalues are less than 1. If all evals of $\mathbf{F}$ are less than 1 then $\mathbf{I} - \mathbf{F}$ is nonsingular (no evals are 0).

Assuming $\|\mathbf{F}\| < 1$, note the following

$$\mathbf{I} - \mathbf{F}^{N+1} = (\mathbf{I} + \mathbf{F} + \cdots + \mathbf{F}^N) - \mathbf{F}(\mathbf{I} + \mathbf{F} + \cdots + \mathbf{F}^N) = \left(\sum_0^N \mathbf{F}^k\right)(\mathbf{I} - \mathbf{F}).$$

Multiply from the right by $(\mathbf{I} - \mathbf{F})^{-1}$

$$(\mathbf{I} - \mathbf{F}^{N+1})(\mathbf{I} - \mathbf{F})^{-1} = \sum_0^N \mathbf{F}^k$$

Take the limit as $N \to \infty$ and recall that $\|\mathbf{F}\| < 1$ implies that $\mathbf{F}^N \to \mathbf{0}$

$$(\mathbf{I} - \mathbf{F})^{-1} = \sum_0^\infty \mathbf{F}^k$$

Note similarity to Taylor series for $1/(1-x)$.

Return to fixed N:

$$\|(\mathbf{I} - \mathbf{F}^{N+1})(\mathbf{I} - \mathbf{F})^{-1}\| \le \sum_0^N \|\mathbf{F}\|^k$$

Now take limits, use geometric series

$$\|(\mathbf{I} - \mathbf{F})^{-1}\| \le \frac{1}{1 - \|\mathbf{F}\|}.$$

Finally, consider

$$(\mathbf{I}-\mathbf{F})^{-1}-\mathbf{I}=\sum_{0}^{\infty}\mathbf{F}^k-\mathbf{I}=\sum_{1}^{\infty}\mathbf{F}^k=\mathbf{F}\sum_{0}^{\infty}\mathbf{F}^k=\mathbf{F}(\mathbf{I}-\mathbf{F})^{-1}.$$

This implies that

$$\|(\mathbf{I}-\mathbf{F})^{-1}-\mathbf{I}\|=\|\mathbf{F}(\mathbf{I}-\mathbf{F})^{-1}\|\leq\frac{\|\mathbf{F}\|}{1-\|\mathbf{F}\|}.$$

5 Now return to a rigorous proof. Consider $\mathbf{A}+\mathbf{E}=(\mathbf{I}-\mathbf{F})\mathbf{A}$ where $\mathbf{F}=-\mathbf{E}\mathbf{A}^{-1}$. Assume that $\|\mathbf{E}\mathbf{A}^{-1}\|<1$ so that $\mathbf{A}+\mathbf{E}$ is nonsingular.

$$(\mathbf{A}+\mathbf{E})\boldsymbol{y}=\boldsymbol{b}+\delta\boldsymbol{b},\quad \mathbf{A}\boldsymbol{x}=\boldsymbol{b}$$

$$(\mathbf{I}-\mathbf{F})\mathbf{A}\boldsymbol{y}=\boldsymbol{b}+\delta\boldsymbol{b}$$

$$\boldsymbol{y}=\mathbf{A}^{-1}(\mathbf{I}-\mathbf{F})^{-1}(\boldsymbol{b}+\delta\boldsymbol{b})$$

$$\boldsymbol{y}-\boldsymbol{x}=\mathbf{A}^{-1}\left[\left((\mathbf{I}-\mathbf{F})^{-1}-\mathbf{I}\right)\boldsymbol{b}+(\mathbf{I}-\mathbf{F})^{-1}\delta\boldsymbol{b}\right]$$

$$\|\boldsymbol{y}-\boldsymbol{x}\|\leq\|\mathbf{A}^{-1}\|\left[\frac{\|\mathbf{F}\|}{1-\|\mathbf{F}\|}\|\boldsymbol{b}\|+\frac{1}{1-\|\mathbf{F}\|}\|\delta\boldsymbol{b}\|\right]$$

Divide by $\|\boldsymbol{x}\|$ and multiply RHS by $\|\mathbf{A}\|/\|\mathbf{A}\|$

$$\frac{\|\boldsymbol{y}-\boldsymbol{x}\|}{\|\boldsymbol{x}\|}\leq\kappa(\mathbf{A})\left[\frac{\|\mathbf{F}\|}{1-\|\mathbf{F}\|}\frac{\|\boldsymbol{b}\|}{\|\mathbf{A}\|\|\boldsymbol{x}\|}+\frac{1}{1-\|\mathbf{F}\|}\frac{\|\delta\boldsymbol{b}\|}{\|\mathbf{A}\|\|\boldsymbol{x}\|}\right]$$

$$\frac{\|\boldsymbol{y}-\boldsymbol{x}\|}{\|\boldsymbol{x}\|}\leq\kappa(\mathbf{A})\left[\frac{\|\mathbf{F}\|}{1-\|\mathbf{F}\|}+\frac{1}{1-\|\mathbf{F}\|}\frac{\|\delta\boldsymbol{b}\|}{\|\boldsymbol{b}\|}\right].$$

Now let $\epsilon=\max\{\|\mathbf{F}\|,\|\delta\boldsymbol{b}\|/\|\boldsymbol{b}\|\}(<1)$

$$\frac{\|\boldsymbol{y}-\boldsymbol{x}\|}{\|\boldsymbol{x}\|}\leq\frac{2\epsilon\kappa(\mathbf{A})}{1-\epsilon}.$$

This relies on the fact that $x/(1-x)$ and $1/(1-x)$ are increasing on $x\in(0,1)$.

The above result depends on $\|\mathbf{F}\|$ while the asymptotic result depended on $\|\mathbf{E}\|/\|\mathbf{A}\|$. You can make the connection using $\mathbf{F}\mathbf{A}=-\mathbf{E}\Rightarrow\|\mathbf{F}\|\|\mathbf{A}\|\geq\|\mathbf{E}\|\Rightarrow\|\mathbf{F}\|\geq\|\mathbf{E}\|/\|\mathbf{A}\|$.

The above result is not asymptotic; it's an upper bound. The relative error might actually be a lot smaller than this.

In summary, if the system is ill-conditioned then small perturbations in the matrix or RHS can lead to large changes in the solution. A large perturbation in the matrix or RHS can lead to large changes in the solution even when the system is well-conditioned.

6 Preconditioning. Suppose that $\mathbf{A}$, $\mathbf{M}$, $\mathbf{M}_l$, and $\mathbf{M}_r$ are all invertible matrices. The following systems all have the same solution

- Standard: $\mathbf{A}\boldsymbol{x}=\boldsymbol{b}$
- Left-preconditioned $\mathbf{M}\mathbf{A}\boldsymbol{x}=\mathbf{M}\boldsymbol{b}$
- Right-preconditioned $\mathbf{A}\mathbf{M}\boldsymbol{y}=\boldsymbol{b}$ and $\mathbf{M}\boldsymbol{y}=\boldsymbol{x}$
- Split-preconditioned $\mathbf{M}_l\mathbf{A}\mathbf{M}_r\boldsymbol{y}=\mathbf{M}_l\boldsymbol{b}$ and $\mathbf{M}_r\boldsymbol{y}=\boldsymbol{x}$.

For the left- and right-preconditioned systems, if $\mathbf{M}\approx\mathbf{A}^{-1}$ then the preconditioned system has the same solution as the original system but a better condition number. For the split-preconditioned system, if $\mathbf{A}=\mathbf{L}\mathbf{U}$ and $\mathbf{M}_l\approx\mathbf{L}^{-1}$ and $\mathbf{M}_r\approx\mathbf{U}^{-1}$ then the preconditioned system has a better condition number. We won't discuss methods for constructing preconditioners; take the numerical linear algebra course APPM 5620 offered every spring.

Gaussian Elimination & LU Factorization

Atkinson §8.1–8.4, NMMC §1.2, & GvL Chapter 3

1 Gaussian Elimination with Partial Pivoting (GEPP) is an algorithm that is guaranteed to compute the exact solution (in the absence of roundoff errors) to a nonsingular linear system in a finite number of steps. Any algorithm that computes an exact solution in a finite number of steps is a 'direct method.'

Form the augmented matrix $[\mathbf{A}|\boldsymbol{b}]$. Then:
For $k = 1 : n-1$

- Swap row k with the row that has the largest element in the k^{th} column on or below the diagonal.
- Add multiples of row k to rows $k+1 : n$ to set all elements below the k^{th} diagonal to 0.

The augmented matrix will now have the form $[\mathbf{U}|\boldsymbol{c}]$. Set $x_n = \boldsymbol{c}_n/u_{n,n}$. For $k = n-1 : -1 : 1$

- Set $x_k = (c_k - \sum_{i=n}^{k+1} u_{k,i}x_i)/u_{k,k}$.

The first loop is the forward solve, the second loop is the back solve. In the forward solve, if there's a zero on the diagonal you have to pivot (pivot means swap rows), and you can pivot in any way that produces a nonzero on the diagonal. Now in finite-precision arithmetic the probability of getting a hard zero is small anyways, so that's not really why we pivot. Instead, the strategy that you use for pivoting has an impact on the magnitude of the roundoff error; we pivot to minimize roundoff errors. I have specified a particular pivoting strategy: pivoting the largest element to the diagonal. We will later quote an error bound that depends on this particular strategy. An even better strategy from the perspective of roundoff errors is to pivot the rows and columns so that the largest element in the trailing submatrix gets moved to the diagonal. This is called 'complete' pivoting and leads to a better roundoff error bound, but is also more costly.

'Gauss-Jordan' elimination is a variant where the second loop is like the first: you eliminate elements *above* the diagonal (without pivoting). After the second loop the augmented matrix has the form $[\mathbf{I}|\boldsymbol{x}]$. Gauss-Jordan is more expensive than Gaussian Elimination.

2 Gaussian Elimination & LU Factorization. If you solve $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ multiple times with different right hand sides, the 'forward solve' part of the loop is the same every time. You will always arrive at an augmented matrix of the form $[\mathbf{U}|\boldsymbol{c}]$; the upper triangular matrix $\mathbf{U}$ will always be the same, only the vector $\boldsymbol{c}$ will depend on the right hand side $\boldsymbol{b}$. So it's wasteful to re-do the whole forward loop each time. Instead, you should save some record of the operations (pivoting, multiplying and adding rows) that occur during the forward loop. If you have to solve the system again with a different $\boldsymbol{b}$, you just re-apply the same sequence of operations to the new RHS vector $\boldsymbol{b}$ to get the new $\boldsymbol{c}$, then proceed to the back solve.

It turns out that storing the steps in the forward loop of Gaussian Elimination is like computing a matrix factorization (called a permuted LU factorization) of the form

$$\mathbf{PA} = \mathbf{LU}$$

where $\mathbf{P}$ is a permutation matrix, $\mathbf{L}$ is lower-triangular with 1 on the diagonal ('special lower triangular') and $\mathbf{U}$ is upper-triangular. Permuting rows is equivalent to multiplying from the left by a permutation matrix $\mathbf{P}_i$, and eliminating everything below the diagonal is equivalent to multiplying from the left by a special lower-triangular elimination matrix $\mathbf{L}_i$, so the whole elimination process can be written as

$$\mathbf{L}_{N-1}\left(\mathbf{P}_{N-1}\mathbf{L}_{N-2}\mathbf{P}_{N-1}\right)\left(\mathbf{P}_{N-1}\mathbf{P}_{N-2}\mathbf{L}_{N-3}\mathbf{P}_{N-2}\mathbf{P}_{N-1}\right)\cdots \\ \left(\mathbf{P}_{N-1}\cdots\mathbf{P}_2\mathbf{L}_1\mathbf{P}_2\cdots\mathbf{P}_{N-1}\right)\mathbf{P}_{N-1}\cdots\mathbf{P}_1\mathbf{A} = \mathbf{U}.$$

The elimination matrix $\mathbf{L}_i$ is an identity except for nonzeros below the diagonal of the i^{th} column. The matrices in parentheses above are symmetric permutations of $\mathbf{L}_i$ where the permutations operate on rows

& columns with indices *greater than i*. As a result, the symmetric permutations have exactly the same lower-triangular structure of nonzeros as $\mathbf{L}_i$, though the entries below the diagonal are different (they are permuted). We can thus write

$$\mathbf{L}_{N-1}\tilde{\mathbf{L}}_{N-2}\cdots\tilde{\mathbf{L}}_1\mathbf{P}_{N-1}\cdots\mathbf{P}_1\mathbf{A}=\mathbf{U}.$$

where all the $\tilde{\mathbf{L}}_i$ are (special) lower-triangular. The inverse of a (special) lower-triangular matrix is also (special) lower triangular (in fact the inverse of an elimination matrix just changes the sign of the off-diagonals), and a product of permutation matrices is itself a permutation matrix, so we can write

$$\mathbf{PA}=\mathbf{LU}$$

where

$$\mathbf{P}=\mathbf{P}_{N-1}\cdots\mathbf{P}_1$$

and

$$\mathbf{L}=\tilde{\mathbf{L}}_1^{-1}\cdots\tilde{\mathbf{L}}_{N-2}^{-1}\mathbf{L}_{N-1}^{-1}.$$

Thus GEPP is equivalent to a matrix factorization.

The matrix $\mathbf{U}$ is computed during the forward loop of GEPP. The $\mathbf{P}$ and $\mathbf{L}$ matrices can be computed during the GEPP algorithm with no extra cost (since they basically just keep a record of the operations performed by the GEPP algorithm). Generally we don't store the matrix $\mathbf{P}$ since it only has n nonzero elements, all of which are 1. Instead we store a permutation vector. The matrix $\mathbf{L}$ can be computed online easily even with pivoting; see, e.g., Algorithm 1.2.4 in NMMC. Do not use the above analysis to compute $\mathbf{L}$!

3 GEPP: Computational Cost. Operation count for row reduction (not applied to RHS). We will ignore the cost of row swaps. In step k you eliminate $n-k$ rows. At each row you compute the 'multiplier' (one division), then you perform $n-k$ multiplications and additions. So the cost at step k is $2(n-k)^2+(n-k)$ floating point operations. So the total cost is

$$\sum_{k=1}^{n-1}2(n-k)^2+(n-k)\sim\mathcal{O}(n^3).$$

The precise coefficient of the leading order term is $2n^3/3$; generally we only keep the term that grows fastest with n so that we can gauge cost for large matrices.

The back solve costs n divisions plus $n(n-1)/2$ multiplications and the same number of additions, so overall it is $\mathcal{O}(n^2)$. We say that the overall cost of GEPP/permuted LU is order n^3. Once you've computed the LU factorization of a matrix it only takes $\mathcal{O}(n^2)$ operations to use it to solve a system.

4 Cholesky. If the matrix $\mathbf{A}$ is symmetric and positive definite, then (i) no pivoting is needed, and (ii) the diagonal elements of $\mathbf{U}$ are positive. (Applying GE to $\mathbf{A}$ is the best way to check if a symmetric matrix is positive definite; not by computing the eigenvalues.) Moreover, symmetry implies

$$\mathbf{A}=\mathbf{LDL}^T$$

where $\mathbf{D}$ is a diagonal matrix whose diagonal entries are those of $\mathbf{U}$. So we can define $\mathbf{G}=\mathbf{LD}^{1/2}$, and we have $\mathbf{A}=\mathbf{GG}^T$ where $\mathbf{G}$ is the Cholesky factor of $\mathbf{A}$.

5 Tridiagonal matrices. If $\mathbf{A}$ is tridiagonal then $\mathbf{L}$ will be lower bidiagonal and $\mathbf{U}$ will have nonzeros at most two elements above the diagonal (graphical proof). If no pivoting is required (e.g. because $\mathbf{A}$ is SPD), then $\mathbf{U}$ will be upper-bidiagonal. In either case the cost of GEPP is only $\mathcal{O}(n)$.

Gaussian Elimination & LU Factorization: Roundoff Errors

Atkinson Theorem 8.5, NMMC §1.4.3, & GvL §3.3

Atkinson Theorem 8.5 is not as accurate as the results presented here, but is still OK.

1 Backward errors in the computed solution using GE with either complete or partial pivoting.

(cf NMMC Theorem 1.4.6) Let $\bar{\boldsymbol{x}}$ be the computed solution to $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$. Then there is a matrix $\mathbf{E}$ such that

$$(\mathbf{A}+\mathbf{E})\bar{\boldsymbol{x}} = \boldsymbol{b}$$

where

$$\|\mathbf{E}\|_\infty \le 1.5\frac{n^2(n+1)\boldsymbol{u}}{1-n\boldsymbol{u}}\rho_n\|\mathbf{A}\|_\infty$$

and ρ_n is the 'growth factor.'

Technically this is not a rigorous bound because it ignores terms on the order of $(n\boldsymbol{u})^2$, but when $n\boldsymbol{u}$ is small it is accurate. You could insert this into our sensitivity analysis to get a forward error bound on the computed solution; set

$$\epsilon = \|\mathbf{E}\|_\infty/\|\mathbf{A}\|_\infty \le 1.5\frac{n^2(n+1)\boldsymbol{u}}{1-n\boldsymbol{u}}\rho_n$$

and insert in

$$\frac{\|\bar{\boldsymbol{x}}-\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \le \frac{2\epsilon\kappa(\mathbf{A})}{1-\epsilon}.$$

2 Some results on the growth factor ρ_n (See NMMC)

- With partial pivoting the growth factor is sharply bounded by $\rho_n \le 2^{n-1}$, but this upper bound is usually not attained. Partial pivoting is usually deemed 'good enough' for most applications.
- With complete pivoting, the upper bound is $\rho_n < 1.8n^{1/2+\ln(n)/4}$ (Atkinson forgets the 1/2 in the exponent). Complete pivoting is extremely stable to roundoff errors.
- For a 'diagonally-dominant' matrix (definition deferred) and for tridiagonal matrices the growth factor is bounded by $\rho_n \le 2$.

3 Small residuals. The residual in the computed solution $\bar{r} = \boldsymbol{b} - \mathbf{A}\bar{\boldsymbol{x}}$ satisfies

$$\frac{\|\bar{r}\|_\infty}{\|\mathbf{A}\|_\infty\|\bar{\boldsymbol{x}}\|_\infty} \le 1.5\frac{n^2(n+1)\boldsymbol{u}}{1-n\boldsymbol{u}}\rho_n.$$

This is independent of the condition number of $\mathbf{A}$. So Gaussian elimination can reliably produce solutions that have relatively small residuals, even when the matrix is ill-conditioned. (Again, this is not a rigorous bound because it ignores terms on the order of $(n\boldsymbol{u})^2$.)

4 It's worth noting that if you're worried about roundoff errors, the easiest thing to do is to compute the solution twice and examine the differences. For example, compute once with partial pivoting and once with complete pivoting and compare. Or compute once in single precision and once in double precision and compare (or double and quad). If roundoff is limiting your accuracy, there are methods for obtaining improved accuracy; all hope is not lost, just do some reading.

Stationary Iterations: Richardson, Jacobi, Seidel, SOR

Atkinson §8.6, NMMC §4.1, GvL §11.2

1 General idea: Let $\mathbf{A} = \mathbf{M} - \mathbf{N}$ where it's easy to solve $\mathbf{M}\boldsymbol{y} = \boldsymbol{c}$, i.e. it's easy to invert $\mathbf{M}$ (though we never compute $\mathbf{M}^{-1}$!). Then define the following iteration

$$\boldsymbol{x}_{k+1} = \mathbf{M}^{-1}\mathbf{N}\boldsymbol{x}_k + \mathbf{M}^{-1}\boldsymbol{b}.$$

This is a 'splitting' for obvious reasons, and it's 'stationary' because it does the same operation at every step. We'll cover some classical splittings and their associated iterations. Generally we want to know whether $\boldsymbol{x}_k$ will converge to the solution of $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ as $k \to \infty$ *for every starting vector* $\boldsymbol{x}_0$.

First let's suppose that the iteration does converge to something, and call it $\boldsymbol{x}_\infty$. We'll verify that this limit does solve $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$:

$$\boldsymbol{x}_\infty = \mathbf{M}^{-1}\mathbf{N}\boldsymbol{x}_\infty + \mathbf{M}^{-1}\boldsymbol{b} \Rightarrow$$
$$(\mathbf{I} - \mathbf{M}^{-1}\mathbf{N})\boldsymbol{x}_\infty = \mathbf{M}^{-1}\boldsymbol{b}$$
$$(\mathbf{M} - \mathbf{N})\boldsymbol{x}_\infty = \mathbf{A}\boldsymbol{x}_\infty = \boldsymbol{b}.$$

Next we'll ask under what conditions the splitting will converge. It's easiest to show that the error goes to 0, rather than that $\boldsymbol{x}_k \to \boldsymbol{x}_\infty$. The error is $\boldsymbol{e}_k = \boldsymbol{x}_\infty - \boldsymbol{x}_k$, and it solves (use $\mathbf{M}\boldsymbol{x}_\infty = \mathbf{N}\boldsymbol{x}_\infty + \boldsymbol{b}$)

$$\boldsymbol{e}_{k+1} = \mathbf{B}\boldsymbol{e}_k, \quad \mathbf{B} = \mathbf{M}^{-1}\mathbf{N}.$$

Now we want to know under what conditions this will converge to 0 *for any initial error.* A necessary and sufficient condition is:

> The iteration is convergent (i.e converges for any initial condition) iff the largest eigenvalue of $\mathbf{B}$ has amplitude less than 1. (I.e. the spectral radius $\rho(\mathbf{B}) < 1$.)

See Atkinson Theorem 7.9. This is basically linear algebra background, so we won't prove it.

It's not always easy to check this condition. The following condition is sufficient but not necessary:

> If $\|\mathbf{B}\| < 1$ for some matrix norm induced by a vector norm, then the iteration is convergent.

The proof of this is simple and is omitted.

How fast is the iteration converging?

> If $\rho(\mathbf{B}) < 1$ then
> $$\lim_{k\to\infty}\left(\frac{\|\boldsymbol{e}_k\|}{\|\boldsymbol{e}_0\|}\right)^{1/k} = \rho(\mathbf{B})$$
> for any vector norm.

This means that for large enough k,

$$\|\boldsymbol{e}_k\| \lesssim \|\boldsymbol{e}_0\|\rho(\mathbf{B})^k.$$

The proof of the above is based on a nontrivial fact about the spectral radius, found in NMMC Lemma 4.1.2. If the iteration matrix $\mathbf{B}$ is non-normal, actual errors can *grow* at low k, even when the method is convergent. They can grow by many orders of magnitude before eventually settling into asymptotic decay, so be careful.

2 The standard splittings first decompose $\mathbf{A}$ into diagonal, upper, and lower parts:

$$\mathbf{A} = \mathbf{D} - \mathbf{E} - \mathbf{F} = \mathbf{D}(\mathbf{I} - \mathbf{L} - \mathbf{U}).$$

- Gauss-Jacobi, aka Jacobi. Based on the idea that it's easy to invert a diagonal matrix. The splitting is $\mathbf{M} = \mathbf{D}$, and the iteration is
$$\boldsymbol{x}_{k+1} = \mathbf{D}^{-1}(\mathbf{E} + \mathbf{F})\boldsymbol{x}_k + \mathbf{D}^{-1}\boldsymbol{b}.$$
- Gauss-Seidel. Based on the idea that a triangular matrix is easy to invert. The splitting is $\mathbf{M} = \mathbf{D} - \mathbf{E}$, and the iteration is
$$\boldsymbol{x}_{k+1} = (\mathbf{D} - \mathbf{E})^{-1}\mathbf{F}\boldsymbol{x}_k + (\mathbf{D} - \mathbf{E})^{-1}\boldsymbol{b}.$$
- Successive Over-Relaxation (SOR). Part of the diagonal goes into $\mathbf{M}$ and part goes into $\mathbf{N}$:
$$\boldsymbol{x}_{k+1} = (\mathbf{D} - \omega\mathbf{E})^{-1}[\omega\mathbf{F} + (1-\omega)\mathbf{D}]\boldsymbol{x}_k + \omega(\mathbf{D} - \omega\mathbf{E})^{-1}\boldsymbol{b}.$$

Both GS and SOR have 'symmetric' variants. A single step of symmetric GS consists of a step with $\mathbf{M} = \mathbf{D} - \mathbf{E}$, followed by a step with $\mathbf{M} = \mathbf{D} - \mathbf{F}$.

Convergence Theorems

(Definition) The square matrix $\mathbf{A}$ is diagonally dominant by rows when the absolute value of the diagonal element in each row is larger than the sum of the absolute values of the remaining terms

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}|.$$

"Strict" diagonal dominance replaces the $\geq$ by $>$. Diagonal dominance by columns is similar.

The Gershgorin Disk theorem (Atkinson Theorem 9.1) guarantees that strictly diagonally dominant matrices are nonsingular.

Convergence Theorem #1 If $\mathbf{A}$ is strictly diagonally dominant then the Gauss Jacobi and Seidel iterations are convergent.

Proof for Jacobi: The Jacobi iteration matrix is $\mathbf{B} = \mathbf{D}^{-1}(\mathbf{E} + \mathbf{F})$. If $\mathbf{A}$ is row-dominant then the ∞ norm is strictly less than 1. (If $\mathbf{A}$ is column-dominant then the 1 norm is strictly less than 1.)

Proof for Seidel: Note $\mathbf{B} = (\mathbf{D} - \mathbf{E})^{-1}\mathbf{F}$. Let $\boldsymbol{e}_k$ be the standard basis vector such that $\|\mathbf{B}^T\|_1 = \|\mathbf{B}\|_\infty = \|\mathbf{B}^T\boldsymbol{e}_k\|_1$. Note that

$$(\mathbf{I} - \mathbf{D}^{-1}\mathbf{E})\mathbf{B} = \mathbf{D}^{-1}(\mathbf{D} - \mathbf{E})(\mathbf{D} - \mathbf{E})^{-1}\mathbf{F} = \mathbf{D}^{-1}\mathbf{F} \Rightarrow$$
$$\mathbf{B} = \mathbf{D}^{-1}\mathbf{E}\mathbf{B} + \mathbf{D}^{-1}\mathbf{F}$$

Take transpose, then multiply by $\boldsymbol{e}_k$ and take the 1-norm

$$\|\mathbf{B}\|_\infty \leq \|\mathbf{B}\|_\infty \|(\mathbf{D}^{-1}\mathbf{E})^T\boldsymbol{e}_k\|_1 + \|(\mathbf{D}^{-1}\mathbf{F})^T\boldsymbol{e}_k\|_1$$

$$\|\mathbf{B}\|_\infty \leq \frac{\|(\mathbf{D}^{-1}\mathbf{F})^T\boldsymbol{e}_k\|_1}{1 - \|(\mathbf{D}^{-1}\mathbf{E})^T\boldsymbol{e}_k\|_1}$$

To justify the division without reversing the inequality we need that $\|(\mathbf{D}^{-1}\mathbf{E})^T\boldsymbol{e}_k\|_1 < 1$. The fact that $\mathbf{A}$ is strictly-diagonally-dominant by rows guarantees this (since the vector is a column sum of a transpose). Now we're trying to prove that $\|\mathbf{B}\|_\infty < 1$; this is transparent if

$$\|(\mathbf{D}^{-1}\mathbf{E})^T\boldsymbol{e}_k\|_1 + \|(\mathbf{D}^{-1}\mathbf{F})^T\boldsymbol{e}_k\|_1 < 1.$$

Again this is just a full row sum of the Jacobi iteration matrix broken into two absolute-value pieces (absolute value of the sum of elements left of the diagonal plus absolute value of the sum of those right of the diagonal); strict diagonal-dominance then guarantees the desired result. Column-wise DD is similar.

It is possible to weaken the conditions to *irreducible diagonal dominance.*

Convergence Theorem #2 If $\mathbf{A}$ is symmetric positive definite then the Gauss Seidel iteration is convergent.

Proof from GvL, Theorem 11.2.3. The proof examines the spectral radius, not the norm (naturally, since we only know something about the eigenvalues of $\mathbf{A}$). The iteration matrix for a symmetric $\mathbf{A}$ is

$$\mathbf{B}_{GS} = (\mathbf{D} - \mathbf{E})^{-1}\mathbf{E}^T.$$

First notice that this matrix has the same eigenvalues as

$$\mathbf{G} = \mathbf{D}^{1/2}\mathbf{B}\mathbf{D}^{-1/2} = (\mathbf{I} - \mathbf{L})^{-1}\mathbf{L}^T$$

where

$$\mathbf{L} = \mathbf{D}^{-1/2}\mathbf{E}\mathbf{D}^{-1/2}.$$

Suppose that $\boldsymbol{v}$ is a unit-2-norm eigenvector of $\mathbf{G}$

$$(\mathbf{I} - \mathbf{L})^{-1}\mathbf{L}^T\boldsymbol{v} = \lambda\boldsymbol{v}.$$

Move inverse to RHS then take standard inner product:

$$\boldsymbol{v}^*\mathbf{L}^T\boldsymbol{v} = \lambda(1 - \boldsymbol{v}^*\mathbf{L}\boldsymbol{v}).$$

Note that (if $\mathbf{A}$ is real) then $\boldsymbol{v}^*\mathbf{L}^T\boldsymbol{v} = \overline{\boldsymbol{v}^*\mathbf{L}\boldsymbol{v}}$. Let $\boldsymbol{v}^*\mathbf{L}\boldsymbol{v} = a + ib$, so that

$$a - ib = \lambda(1 - a - ib) \Rightarrow |\lambda|^2 = \frac{a^2 + b^2}{1 - 2a + a^2 + b^2}.$$

Now consider that since $\mathbf{A}$ is SPD, we have

$$0 < \boldsymbol{v}^*\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\boldsymbol{v} = 1 - \boldsymbol{v}^*\mathbf{L}\boldsymbol{v} - \boldsymbol{v}^*\mathbf{L}^T\boldsymbol{v} = 1 - 2a.$$

This implies that $|\lambda|^2 < 1$, as desired.

SOR First show that a necessary condition is $0 < \omega < 2$. The SOR iteration matrix is

$$\mathbf{B} = (\mathbf{D} - \omega\mathbf{E})^{-1}[\omega\mathbf{F} + (1 - \omega)\mathbf{D}] = (\mathbf{I} - \mathbf{L})^{-1}[(1 - \omega)\mathbf{I} + \omega\mathbf{U}].$$

The determinant of $\mathbf{B}$ is

$$\det[\mathbf{B}] = \det[(\mathbf{I} - \mathbf{L})^{-1}]\det[(1 - \omega)\mathbf{I} + \omega\mathbf{U}] = 1 \times (1 - \omega)^n.$$

The determinant is the product of the eigenvalues. Let λ_i be the eigenvalues of $\mathbf{B}$, then

$$\rho(\mathbf{B})^n = |\lambda_{\max}|^n \geq \Pi_{i=1}^n|\lambda_i|^n = |1 - \omega|^n.$$

In order to have $\rho(\mathbf{B}) < 1$ we have to have $0 < \omega < 2$.

Convergence Theorem #3 If $\mathbf{A}$ is symmetric positive definite and $0 < \omega < 2$ then SOR is convergent.

This is Theorem 4.10 from Saad "Iterative methods for sparse linear systems" given there without proof or reference. SOR was a hot item ~40 years ago, and has a very well-developed theory. It's not widely used any more as a stand-alone method.

Steepest Descent & Conjugate Gradients

1 We will next look at Conjugate Gradient algorithm, which is a bridge between iterative & direct methods. It is 'iterative' in the sense that it produces a new, better approximation at each step, but it is also 'direct' in the sense that after n steps in exact arithmetic it produces the true solution. To understand CG, we start with steepest descent, which is a true iterative method. Both CG and steepest descent apply to systems with $\mathbf{A}$ symmetric positive-definite.

Consider

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\mathbf{A}\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{b}$$

where $\mathbf{A}$ is SPD. The gradient is

$$\nabla f = \mathbf{A}\boldsymbol{x} - \boldsymbol{b}.$$

The only critical point is the solution to $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$. The Hessian is $\mathbf{A}$; since it's SPD, the critical point is a minimizer. This function is quadratic, so the solution to $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ is the unique global minimizer of f. Solving $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$ is therefore equivalent to finding the minimizer of f.

Steepest descent is an iterative method for finding the minimum of f. It takes the form

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \boldsymbol{g}_k$$

where $\boldsymbol{g}_k$ is the gradient of f at $\boldsymbol{x}_k$, and α_k is some scalar chosen so that $f(x_{k+1}) < f(x_k)$. The gradient of f at x_k is

$$\boldsymbol{g}_k = \nabla f|_{x_k} = \mathbf{A}\boldsymbol{x}_k - \boldsymbol{b} = -\boldsymbol{r}_k$$

where $\boldsymbol{r}_k$ is the residual at step k. (Recall that the direction of steepest increase of a function is the gradient, and the direction of steepest decrease/descent is minus the gradient.)

How can we choose α_k? Note that

$$\begin{aligned}
f(\boldsymbol{x}_{k+1}) &= \frac{1}{2}(\boldsymbol{x}_k - \alpha_k\boldsymbol{g}_k)^T\mathbf{A}(\boldsymbol{x}_k - \alpha_k\boldsymbol{g}_k) - (\boldsymbol{x}_k - \alpha_k\boldsymbol{g}_k)^T\boldsymbol{b} \\
&= f(x_k) - \alpha_k\left(\boldsymbol{g}_k^T\mathbf{A}\boldsymbol{x}_k - \boldsymbol{g}_k^T\boldsymbol{b}\right) + \frac{\alpha_k^2}{2}\boldsymbol{g}_k^T\mathbf{A}\boldsymbol{g}_k \\
&= f(x_k) - \alpha_k\boldsymbol{g}_k^T\boldsymbol{g}_k + \frac{\alpha_k^2}{2}\boldsymbol{g}_k^T\mathbf{A}\boldsymbol{g}_k
\end{aligned}$$

An exact line search chooses α_k to minimize this expression; the minimum occurs at

$$\alpha_k = \frac{\boldsymbol{g}_k^T\boldsymbol{g}_k}{\boldsymbol{g}_k^T\mathbf{A}\boldsymbol{g}_k} = \frac{\|\boldsymbol{r}_k\|_2^2}{\boldsymbol{r}_k^T\mathbf{A}\boldsymbol{r}_k}.$$

This is the steepest descent method with exact line search. It can be quite slow, especially when $\mathbf{A}$ is ill-conditioned; the objective function f has level sets that are highly-anisotropic ellipsoids and the steepest-descent method jumps back & forth a lot.

Now consider if and how the error decreases. The error dynamics is

$$\boldsymbol{e}_{k+1} = \boldsymbol{e}_k - \alpha_k\boldsymbol{r}_k.$$

Since $\mathbf{A}$ is SPD, it induces a norm:

$$\|\boldsymbol{x}\|_A^2 = \boldsymbol{x}^T\mathbf{A}\boldsymbol{x}.$$

The steepest-descent method decreases the A-norm of the error at every step:

$$\begin{aligned}
\|\boldsymbol{e}_{k+1}\|_A^2 &= \|\boldsymbol{e}_k\|_A^2 - 2\alpha_k\boldsymbol{r}_k^T\mathbf{A}\boldsymbol{e}_k + \alpha_k^2\|\boldsymbol{r}_k\|_A^2 \\
&= \|\boldsymbol{e}_k\|_A^2 - 2\alpha_k\|\boldsymbol{r}_k\|_2^2 + \alpha_k^2\|\boldsymbol{r}_k\|_A^2 \\
&= \|\boldsymbol{e}_k\|_A^2 - \frac{\|\boldsymbol{r}_k\|_2^4}{\|\boldsymbol{r}_k\|_A^2} < \|\boldsymbol{e}_k\|_A^2
\end{aligned}$$

This fact by itself is not enough to guarantee convergence, but steepest descent is in fact convergent whenever **A** is SPD. The rate of convergence can be bounded as follows (Saad §5.3.1). The A-norm of the error at step $k+1$ is bounded by

$$\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}_*\|_A \le \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} \|\boldsymbol{x}^{(k)} - \boldsymbol{x}_*\|_A$$

where λ are eigenvalues of **A**. As usual, the convergence can be slow when **A** is ill-conditioned.

2 Steepest-descent minimizes f over a 1D affine subspace. Perhaps we could improve by minimizing over a larger subspace, but which subspace to choose? Steepest descent uses the residual, which is easy to compute. Consider the following:

1. First do a steepest-descent step.
2. Now, compute the new residual, but find the minimum in the affine subspace $\boldsymbol{x}_1$+span$\{\boldsymbol{r}_0, \boldsymbol{r}_1\}$ instead of just $\boldsymbol{x}_1 + \text{span}\{\boldsymbol{r}_1\}$.
3. At step k, find the minimum in the affine subspace $\boldsymbol{x}_k$+span$\{\boldsymbol{r}_0, \ldots, \boldsymbol{r}_k\}$.

Try it for step 2:

$$\boldsymbol{x}_2 = \boldsymbol{x}_1 + \alpha_0 \boldsymbol{r}_0 + \alpha_1 \boldsymbol{r}_1$$

$$f(\boldsymbol{x}_2) = f(\boldsymbol{x}_1) + \alpha_0 \boldsymbol{r}_0^T(\mathbf{A}\boldsymbol{x}_1 - \boldsymbol{b}) + \alpha_1 \boldsymbol{r}_1^T(\mathbf{A}\boldsymbol{x}_1 - \boldsymbol{b}) + \alpha_0\alpha_1 \boldsymbol{r}_0^T \mathbf{A} \boldsymbol{r}_1$$

$$f(\boldsymbol{x}_2) = f(\boldsymbol{x}_1) + \alpha_0 \boldsymbol{r}_0^T \boldsymbol{r}_1 + \alpha_1 \boldsymbol{r}_1^T \boldsymbol{r}_1 + \alpha_0\alpha_1 \boldsymbol{r}_0^T \mathbf{A} \boldsymbol{r}_1 + \frac{1}{2}(\alpha_0^2 \boldsymbol{r}_0^T \mathbf{A} \boldsymbol{r}_0 + \alpha_1^2 \boldsymbol{r}_1^T \mathbf{A} \boldsymbol{r}_1)$$

We want to find the α_0 and α_1 that minimize $f(\boldsymbol{x}_2)$. This is exactly the same kind of problem we started with, but 2D; at step k it will be a k-dimensional quadratic minimization problem. How will we solve it? Steepest descents? The same algorithm we're using now? That way lies madness.

3 The problem with the foregoing algorithm is that at step k we have to minimize a quadratic in k variables, which is equivalent to solving a $k \times k$ SPD linear system, which is basically the problem we were trying to solve originally.

The key to the solution is to change basis. We used the obvious basis for span$\{\boldsymbol{r}_0, \boldsymbol{r}_1\}$, but this basis is inconvenient. Every n-dimensional subspace has an orthogonal basis; orthogonality is always defined with respect to an inner product. Rather than use the standard inner product (dot), use a basis that's orthogonal with respect to the following inner product: $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \boldsymbol{u}^T \mathbf{A} \boldsymbol{v}$. This is only an inner product when **A** is SPD (in fact, any inner product on $\mathbb{R}^n$ can be written in this form for some SPD matrix). Vectors that are 'orthogonal' with respect to this inner product are called 'conjugate' or '**A**-conjugate.'

4 For convenience suppose that $\boldsymbol{x}_0 = 0$ so that $\boldsymbol{r}_0 = \boldsymbol{b}$. The CG algorithm starts with a steepest-descent step

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 + a\boldsymbol{r}_0 = a\boldsymbol{r}_0.$$

The next step seeks to minimize over the subspace $\boldsymbol{x}_2 = \boldsymbol{x}_1 + b\boldsymbol{r}_0 + c\boldsymbol{r}_1 = (a+b)\boldsymbol{r}_0 + c\boldsymbol{r}_1$. Because we started at 0, we're seeking a solution within the subspace spanned by $\{\boldsymbol{r}_0, \boldsymbol{r}_1\}$ (rather than an affine subspace if we started at a nonzero location). At step $k+1$ we're seeking $\boldsymbol{x}_{k+1} \in$span$\{\boldsymbol{r}_0, \ldots, \boldsymbol{r}_k\}$.

Suppose $\{\boldsymbol{p}_0, \ldots, \boldsymbol{p}_k\}$ is any **A**-conjugate basis for the span of $\boldsymbol{r}_0, \ldots, \boldsymbol{r}_k$. Then

$$\boldsymbol{x}_{k+1} = \sum_{i=0}^{k} \alpha_i \boldsymbol{p}_i$$

I really should have used $\alpha_{i,k}$, but you'll see why I don't need to shortly. Look at $f(\boldsymbol{x}_{k+1})$ using conjugacy

$$f(\boldsymbol{x}_{k+1}) = \sum_{i=0}^{k} \boldsymbol{p}_i^T \left(\frac{\alpha_i^2}{2} \mathbf{A} \boldsymbol{p}_i - \alpha_i \boldsymbol{b} \right).$$

The α_i that minimize this are

$$\alpha_i = \frac{\boldsymbol{p}_i^T \boldsymbol{b}}{\boldsymbol{p}_i^T \mathbf{A} \boldsymbol{p}_i}.$$

Clearly this doesn't depend on k, unless we choose a different set of $\boldsymbol{p}_i$ at every step. The next section shows that we don't need to do that: at each new step k we just add a new $\boldsymbol{p}_k$ to our basis. It also means that

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k$$

(because the $\boldsymbol{p}$ vectors don't change as you go from step to step). We will also need to note that $\alpha_k \neq 0$ unless $\boldsymbol{x}_k$ is the solution. Since the Hessian of f is positive definite, expanding the search space will always lead to a decrease in f unless you're already at the minimizer.

5 To move to step $k+1$ we need to construct an $\mathbf{A}$-conjugate basis for the span of $\boldsymbol{r}_0, \ldots, \boldsymbol{r}_{k+1}$. We could do this using Gram-Schmidt:

$$\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} - \frac{\langle \boldsymbol{p}_0, \boldsymbol{r}_{k+1} \rangle}{\|\boldsymbol{p}_0\|_A^2} \boldsymbol{p}_0 - \ldots - \frac{\langle \boldsymbol{p}_k, \boldsymbol{r}_{k+1} \rangle}{\|\boldsymbol{p}_k\|_A^2} \boldsymbol{p}_k.$$

This will only work if $\boldsymbol{r}_{k+1}$ is not in the span of the previous residuals.

Now I will show that $\boldsymbol{r}_{k+1}$ is orthogonal (not conjugate) to all previous residuals, which implies that it's not in the span (unless it's 0, in which case we're done: $\boldsymbol{x}_{k+1}$ is the solution). First compute the residual at $k+1$:

$$\boldsymbol{r}_{k+1} = \boldsymbol{b} - \mathbf{A}\boldsymbol{x}_{k+1} = \boldsymbol{b} - \sum_{i=0}^{k} \alpha_i \mathbf{A} \boldsymbol{p}_i.$$

Take the dot product with $\boldsymbol{p}_j$:

$$\boldsymbol{p}_j^T \boldsymbol{r}_{k+1} = \boldsymbol{p}_j^T \boldsymbol{b} - \sum_{i=0}^{k} \alpha_i \boldsymbol{p}_j^T \mathbf{A} \boldsymbol{p}_i$$

Using conjugacy

$$\boldsymbol{p}_j^T \boldsymbol{r}_{k+1} = \boldsymbol{p}_j^T \boldsymbol{b} - \alpha_j \boldsymbol{p}_j^T \mathbf{A} \boldsymbol{p}_j = 0, \; j = 0, \ldots, k$$

Last equality used definition of α_j. This only shows that $\boldsymbol{r}_{k+1}$ is orthogonal to $\boldsymbol{p}_i$ for $i = 1, \ldots, k$, but since the $\boldsymbol{p}$ vectors span the same subspace as the $\boldsymbol{r}$ vectors this result implies what we want to prove, i.e. that $\boldsymbol{r}_{k+1}$ is orthogonal to all previous residuals.

This section shows that either (i) $\boldsymbol{r}_{k+1} = 0$ in which case you've found the exact solution, or (ii) $\boldsymbol{r}_{k+1}$ is orthogonal to all previous ones so you can use Gram-Schmidt to construct $\boldsymbol{p}_{k+1}$.

6 We could stop there, but when you implement you'll find that almost all the terms in the Gram-Schmidt construction of $\boldsymbol{p}_{k+1}$ are zero, leaving

$$\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} - \frac{\langle \boldsymbol{p}_k, \boldsymbol{r}_{k+1} \rangle}{\|\boldsymbol{p}_k\|_A^2} \boldsymbol{p}_k.$$

The goal of this section is to prove that statement, i.e. to prove that

$$\langle \boldsymbol{p}_j, \boldsymbol{r}_{k+1} \rangle = 0, \; j = 0, \ldots, k-1.$$

- Recall that $\boldsymbol{r}_{k+1}$ is orthogonal to the span of the $\boldsymbol{p}_j$; this implies that $\boldsymbol{r}_{k+1}$ must also be orthogonal to all the $\boldsymbol{r}_j$ $(j = 1, \ldots, k)$ (because span of p's is equal to the span of the r's).

- Now notice a recursion for the residuals

$$\boldsymbol{r}_{k+1} = \boldsymbol{b} - \mathbf{A}\boldsymbol{x}_{k+1} = \boldsymbol{b} - \mathbf{A}(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) = \boldsymbol{r}_k - \alpha_k \mathbf{A} \boldsymbol{p}_k.$$

  Now take the dot product with $\boldsymbol{r}_{k+1}$, using orthogonality noted above

$$\|\boldsymbol{r}_{k+1}\|_2^2 = -\alpha_k \langle \boldsymbol{r}_{k+1}, \boldsymbol{p}_k \rangle \qquad (*)$$

- Now expand the recursion

$$\boldsymbol{r}_{k+1} = \boldsymbol{b} - \mathbf{A}(\boldsymbol{x}_{k-1} + \alpha_{k-1}\boldsymbol{p}_{k-1} + \alpha_k \boldsymbol{p}_k) = \boldsymbol{r}_{k-1} - \alpha_{k-1}\mathbf{A}\boldsymbol{p}_{k-1} - \alpha_k \mathbf{A}\boldsymbol{p}_k.$$

  Now take the dot product with $\boldsymbol{r}_{k+1}$

$$\|\boldsymbol{r}_{k+1}\|_2^2 = -\alpha_{k-1}\langle \boldsymbol{r}_{k+1}, \boldsymbol{p}_{k-1}\rangle - \alpha_k \langle \boldsymbol{r}_{k+1}, \boldsymbol{p}_k\rangle$$

  Using (*) this implies

$$0 = -\alpha_{k-1}\langle \boldsymbol{r}_{k+1}, \boldsymbol{p}_{k-1}\rangle$$

  (Recall that we noted that $\alpha_k \neq 0$ unless $\boldsymbol{x}_k$ is the solution.)

- Continuing inductively yields the desired result. I.e. expand again and use previous results to show $\alpha_{k-j}\langle \boldsymbol{r}_{k+1}, \boldsymbol{p}_{k-1}\rangle = 0$.

7 We just showed that

$$\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} - \frac{\langle \boldsymbol{p}_k, \boldsymbol{r}_{k+1}\rangle}{\|\boldsymbol{p}_k\|_A^2}\boldsymbol{p}_k.$$

Computing the coefficient requires $\mathcal{O}(n^2)$ flops for a non-sparse matrix $\mathbf{A}$. It turns out that there is an equivalent way of computing the coefficient that only costs $\mathcal{O}(n)$ flops. To wit,

$$\langle \boldsymbol{p}_k, \boldsymbol{r}_{k+1}\rangle = \boldsymbol{p}_k^T \mathbf{A}\boldsymbol{r}_{k+1} = \boldsymbol{r}_{k+1}^T \mathbf{A}\boldsymbol{p}_k$$

Now use

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k \mathbf{A}\boldsymbol{p}_k \Rightarrow \mathbf{A}\boldsymbol{p}_k = -\frac{1}{\alpha_k}(\boldsymbol{r}_{k+1} - \boldsymbol{r}_k)$$

Plugging in and using that $\boldsymbol{r}_{k+1}^T\boldsymbol{r}_k = 0$

$$\langle \boldsymbol{p}_k, \boldsymbol{r}_{k+1}\rangle = \frac{1}{\alpha_k}\boldsymbol{r}_{k+1}^T\boldsymbol{r}_{k+1} = -\frac{\|\boldsymbol{r}_{k+1}\|_2^2}{\alpha_k}.$$

Now look at the denominator

$$\|\boldsymbol{p}_k\|_A^2 = \boldsymbol{p}_k^T\mathbf{A}\boldsymbol{p}_k = (\boldsymbol{r}_k - ()\boldsymbol{p}_{k-1})^T\mathbf{A}\boldsymbol{p}_k = \boldsymbol{r}_k^T\left(\frac{1}{\alpha_k}(\boldsymbol{r}_{k+1} - \boldsymbol{r}_k)\right) = \frac{\boldsymbol{r}_k^T\boldsymbol{r}_k}{\alpha_k} = \frac{\|\boldsymbol{r}_k\|_2^2}{\alpha_k}$$

(using conjugacy of $\boldsymbol{p}_{k-1}$ and $\boldsymbol{p}_k$ and orthgonality of $\boldsymbol{r}_{k+1}$ and $\boldsymbol{r}_k$). So we can write

$$\boldsymbol{p}_{k+1} = \boldsymbol{r}_{k+1} - \frac{\langle \boldsymbol{p}_k, \boldsymbol{r}_{k+1}\rangle}{\|\boldsymbol{p}_k\|_A^2}\boldsymbol{p}_k = \boldsymbol{r}_{k+1} + \frac{\|\boldsymbol{r}_{k+1}\|_2^2}{\|\boldsymbol{r}_k\|_2^2}\boldsymbol{p}_k.$$

The rightmost expression only costs $\mathcal{O}(n)$ flops to evaluate, and is therefore cheaper than the middle expression.

8 The CG method will produce the exact solution for any SPD $\mathbf{A}$ after at most n steps in exact arithmetic. In finite-precision arithmetic we usually stop the iteration well before n steps, once the residual is small enough. We will not discuss methods for deciding when to stop the iteration.