Lecture `iterate`: Iterative methods for linear systems

November 11, 2022

Summary: Introduction to classical iterative methods for solving linear systems: Jacobi, Gauss-Seidel, and Successive Over-Relaxation (SOR). Comments on modern Krylov methods.

References: Some aspects taken from C. T. Kelley's *Iterative Methods for Linear and Nonlinear Equations*; see Chapter 1.

1 Introduction

Throughout we consider a square nonsingular linear system

$$A\mathbf{x} = \mathbf{b} \tag{1}$$

of equations, where $A \in \mathbb{R}^{n\times n}$ is an invertible matrix. Broadly, there are two approaches for solving this system.

- **Direct methods.** For example, solution via LU or QR factorization. In *exact arithmetic* these methods yield the exact solution in a finite number of arithmetic operations. How these methods perform when executed in floating-point arithmetic is separate issue.
- **Iterative methods.** These methods produce a sequence[1] $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \cdots$ of *approximate* solutions, and we hope that $\mathbf{x}_k \to \mathbf{x} = A^{-1}\mathbf{b}$ as $k \to \infty$. Even in exact arithmetic one might need to "iterate forever" (indeed send $k \to \infty$) to reach the exact solution. How these methods perform when executed in floating-point arithmetic is also a separate issue. Iterative methods may themselves be classified as follows.
  - **Classical methods.** These are also called *stationary iterative methods*, and are a form of fixed-point iteration. Their defining attribute is that the kth iterate $\mathbf{x}_k$ is defined solely in terms of the $(k-1)$st iterate $\mathbf{x}_{k-1}$. The Jacobi, Gauss-Seidel, and Successive Over-Relaxation methods are stationary iterative methods.
  - **Krylov methods.** These methods determine the kth iterate $\mathbf{x}_k$ in terms of the whole history $\{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{k-1}\}$ of previous iterates. The most influential and useful of these modern methods is the *generalized minimum residual method* (GMRES) developed by Yousef Saad and Martin H. Schultz in 1986.

[1]The index k in the sequence $\{\mathbf{x}_k\}_{k=0}^{\infty}$ is the "name label" identifying which vector element $\mathbf{x}_k$ of the sequence we are considering. To denote the jth component of $\mathbf{x}_k$, one might use either x_{jk} or $\mathbf{x}_k(j)$. In these notes we mostly adopt the latter notation.

These notes focus on the classical methods. Given the comments above, one might worry that these method are out-of-date and of little use. However, successful use of a Krylov method typically relies on *preconditioning.* We won't attempt to describe this subject here, but note that often aspects of the classical methods are used in the design of effective *preconditioners.* Therefore, the ideas associated with the old methods remain relevant.

2 Richardson iteration

If $B \in \mathbb{R}^{n\times n}$ is an invertible matrix, then the systems $A\mathbf{x} = \mathbf{b}$ and $BA\mathbf{x} = B\mathbf{b}$ are *equivalent*, that is they share the same solution space. Since A has also been assumed invertible, this means both systems share the same unique solution: $(BA)^{-1}B\mathbf{b} = A^{-1}B^{-1}B\mathbf{b} = A^{-1}\mathbf{b}$.

2.1 Derivation and convergence

If $BA\mathbf{x} = B\mathbf{b}$, then $\mathbf{0} = -BA\mathbf{x} + B\mathbf{b}$, and, upon addition of $\mathbf{x}$ to both sides,

$$\mathbf{x} = (I - BA)\mathbf{x} + B\mathbf{b}. \tag{2}$$

This identity yields the fixed-point scheme known as *Richardson iteration*[2]

$$\mathbf{x}_{k+1} = (I - BA)\mathbf{x}_k + B\mathbf{b}. \tag{3}$$

This is a stationary iterative method of the general form $\mathbf{x}_{k+1} = M\mathbf{x}_k + \mathbf{c}$; assuming M and $\mathbf{c}$ are fixed and given, $\mathbf{x}_{k+1}$ is determined solely by $\mathbf{x}_k$. In this context, the matrix B is called an "approximate inverse" of A, and clearly the scheme converges if (say in the infinity norm) $\rho = \|I - BA\|_\infty < 1$. Indeed, if $\mathbf{x}^\star$ is the unique solution to the linear system $A\mathbf{x} = \mathbf{b}$, then subtraction of $\mathbf{x}^\star = (I - BA)\mathbf{x}^\star + B\mathbf{b}$ from (3) yields

$$\mathbf{e}_{k+1} = (I - BA)\mathbf{e}_k, \tag{4}$$

where $\mathbf{e}_k = \mathbf{x}_k - \mathbf{x}^\star$. From this equation $e_{k+1} \le \rho e_k$, where $e_k = \|\mathbf{e}_k\|_\infty$. By induction then, $e_k \le \rho^k e_0$, which is the statement of convergence $\mathbf{x}_k \to \mathbf{x}^\star$ as $k \to \infty$. The observations above show that B needs to be a "good enough" approximate inverse to ensure $\rho < 1$. Notice that if $B = A^{-1}$, then $\rho = 0$. $B = A^{-1}$ is the perfect "approximate inverse" of A!

2.2 Richardson iteration as a "quasi-Newton method"

Consider again the Newton method for finding the root of a nonlinear system $\mathbf{F}(\mathbf{x}) = \mathbf{0}$ of equations:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [D\mathbf{F}(\mathbf{x}_k)]^{-1}\mathbf{F}(\mathbf{x}_k) \tag{5}$$

As described further in the **root4** lecture, a *quasi-Newton method* replaces the exact Jacobian $D\mathbf{F}(\mathbf{x})$ with an approximate one.

[2]According to Wikipedia, Lewis Fry Richardson, FRS (1881 to 1953) was an "English mathematician, physicist, meteorologist, psychologist, and pacifist who pioneered modern mathematical techniques of weather forecasting, and the application of similar techniques to studying the causes of wars and how to prevent them." Richardson's original iterative scheme (1910) assumed $B = \mu I$, where μ was a nonzero scalar.

Turn now to the application of Newton's method to the linear system (1), where the "nonlinear residual" $\mathbf{F}(\mathbf{x})$ is in fact the linear residual $\mathbf{F}(\mathbf{x}) = A\mathbf{x} - \mathbf{b}$. In this case

$$\mathbf{F}(\mathbf{x})(i) = \sum_{p=1}^{n} a_{ip}\mathbf{x}(p) - \mathbf{b}(i). \tag{6}$$

As indicated above, here we use $\mathbf{x}(p)$ as the components of the vector $\mathbf{x}$. The components of the Jacobian $D\mathbf{F}(\mathbf{x})$ are therefore

$$\frac{\partial \mathbf{F}(\mathbf{x})(i)}{\partial \mathbf{x}(j)} = \sum_{p=1}^{n} a_{ip}\frac{\partial \mathbf{x}(p)}{\partial \mathbf{x}(j)} = \sum_{p=1}^{n} a_{ip}\delta_{pj} = a_{ij},$$

and so $D\mathbf{F}(\mathbf{x}) = A$. For a nonsingular linear system Newton's method is therefore

$$\mathbf{x}_{k+1} = \mathbf{x}_k - [D\mathbf{F}(\mathbf{x}_k)]^{-1}\mathbf{F}(\mathbf{x}_k) = \mathbf{x}_k - A^{-1}(A\mathbf{x}_k - \mathbf{b}) = A^{-1}\mathbf{b},$$

and we see that it converges to the exact solution $\mathbf{x}^\star = A^{-1}\mathbf{b}$ in one step. However, consider replacement of the Jacobian A with the *approximate Jacobian* B^{-1} (recall B was the notation for an approximate inverse of A). Then we arrive at the scheme

$$\mathbf{x}_{k+1} = \mathbf{x}_k - B(A\mathbf{x}_k - \mathbf{b}),$$

namely Richardson iteration! In this sense Richardson iteration is a quasi-Newton method.

3 Jacobi and Gauss-Seidel methods

To find a reasonable approximate inverse B of A, we split $A = L + D + U$ into its strictly lower triangular, diagonal, and upper triangular parts. Take note: the L and U here are not the same L and U which appear in a "$PA = LU$" factorization. Rather, for example, the components of L are $\ell_{ij} = 0$ for $i \le j$ and $\ell_{ij} = a_{ij}$ for $i > j$.

Two simple possibilities are the following.

- *Jacobi method.* $B = D^{-1}$ corresponding to the stationary method

$$\mathbf{x}_{k+1} = \mathbf{x}_k - D^{-1}(A\mathbf{x}_k - \mathbf{b}). \tag{7}$$

- *Gauss-Seidel method.* $B = (L + D)^{-1}$ corresponding to the stationary method

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (L + D)^{-1}(A\mathbf{x}_k - \mathbf{b}). \tag{8}$$

Both iterative schemes are defined only if D is invertible, i.e., all diagonal elements of A are nonzero ($a_{ii} \neq 0$ for $i = 1, 2, \ldots, n$). Since formation of the residual $\mathbf{r} = A\mathbf{x} - \mathbf{b}$ involves a matrix-vector product, generically this formation has $O(n^2)$ cost. Once $\mathbf{r}$ is known, formation of $\mathbf{z} = D^{-1}\mathbf{r}$ involves n divisions, and so $O(n)$ cost. Once $\mathbf{r}$ is known, formation of $\mathbf{z} = (L + D)^{-1}\mathbf{r}$ is tantamount to solution of the lower-triangular linear system $(L + D)\mathbf{z} = \mathbf{r}$ by forward substitution at $O(n^2)$ cost (as we have seen earlier in class). These observation establish the following.

Theorem 1. *Let $A \in \mathbb{R}^{n\times n}$ be a generic full matrix with nonzero diagonal elements, with splitting $A = L + D + U$ into strict lower triangular, diagonal, and upper triangular parts. Then one step of either Jacobi or Gauss-Seidel iterations has $O(n^2)$ cost.*

However, consider the case when A has the special structure of a *sparse matrix* with $O(n)$ nonzero entries as $n \to \infty$. Then formation of $\mathbf{r} = A\mathbf{x} - \mathbf{b}$ has $O(n)$ cost, and one step of either method has $O(n)$ cost (this is immediately evident for Jacobi).

3.1 Jacobi method: further analysis and example

The Jacobi method (7) can be expressed as

$$\begin{aligned} \mathbf{x}_{k+1} &= (I - D^{-1}A)\mathbf{x}_k + D^{-1}\mathbf{b}. \\ &= (I - D^{-1}(L + D + U))\mathbf{x}_k + D^{-1}\mathbf{b}. \\ &= -D^{-1}(L + U)\mathbf{x}_k + D^{-1}\mathbf{b}. \end{aligned} \tag{9}$$

This alternative formula is useful theoretically and in practice. First, consider a nonsingular matrix A which is *diagonally dominant*: for each i we have $|a_{ii}| > \sum_{j\neq i} |a_{ij}|$. Roughly this means the matrix is concentrated on the diagonal. Diagonal dominance implies that the ith row sum of $D^{-1}(L + U)$ in absolute value is

$$\sum_{j=1}^{n} |D^{-1}(L + U)|_{ij} = \sum_{j<i} \frac{|\ell_{ij}|}{|a_{ii}|} + \sum_{j>i} \frac{|u_{ij}|}{|a_{ii}|} = \frac{1}{|a_{ii}|} \sum_{j\neq i} |a_{ij}| < 1.$$

Whence for A diagonally dominant we conclude that $\|I - D^{-1}A\|_\infty = \|D^{-1}(L + U)\|_\infty < 1$. *Jacobi iteration will therefore converge for any choice of initial iterate* $\mathbf{x}_0$.

In terms of the elements of A the last line in (9) is simply

$$\mathbf{x}_{k+1}(i) = \frac{1}{a_{ii}}\Big[\mathbf{b}(i) - \sum_{j\neq i} a_{ij}\mathbf{x}_k(j)\Big]. \tag{10}$$

Consider now the system

$$\begin{pmatrix} 3 & 1 & -1 \\ 2 & 4 & 1 \\ -1 & 2 & 5 \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} 4 \\ 1 \\ 1 \end{pmatrix} \tag{11}$$

which has exact solution $\mathbf{x}^\star = (u^\star, v^\star, w^\star)^T = (2, -1, 1)^T$. Notice here that we use the notation

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}(1) \\ \mathbf{x}(2) \\ \mathbf{x}(3) \end{pmatrix} = \begin{pmatrix} u \\ v \\ u \end{pmatrix}.$$

By inspection, we see that the coefficient matrix in (11) diagonally dominant. *Note that this statement depends on the ordering of the equation; if, for example, we exchanged the first and second equations, the diagonal dominance would be lost!* Rewrite the system (11) as

$$\begin{pmatrix} 3 & & \\ & 4 & \\ & & 5 \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} = -\begin{pmatrix} 0 & 1 & -1 \\ 2 & 0 & 1 \\ -1 & 2 & 0 \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} + \begin{pmatrix} 4 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 4 - v + w \\ 1 - 2u - w \\ 1 + u - 2v \end{pmatrix}.$$

From (10) the Jacobi's method for this system is therefore

$$\begin{aligned} u_{k+1} &= \tfrac{1}{3}(4 - v_k + w_k) \\ v_{k+1} &= \tfrac{1}{4}(1 - 2u_k - w_k) \\ w_{k+1} &= \tfrac{1}{5}(1 + u_k - 2v_k). \end{aligned} \tag{12}$$

Choosing $\mathbf{x}_0 = (0,0,0)^T$, immediately one finds $\mathbf{x}_1 = (\frac{4}{3}, \frac{1}{4}, \frac{1}{5})$, and clearly

$$e_0 = \|\mathbf{x}_0 - \mathbf{x}^\star\|_\infty = 2, \qquad e_1 = \|\mathbf{x}_1 - \mathbf{x}^\star\|_\infty = \|(-\tfrac{2}{3}, \tfrac{5}{4}, -\tfrac{4}{5})^T\|_\infty = \tfrac{5}{4}.$$

Next, we compute

$$\begin{aligned} u_2 &= \tfrac{1}{3}(4 - \tfrac{1}{4} + \tfrac{1}{5}) = \tfrac{79}{60} \\ v_2 &= \tfrac{1}{4}(1 - \tfrac{8}{3} - \tfrac{1}{5}) = -\tfrac{7}{15} \\ w_2 &= \tfrac{1}{5}(1 + \tfrac{4}{3} - \tfrac{1}{2}) = \tfrac{11}{30}. \end{aligned} \tag{13}$$

Now $\mathbf{x}_2 - \mathbf{x}^\star = (-\frac{41}{60}, \frac{8}{15}, -\frac{19}{30})^T$ and $e_2 = \frac{41}{60}$. As expected, $e_0 > e_1 > e_2$.

3.2 Gauss-Seidel method: example and further analysis

We will implement the Gauss-Seidel method for the example just consider via a simple modification of the iterative scheme (12). Only after the fact, will we confirm that the described modification is in fact the Gauss-Seidel iteration introduced above. The key idea is as follows. If we view the updates performed in (12) as carried out top-to-bottom, then why not use the updates already found when forming the right-side That is, use the iteration (performed top-to-bottom!)

$$\begin{aligned} u_{k+1} &= \tfrac{1}{3}(4 - v_k + w_k) \\ v_{k+1} &= \tfrac{1}{4}(1 - 2u_{k+1} - w_k) \\ w_{k+1} &= \tfrac{1}{5}(1 + u_{k+1} - 2v_{k+1}). \end{aligned} \tag{14}$$

The thinking here is that u_{k+1} obtained from the top equation should be a better approximation to $u^\star$ than is u_k. Whence, why not use u_{k+1} in the middle equation (as we already have it from the first equation) rather than u_k. Similarly, use the updated variables in the third equation. For this iteration again with $\mathbf{x}_0 = (0,0,0)^T$ we now have $\mathbf{x}_1 = (\frac{4}{3}, -\frac{5}{12}, \frac{19}{30})$. Therefore, now

$$e_0 = \|\mathbf{x}_0 - \mathbf{x}^\star\|_\infty = 2, \qquad e_1 = \|\mathbf{x}_1 - \mathbf{x}^\star\|_\infty = \|(-\tfrac{2}{3}, \tfrac{7}{12}, -\tfrac{11}{30})^T\|_\infty = \tfrac{2}{3}.$$

One more iteration then yields

$$\begin{aligned} u_2 &= \tfrac{1}{3}(4 + \tfrac{5}{12} + \tfrac{19}{30}) = \tfrac{101}{60} \\ v_2 &= \tfrac{1}{4}(1 - \tfrac{101}{30} - \tfrac{19}{30}) = -\tfrac{3}{4} \\ w_2 &= \tfrac{1}{5}(1 + \tfrac{101}{60} + \tfrac{3}{2}) = \tfrac{251}{300}. \end{aligned} \tag{15}$$

Now $e_2 = \|(-\frac{19}{60}, \frac{1}{4}, -\frac{49}{300})^T\|_\infty = \frac{19}{60}$. Again $e_0 > e_1 > e_2$. Moreover, e_1 and e_2 are smaller than the corresponding errors found with Jacobi iteration.

Starting at the form of (14), we see that this update is

$$\mathbf{x}_{k+1}(i) = \frac{1}{a_{ii}}\Big[\mathbf{b}(i) - (L\mathbf{x}_{k+1})(i) - (U\mathbf{x}_k)(i)\Big]. \tag{16}$$

Despite having $\mathbf{x}_{k+1}$ on both the left and right-hand sides, the iteration can be carried out top-to-bottom. Indeed, for a 3-by-3 system with $\mathbf{x} = (u, v, w)^T$ these equations are (using the strict triangular forms of L and U)

$$\begin{aligned}
u_{k+1} &= \frac{1}{a_{11}}\big[\mathbf{b}(1) - u_{12}v_k - u_{13}w_k\big] = \frac{1}{a_{11}}\big[\mathbf{b}(1) - a_{12}v_k - a_{13}w_k\big]\\
v_{k+1} &= \frac{1}{a_{22}}\big[\mathbf{b}(2) - \ell_{21}u_{k+1} - u_{23}w_k\big] = \frac{1}{a_{22}}\big[\mathbf{b}(2) - a_{21}u_{k+1} - a_{23}w_k\big]\\
w_{k+1} &= \frac{1}{a_{33}}\big[\mathbf{b}(2) - \ell_{31}u_{k+1} - \ell_{32}v_{k+1}\big] = \frac{1}{a_{33}}\big[\mathbf{b}(2) - a_{31}u_{k+1} - a_{32}v_{k+1}\big].
\end{aligned}$$

Clearly, this iteration works when performed top-to-bottom. Notice that (16) is

$$\mathbf{x}_{k+1} = D^{-1}(\mathbf{b} - U\mathbf{x}_k - L\mathbf{x}_{k+1}) \iff D\mathbf{x}_{k+1} = \mathbf{b} - U\mathbf{x}_k - L\mathbf{x}_{k+1}.$$

Further rearrangement then gives

$$(L + D)\mathbf{x}_{k+1} = \mathbf{b} - U\mathbf{x}_k = \mathbf{b} + (L + D - A)\mathbf{x}_k$$

from which we find precisely (8).

4 Successive Over-Relaxation

We stick with the three-variable example system considered in the context of Jacobi and Gauss-Seidel iterations. Introduce a scalar *relaxation parameter* ω (the reason for this terminology becomes clear below) and consider the trivial identity

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = (1-\omega)\begin{pmatrix} u \\ v \\ w \end{pmatrix} + \omega\begin{pmatrix} u \\ v \\ w \end{pmatrix}.$$

Be careful here to distinguish between w, the third component $\mathbf{x}(3)$ of $\mathbf{x}$, and the scalar ω. The iterative method SOR(ω) is

$$\begin{pmatrix} u_{k+1}^{\mathrm{SOR}(\omega)} \\ v_{k+1}^{\mathrm{SOR}(\omega)} \\ w_{k+1}^{\mathrm{SOR}(\omega)} \end{pmatrix} = (1-\omega)\begin{pmatrix} u_k \\ v_k \\ w_k \end{pmatrix} + \omega\begin{pmatrix} u_{k+1}^{\mathrm{GS}} \\ v_{k+1}^{\mathrm{GS}} \\ w_{k+1}^{\mathrm{GS}} \end{pmatrix},$$

where the Gauss-Seidel update formulas appear (multiplied by ω) in the second term on the right-hand side. The idea is that the Gauss-Seidel update is good, so let's try to take more of it! Notice that $\omega = 1$ corresponds to the Gauss-Seidel method. However, if we choose, say,

$\omega = \frac{5}{4}$, then we are "taking more of the Gauss-Seidel update." The update again proceeds top-to-bottom and amounts to the method

$$\begin{pmatrix} u_{k+1} \\ \\ v_{k+1} \\ \\ w_{k+1} \end{pmatrix} = (1-\omega)\begin{pmatrix} u_k \\ \\ v_k \\ \\ w_k \end{pmatrix} + \omega \begin{pmatrix} \frac{1}{3}(4 - v_k + w_k) \\ \\ \frac{1}{4}(1 - 2u_{k+1} - w_k) \\ \\ \frac{1}{5}(1 + u_{k+1} - 2v_{k+1}) \end{pmatrix}. \tag{17}$$

Remarkably, this strategy can improve convergence, with SOR(ω) out-performing Jacobi and Gauss-Seidel in some instances.

SOR(ω) can also be interpreted as Richardson iteration, but it takes some algebra to see this. For a general system the SOR(ω) is

$$\mathbf{x}_{k+1} = (1-\omega)\mathbf{x}_k + \omega D^{-1}(\mathbf{b} - U\mathbf{x}_k - L\mathbf{x}_{k+1}) \tag{18}$$

Multiplication of both sides by D, followed by rearrangement of terms, then yields

$$(D+\omega L)\mathbf{x}_{k+1} = (1-\omega)D\mathbf{x}_k + \omega(\mathbf{b} - U\mathbf{x}_k) = [(1-\omega)D - \omega U]\mathbf{x}_k + \omega\mathbf{b}. \tag{19}$$

Notice that $(1-\omega)D - \omega U = D - \omega(D+U) = D + \omega L - \omega A$. Substitution of this result into the previous numbered equation gives

$$(D+\omega L)\mathbf{x}_{k+1} = (D+\omega L)\mathbf{x}_k - \omega(A\mathbf{x}_k - \mathbf{b}). \tag{20}$$

Finally, provided that the diagonal elements of A are nonzero, the matrix $D+\omega L$ is invertible and

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k - \omega(D+\omega L)^{-1}(A\mathbf{x}_k - \mathbf{b}) \\ &= \big[I - \omega(D+\omega L)^{-1}A\big]\mathbf{x}_k + \omega(D+\omega L)^{-1}\mathbf{b}. \end{aligned} \tag{21}$$

This is Richard iteration with approximate inverse $B = \omega(D+\omega L)^{-1}$.

5 Numerical tests

For the presented example, we implement the three classical methods in the following script.

```
% script: Jacobi_GS_SOR_compare
% UNM CS-Math 375, Fall 2022

A = [  3    1   -1    ;           % coefficient matrix
       2    4    1    ;
      -1    2    5    ];
%
b = [  4  ;                       % right-hand side
       1  ;
       1  ];
%
xexact = [  2  ;                  % exact solution
           -1  ;
```

```
            1  ];

kmax = 15;                          % number of iterations to perform

%  Jacobi method
uold = 0; vold = 0; wold = 0;
for k = 1:kmax
    u =  (4 -   vold +   wold)/3;
    v =  (1 - 2*uold -   wold)/4;
    w =  (1 +   uold - 2*vold)/5;
    uold = u;
    vold = v;
    wold = w;
end
JacobiErr = num2str(norm([u; v; w] - xexact,inf),'%0.4e');
disp([char(10),'Jacobi inf-norm error       : ',JacobiErr])

%  Gauss-Seidel method
u = 0; v = 0; w = 0;
for k = 1:kmax
    u =  (4 -   v +   w)/3;
    v =  (1 - 2*u -   w)/4;
    w =  (1 +   u - 2*v)/5;
end
GSErr = num2str(norm([u; v; w] - xexact,inf),'%0.4e');
disp(['Gauss-Seidel inf-norm error : ',GSErr])

%  SOR(omega)
omega = 1.25;
uold = 0; vold = 0; wold = 0;
for k = 1:kmax
    u = (1-omega)*uold + (omega/3)*(4 -   vold +   wold);
    v = (1-omega)*vold + (omega/4)*(1 -    2*u -   wold);
    w = (1-omega)*wold + (omega/5)*(1 +      u -    2*v);
    uold = u;
    vold = v;
    wold = w;
end
SORErr=num2str(norm([u; v; w] - xexact,inf),'%0.4e');
disp(['SOR(omega) inf-norm error   : ',SORErr])
```

With `kmax = 5`, the output of the script is as follows.

```
octave:1> Jacobi_GS_SOR_compare

Jacobi inf-norm error       : 2.1111e-01
Gauss-Seidel inf-norm error : 2.6452e-02
SOR(omega) inf-norm error   : 1.5381e-03
```

With `kmax = 15`, the output of the script is as follows.

```
octave:2> Jacobi_GS_SOR_compare
```

```
Jacobi inf-norm error       : 4.0525e-03
Gauss-Seidel inf-norm error : 6.8813e-06
SOR(omega) inf-norm error   : 1.4901e-09
```

The following script explores the optimal choice of relaxation parameter ω for the system.

```
% script: OptimalOmegaForSOR
% CS-Math 375, Fall 2022
A = [3 1 -1; 2 4 1; -1 2 5];
b = [4; 1; 1];
xexact = [2; -1; 1];
kmax = 6;
omgs = [1:0.01:1.5];
errs = zeros(size(omgs));
disp(['omega    2-norm error'])
for p = 1:length(omgs)
    omega = omgs(p);
    uold = 0; vold = 0; wold = 0;
    for k = 1:kmax
        u = (1-omega)*uold + (omega/3)*(4 -   vold +   wold);
        v = (1-omega)*vold + (omega/4)*(1 -    2*u -   wold);
        w = (1-omega)*wold + (omega/5)*(1 +      u -    2*v);
        uold = u;
        vold = v;
        wold = w;
    end
    errs(p) = norm([u; v; w] - xexact,inf);
    disp([num2str(omega,'%0.4f'), '   ',num2str(errs(p),'%0.5e')])
end
explot(omgs,errs,'o')
title('\infty-norm error after 6 steps of SOR(\omega)')
xlabel('\omega')
saveas(gcf,'SORerrors.eps','eps')
```

Figure 1 shows the plot outputted by the script. Notice the $\omega = 1.2 \in (1, 2)$ is close to the optimal choice of ω, that is, the choice which yields the fastest decay in the error as the iteration proceeds.

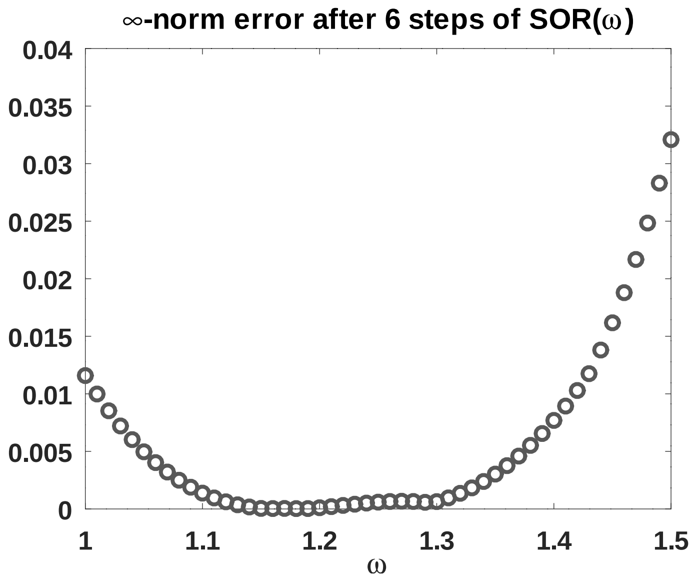

Figure 1: Optimal choice of ω.