Assignment 7: Iterative methods

Gregory Smetana
ID 1917370
ACM 106a

December 3, 2013

1 Gauss-Seidel as convex optimization

In this exercise, we will prove the following:

Suppose that the matrix $A \in R^{n\times n}$ is symmetric positive definite. Then the Gauss-Seidel method applied to the linear system $Ax = b$ converges to the solution $x = A^{-1}b$.

a)

In this part, we show that if $A \in \mathbf{R}^{n\times n}$ is symmetric positive definite, then the solution $\overline{x} = A^{-1}b$ is the unique minimizer of the unconstrained convex program

$$\min_{x\in\mathbf{R}^n} f(x) := \min_{x\in\mathbf{R}^n} \frac{1}{2}x^T Ax - b^T x \tag{1}$$

For unconstrained problems, x is a global minimizer if and only if $\nabla f(x) = 0$.

$$\begin{aligned}
[\nabla f(x)]_i &= \frac{\partial f}{\partial x_i} \\
&= \frac{\partial}{\partial x_i}\left(\frac{1}{2}x_j A_{jk} x_k - b_k x_k\right) \\
&= \frac{1}{2}\left(\delta_{ij} A_{jk} x_k + x_j A_{jk}\delta_{ik}\right) - b_k\delta_{ik} \\
&= \frac{1}{2}\left(A_{ik}x_k + x_j A_{ji}\right) - b_i \\
&= A_{ij}x_j - b_i
\end{aligned} \tag{2}$$

$$\nabla f(x) = Ax - b \tag{3}$$

where we have used the symmetry of A. Now, considering the solution $x = A^{-1}b$,

$$\begin{aligned}\nabla f(x) &= A(A^{-1}b) - b \\ &= 0\end{aligned} \tag{4}$$

so we have shown that $x = A^{-1}b$ is a minimizer of the unconstrained convex program of Equation 1. To prove uniqueness, assume we have $f(\overline{x}) = f(\overline{x} + d)$ for some $d \neq 0$

$$\begin{aligned}\overline{x}^T \frac{A}{2}\overline{x} - b^T\overline{x} &= (\overline{x}+d)^T \frac{A}{2}(\overline{x}+d) - b^T(\overline{x}+d) \\ \overline{x}^T \frac{A}{2}\overline{x} - b^T\overline{x} &= \overline{x}^T \frac{A}{2}\overline{x} + \overline{x}^T\frac{A}{2}d + d^T\frac{A}{2}\overline{x} + d^T\frac{A}{2}d - b^T\overline{x} - b^Td \\ 0 &= \overline{x}^T\frac{A}{2}d + d^T\frac{A}{2}\overline{x} + d^T\frac{A}{2}d - b^Td\end{aligned} \tag{5}$$

Now substituting in $b = A\overline{x}$

$$0 = \overline{x}^T\frac{A}{2}d + d^T\frac{A}{2}\overline{x} + d^T\frac{A}{2}d - (A\overline{x})^Td \tag{6}$$

Using the symmetry of A,

$$0 = \overline{x}^TAd + d^T\frac{A}{2}d - \overline{x}^TAd \tag{7}$$

$$\frac{1}{2}d^TAd = 0 \tag{8}$$

Recall the definition of a positive definite matrix:

$$x^TAx > 0 \qquad \forall x \neq 0 \tag{9}$$

So Equation 8 contradicts our assumption of $d \neq 0$. Therefore, $\overline{x}$ is unique.

b)

Suppose we greedily choose the step length to maximize decrease in objective value:

$$\alpha_i = {}^{\arg}\underset{\alpha}{\min} f(x + \alpha e_i) \tag{10}$$

Expanding,

$$\begin{aligned}\alpha_i &= {}^{\arg}\underset{\alpha}{\min}(x+\alpha e_i)^T\frac{A}{2}(x+\alpha e_i) - b^T(x-\alpha e_i) \\ &= {}^{\arg}\underset{\alpha}{\min}\ x^T\frac{A}{2}x + \alpha x^T\frac{A}{2}e_i + \alpha e_i^T\frac{A}{2}x + \alpha^2 e_i^T\frac{A}{2}e_i - b^Tx - \alpha b^Te_i\end{aligned} \tag{11}$$

A is a positive definite matrix, so setting the first derivative with respect to α to zero will give the minimum value

$$0 = x^T \frac{A}{2} e_i + e_i^T \frac{A}{2} x + \alpha_i e_i^T A e_i - b^T e_i \tag{12}$$

writing out indicies,

$$0 = x_j \frac{A_{ji}}{2} + \frac{A_{ij}}{2} x_j + \alpha_i A_{ii} - b_i \tag{13}$$

$$\alpha_i = \frac{1}{A_{ii}} \left(b_i - x_j A_{ji} \right) \tag{14}$$

$$x^{new} = x + \frac{1}{A_{ii}} \left(b_i - A_{ij} x_j \right) \tag{15}$$

$$\boxed{x^{new} = \frac{1}{A_{ii}} \left(b_i - \sum_{j \neq i} A_{ij} x_j \right)} \tag{16}$$

The update formula for Gauss-Seidel is

$$x_i^{k+1} = \frac{1}{A_{ii}} \left(b_i - \sum_{j<i} A_{ij} x_j^{k+1} - \sum_{j>i} A_{ij} x_j^k \right) \tag{17}$$

In Gauss-Seidel iteration, the entries of x are updated one by one. If you overwrite the old entries using the new entry at each step, the updating formula is

$$\boxed{x^{new} = \frac{1}{A_{ii}} \left(b_i - \sum_{j \neq i} A_{ij} x_j \right)} \tag{18}$$

And we see that the optimization approach uses the same operation to update x_i as Gauss-Seidel.

c)

Suppose that $x = \overline{x} + e$ is an approximation of $\overline{x} = A^{-1}b$. In this part, we show that the A-norm defined by

$$\|e\|_A = \sqrt{e^T A e} \tag{19}$$

of the error e satisfies

$$|f(x) - f(\overline{x})| = \frac{1}{2} \|e\|_A^2 \tag{20}$$

Starting with the left hand side,

$$|f(x) - f(\overline{x})| = \frac{1}{2}x^T Ax - b^T x - \frac{1}{2}\overline{x}^T A\overline{x} + b^T\overline{x} \tag{21}$$

Now substituting in $b = A\overline{x}$ and using symmetry of A,

$$\begin{aligned}|f(x) - f(\overline{x})| &= \frac{1}{2}x^T Ax - (A\overline{x})^T x - \frac{1}{2}\overline{x}^T A\overline{x} + (A\overline{x})^T\overline{x} \\ &= \frac{1}{2}x^T Ax - \overline{x}^T Ax - \frac{1}{2}\overline{x}^T A\overline{x} + \overline{x}^T A\overline{x} \\ &= \frac{1}{2}\left(x^T Ax - 2\overline{x}^T Ax + \overline{x}^T A\overline{x}\right) \\ &= \frac{1}{2}\left(x^T Ax - \overline{x}^T Ax - x^T A\overline{x} + \overline{x}^T A\overline{x}\right) \\ &= \frac{1}{2}(x - \overline{x})^T A(x - \overline{x}) \\ &= \frac{1}{2}e^T Ae \\ &= \frac{1}{2}\|e\|_A^2\end{aligned} \tag{22}$$

Therefore, it has been shown that

$$\boxed{|f(x) - f(\overline{x})| = \frac{1}{2}\|e\|_A^2} \tag{23}$$

Completing the proof: Parts (a), (b), and (c) in tandem prove the theorem. Indeed, Parts (a) and (b) establish that the function value of f decreases during each Gauss-Seidel step (because it decreases during each cycle of coordinate descent unless we have already converged to the solution) and that this value of f is bounded below by the optimal value $f(\overline{x})$. Because these function values form a monotonically decreasing x sequence with infimum $f(\overline{x})$, Part (c) implies that the sequence of errors converge to 0 (with respect to the A-norm) or, equivalently, that the sequence of iterates $\{x^{(k)}\}$ generated by Gauss-Seidel converges to $\overline{x}$

2 Early convergence of Arnoldi iteration

Suppose that at the nth step of Arnoldi iteration (applied to the matrix $A \in C^{m\times m}$ with vector $b \in C^m$) we obtain $H_{n+1,n} = 0$ in the recurrence relation

$$AQ_n = Q_{n+1}\tilde{H}_n \tag{24}$$

The following exercises will establish that we can stop the Arnoldi iteration after this step; that is, we've found a basis for all Krylov subspaces of A generated by b, and the maximal such subspace contains our desired matrix equation solutions.

a)

We let $H_n = H(1:n, 1:n)$, $\tilde{H}_n = H(1:n+1, 1:n)$, and $Q_n = [q_1, q_2, ..., q_n]$ be the first n columns of Q. Then Equation 24 may be rewritten

$$AQ_n = Q_n H_n + q_{n+1} H_{n+1,n} e_n^T \tag{25}$$

where $e_n^T = [0, 0, ..., 0, 1] \in \mathbf{R}^n$. In the case of $H_{n+1,n} = 0$, the equation is

$$\boxed{AQ_n = Q_n H_n} \tag{26}$$

b)

Using Part (a), we now show that $\mathcal{K}_n$ is an invariant subspace of A, i.e., $A\mathcal{K}_n \subseteq \mathcal{K}_n$.

$$\mathcal{K}_n = \text{span}\{q_1, q_2, ..., q_n\} \tag{27}$$

$$\begin{aligned} A\mathcal{K}_n &= \text{span}\{Aq_1, Aq_2, ..., Aq_n\} \\ &= \text{span}\{AQ_n\} \end{aligned} \tag{28}$$

It was shown in Part (a) that $AQ_n = Q_n H_n$

$$\begin{aligned} Q_n H_n(:, i) &= Q_n \begin{bmatrix} H_{1i} \\ \vdots \\ H_{ni} \end{bmatrix} \\ &= [q_1, ..., q_n] \begin{bmatrix} H_{1i} \\ \vdots \\ H_{ni} \end{bmatrix} \\ &= H_{1i} q_1 + ... + H_{ni} q_n \end{aligned} \tag{29}$$

which is a linear combination of the vectors $\{q_1, q_2, ..., q_n\}$. Therefore, $A\mathcal{K}_n = \text{span}\{q_1, q_2, ..., q_n\}$ and it has been shown that

$$\boxed{A\mathcal{K}_n \subseteq \mathcal{K}_n} \tag{30}$$

c)

Using Part (b), we now show that we have $\mathcal{K}_n = \mathcal{K}_{n+1} = ...$

It was shown in Part (b) that $A\mathcal{K}_n \subseteq \mathcal{K}_n$, where

$$\mathcal{K}_n = \text{span}\{b, Ab, ..., A^{n-1}b\} \tag{31}$$

$$\mathcal{K}_{n+1} = \text{span}\{b, Ab, ..., A^n b\} \tag{32}$$

Examining the last element

$$A^n b = A(A^{n-1}b) \tag{33}$$

$$A^{n-1}b \subseteq \mathcal{K}_n \tag{34}$$

so

$$A^n b \subseteq A\mathcal{K}_n = \mathcal{K}_n \tag{35}$$

Therefore, we have

$$\boxed{\mathcal{K}_n = \mathcal{K}_{n+1} = ...} \tag{36}$$

d)

Suppose that λ is an eigenvalue of H_n with corresponding eigenvector v

$$H_n v = \lambda v \tag{37}$$

$$Q_n H_n v = \lambda Q_n v \tag{38}$$

$$A_n Q_n v = \lambda Q_n v \tag{39}$$

$$A_n w = \lambda w \tag{40}$$

Therefore, $\underline{\lambda \text{ is also an eigenvalue of } A}$, with corresponding eigenvalue

$$\boxed{w = Q_n v} \tag{41}$$

e)

For any nonsingular matrix $A \in \mathbf{C}^{m\times m}$, it is known that we may decompose its inverse as the matrix polynomial

$$A^{-1} = \sum_{k=0}^{m-1} c_k A^k \tag{42}$$

for some scalars $c_0, c_1, ..., c_{m-1}$.

$$x = A^{-1}b = \sum_{k=0}^{m-1} c_k A^k b \tag{43}$$

$$\text{span}(x) = \{b, Ab, A^2b, ..., A^{m-1}b\} = \mathcal{K}_m \tag{44}$$

In part c), it was shown that $\mathcal{K}_m = \mathcal{K}_n$. Therefore, $\underline{\text{the solution of } Ax = b \text{ belongs to } \mathcal{K}_n}$

3 Practical convergence of Gauss-Seidel and Jacobi

Consider the linear system $Ax = b$ defined by

$$A = \begin{pmatrix} 3 & -5 & 2 \\ 5 & 4 & 3 \\ 2 & 5 & 3 \end{pmatrix} \qquad b = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \tag{45}$$

a)

The spectral radius of a matrix A is defined as

$$\rho(A) = \max\{|\lambda| : \lambda \text{ is an eigenvalue of } A\} \tag{46}$$

The update matrix of the Jacobi method is

$$R_J = D^{-1}(\tilde{L} + \tilde{U}) \tag{47}$$

The update matrix of Gauss-Seidel is

$$R_{GS} = (D - \tilde{L})^{-1}\tilde{U} \tag{48}$$

where we adopt the Demmel notation of

$$A = D - \tilde{L} - \tilde{U} \tag{49}$$

where $-\tilde{L}$ is the strictly lower triangular part of A and $-\tilde{U}$ is the strictly upper triangular part of A.

Computing the spectral radius of the update matrices corresponding with the system of Equation 45,

$$\boxed{\rho(R_J) = 0.913} \tag{50}$$

$$\boxed{\rho(R_{GS}) = 2.163} \tag{51}$$

A splitting method converges if and only if the update matrix has spectral radius satisfying

$$\rho(R) < 1 \tag{52}$$

Therefore, we expect that the Jacobi method with converge and the Gauss-Seidel method will diverge. The Gauss-Seidel method will not be useful, but the Jacobi method may be used to solve the linear system.

b)

Four Matlab functions were written implementing Jacobi and Gauss-Seidel methods using for loops and update matrices. The inputs are A, b, initial iterate $x^{(0)}$, and the relative residual error tolerance. They are attached in the Appendix.

c)

Each of the codes was used to solve the system $Ax = b$ with $x^{(0)} = (0, 0, 0)$, $tol = 10^{-3}$, and a maximum of 100 iterations. A aggregate time was calculated using "tic/toc" over 1000 experiment runs. The results are displayed in Table 1.

	Relative error	Number of iterates	Total time [s]
Jacobi Matrix	1.4306e-03	77	3.3907e-01
Jacobi Sequential	1.4306e-03	77	4.0913e-01
Gauss-Seidel Matrix	7.4055e+32	100	4.3540e-01
Gauss-Seidel Sequential	7.4055e+32	100	3.7104e-01

Table 1

The convergence phenomena matches that predicted in Part (a). The Jacobi method converges to the correct solution, while the Gauss-Seidel method diverges. It was found that the Jacobi method using matrices and the Gauss-Seidel using for loops were the fastest. It must be that the time required to assemble the Gauss-Seidel update matrix and store it in memory was a large component of the algorithm run time. For most problems, it is much quicker to use matrix-vector multiplication (optimized by Matlab) than for-loops.

d)

The experiment was repeated for a few randomly chosen values of b and $x^{(0)}$. The observed convergence phenomena did not change, with the Jacobi method converging and the Gauss-Seidel method diverging. These results coincide with that predicted by theory. The convergence of a splitting method depends only on the update matrix R, not on the value of b or the initial iterate.

A jacobi_mat.m

```
%Smetana_Gregory_1917370_A6_P3
function [ x, k ] = jacobi_mat( A, b, x0, tol, maxiterations )
%JACOBI_MAT jacobi method using matrices
n = length(b);

%% calculate update matrix
D = diag(A);
Lt = -tril(A,-1);
Ut = -triu(A,1);
Rj = diag(1./D) * (Lt + Ut);

c = diag(1./D) * b;

%rhoJ = max(abs(eig(Rj))) % output spectral radius

k = 0;
err = 1;
x = x0;
while k < maxiterations && err > tol
    x = Rj*x + c;
    err = norm(A*x - b)/norm(b);
    k=k+1;
end
```

B jacobi_seq.m

```
%Smetana_Gregory_1917370_A6_P3
function [ x, k ] = jacobi_seq( A, b, x0, tol, maxiterations )
%JACOBI_SEQ jacobi method using for loops


n = length(b);

k = 0;
err = 1;
x = x0;

while k < maxiterations && err > tol
  x_new = zeros ( n, 1 );
  for i = 1 : n
    sigma =0;
    for j = 1:n
        if j ~= i
            sigma = sigma + A(i,j)*x(j);
        end
    end
    x_new(i) = (b(i)-sigma)/A(i,i);
  end
  x = x_new;
  err = norm(A*x - b)/norm(b);
  k=k+1;
end
```

C gs_mat.m

```
function [ x, k ] = gs_mat( A, b, x0, tol, maxiterations )
%GS_MAT gauss-seidel method using matrices
n = length(b);

%% calculate update matrix
D = diag(A);
Lt = -tril(A,-1);
Ut = -triu(A,1);
Rgs = (diag(D) - Lt)\Ut;
c = (diag(D) - Lt)\ b;

%rhoGS = max(abs(eig(Rgs))) % output spectral radius

k = 0;
err = 1;
x = x0;
while k < maxiterations && err > tol
    x = Rgs*x + c;
    err = norm(A*x - b)/norm(b);
    k=k+1;
end
```

D gs_seq.m

```
function [ x, k ] = gs_seq( A, b, x0, tol, maxiterations )
%GS_SEQ gauss-seidel method using for loops

n = length(b);

k = 0;
err = 1;
x = x0;

while k < maxiterations && err > tol
  for i = 1 : n
```

```
    sigma =0;
    for j = 1:n
        if j ~= i
            sigma = sigma + A(i,j)*x(j);
        end
    end
    x(i) = (b(i)-sigma)/A(i,i);
  end
  err = norm(A*x - b)/norm(b);
  k=k+1;
end
```

E Smetana_Gregory_1917370_A7_P3_DIARY.txt

```
run('Smetana_Gregory_1917370_A7_P3.m');

tjmat =

   3.3907e-01


kjmat =

    77


xjmat_err =

   1.4306e-03


tjseq =

   4.0913e-01


kjseq =

    77


xjseq_err =

   1.4306e-03


tgsmat =
   4.3540e-01


kgsmat =

   100


xgsmat_err =

   7.4055e+32


tgsseq =

   3.7104e-01


kgsseq =

   100


xgsseq_err =

   7.4055e+32


b =

     2
     9
     3


x0 =

     6
     4
     3


xjmat_err =

   1.6049e-03


xjseq_err =

   1.6049e-03
```

```
xgsmat_err =

   7.5550e+33


xgsseq_err =

   7.5550e+33


b =

     4
     7
     1


x0 =

     5
     0
     2


xjmat_err =

   1.5330e-03


xjseq_err =

   1.5330e-03


xgsmat_err =

   2.4590e+32


xgsseq_err =

   2.4590e+32


b =

     7
     9
     8


x0 =

     2
     5
     6


xjmat_err =

   1.9766e-03


xjseq_err =

   1.9766e-03


xgsmat_err =

   9.9964e+33


xgsseq_err =

   9.9964e+33

diary off
```

F Smetana_Gregory_1917370_A6_P3.m

```
%Smetana_Gregory_1917370_A6_P3
clear;
clc;
close all;
path(path,'export_fig/');

A = [3, -5, 2;
    5, 4, 3;
    2, 5, 3 ];

b = [1;1;1];

xc = A\b;

x0=[0;0;0];
```

```
tol = 1E-3;
maxiterations = 100;

tic
for i = 1:1000
    [xjmat, kjmat] = jacobi_mat(A,b,x0, tol, maxiterations);
end
tjmat = toc
kjmat
xjmat_err = norm(xc - xjmat)/norm(xc)

tic
for i = 1:1000
    [xjseq, kjseq] = jacobi_seq(A,b,x0, tol, maxiterations);
end
tjseq = toc
kjseq
xjseq_err = norm(xc- xjseq)/norm(xc)

tic
for i = 1:1000
    [xgsmat, kgsmat] = gs_mat(A,b,x0, tol, maxiterations);
end
tgsmat = toc
kgsmat
xgsmat_err = norm(xc-xgsmat)/norm(xc)

tic
for i = 1:1000
    [xgsseq, kgsseq] = gs_seq(A,b,x0, tol, maxiterations);
end
tgsseq = toc
kgsseq
xgsseq_err = norm(xc - xgsseq)/norm(xc)

b = [2;9;3]
xc = A\b;
x0=[6;4;3]


[xjmat, kjmat] = jacobi_mat(A,b,x0, tol, maxiterations);
xjmat_err = norm(xc - xjmat)/norm(xc)

[xjseq, kjseq] = jacobi_seq(A,b,x0, tol, maxiterations);
xjseq_err = norm(xc- xjseq)/norm(xc)

[xgsmat, kgsmat] = gs_mat(A,b,x0, tol, maxiterations);
xgsmat_err = norm(xc-xgsmat)/norm(xc)

[xgsseq, kgsseq] = gs_seq(A,b,x0, tol, maxiterations);
```

```
xgsseq_err = norm(xc - xgsseq)/norm(xc)

b = [4;7;1]
xc = A\b;
x0=[5;0;2]


[xjmat, kjmat] = jacobi_mat(A,b,x0, tol, maxiterations);
xjmat_err = norm(xc - xjmat)/norm(xc)

[xjseq, kjseq] = jacobi_seq(A,b,x0, tol, maxiterations);
xjseq_err = norm(xc- xjseq)/norm(xc)

[xgsmat, kgsmat] = gs_mat(A,b,x0, tol, maxiterations);
xgsmat_err = norm(xc-xgsmat)/norm(xc)

[xgsseq, kgsseq] = gs_seq(A,b,x0, tol, maxiterations);
xgsseq_err = norm(xc - xgsseq)/norm(xc)

b = [7;9;8]
xc = A\b;
x0=[2;5;6]


[xjmat, kjmat] = jacobi_mat(A,b,x0, tol, maxiterations);
xjmat_err = norm(xc - xjmat)/norm(xc)

[xjseq, kjseq] = jacobi_seq(A,b,x0, tol, maxiterations);
xjseq_err = norm(xc- xjseq)/norm(xc)

[xgsmat, kgsmat] = gs_mat(A,b,x0, tol, maxiterations);
xgsmat_err = norm(xc-xgsmat)/norm(xc)

[xgsseq, kgsseq] = gs_seq(A,b,x0, tol, maxiterations);
xgsseq_err = norm(xc - xgsseq)/norm(xc)
```