

**1. Problem 11, p. 248**

Let  $A \in \mathbb{R}^{n \times n}$  with  $A^T = A$  and  $\mathbf{x}^T A \mathbf{x} > 0 \Leftrightarrow \mathbf{x} \neq \mathbf{0}$ . Suppose we have vectors  $(\mathbf{v}_i)_{i=1}^n \subset \mathbb{R}^n \setminus \{\mathbf{0}\}$  such that  $\mathbf{v}_i^T A \mathbf{v}_j = 0$  if  $i \neq j$ .

We need to show that in this situation, if we are given a bunch of  $(c_i)_{i=1}^n \subset \mathbb{R}$  and they happen to fulfill

$$\sum_{i=1}^n c_i \mathbf{v}_i = \mathbf{0},$$

then necessarily  $c_i = 0$  for  $i = 1, \dots, n$ . (This the same as saying “the  $\mathbf{v}_i$  are linearly independent” or “the  $\mathbf{v}_i$  form a basis of  $\mathbb{R}^n$ ”.)

**Proof.** Suppose we are given  $c_i$  and  $\mathbf{v}_i$  that satisfy the above. Then necessarily

$$\begin{aligned} 0 &= \left( \sum_{i=1}^n c_i \mathbf{v}_i \right)^T A \underbrace{\left( \sum_{j=1}^n c_j \mathbf{v}_j \right)}_{=0 \text{ by assumption}} \\ &= \sum_{i,j=1}^n c_i c_j \underbrace{\mathbf{v}_i^T A \mathbf{v}_j}_{=0 \text{ if } i \neq j} \\ &= \sum_{i=1}^n c_i c_i \mathbf{v}_i^T A \mathbf{v}_i \\ &= \sum_{i=1}^n \underbrace{c_i^2}_{\geq 0} \underbrace{\mathbf{v}_i^T A \mathbf{v}_i}_{> 0}. \end{aligned}$$

If you assume that even one  $c_i \neq 0$  in the last expression, then necessarily the whole expression is greater than zero, contradicting our assumption. Thus all the  $c_i = 0$ , which shows our claim.  $\square$

**2. Problem 12, p. 249**

Code:

```
function p2
    A = [...
        7,-3,0,0,0;...
        -3,9,1,0,0;...
        0,1,3,-1,0;...
        0,0,-1,10,4;...
        0,0,0,4,6];

    b = [4;-6;3;7;2];

    disp('Problem 7.')
    disp('-----')
    run_comparison('p7', A, b, 1.0923)

    disp('')
    disp('Problem 8.')
    disp('-----')
    A = [...
        4,-1,0,-1,0,0;...
        -1,4,-1,0,-1,0;...
```

```

0,-1,4,0,0,-1;...
-1,0,0,4,-1,0;...
0,-1,0,-1,4,-1;...
0,0,-1,0,-1,4];

b = [-1;0;1;-2;1;2];

run_comparison('p8', A, b, 1.1128)
end
% -----
function run_comparison(id, A, b, sor_optimal_omega)
A
b
tol = 5e-7;
[x_steepest, itcount_steepest, hist_steepest] = it_steepest(A, b, tol);
[x_cg, itcount_cg, hist_cg] = it_cg(A, b, tol);
[x_jacobi, itcount_jacobi, hist_jacobi] = it_jacobi(A, b, tol);
[x_gs, itcount_gs, hist_gs] = it_gauss_seidel(A, b, tol);
[x_sor, itcount_sor, hist_sor] = it_sor(A, b, tol, sor_optimal_omega);

% make sure we got roughly the same answer using every method
x = A\b;

assert(abs(x-x_steepest)<1e-5)
assert(abs(x-x_cg)<1e-5)
assert(abs(x-x_jacobi)<1e-5)
assert(abs(x-x_gs)<1e-5)
assert(abs(x-x_sor)<1e-5)

fprintf('Steepest descent took %3d iterations.\n', itcount_steepest);
fprintf('CG took %3d iterations.\n', itcount_cg);
fprintf('SOR took %3d iterations.\n', itcount_sor);
fprintf('Gauss-Seidel took %3d iterations.\n', itcount_gs);
fprintf('Jacobi took %3d iterations.\n', itcount_jacobi);

if (isempty('OCTAVE_VERSION'))
    clf;
    hold on;
    semilogy(1:numel(hist_steepest), hist_steepest, 'r-;steepest;')
    semilogy(1:numel(hist_cg), hist_cg, 'g-;cg;')
    semilogy(1:numel(hist_sor), hist_sor, 'b-;sor;')
    semilogy(1:numel(hist_gs), hist_gs, 'm-;gs;')
    semilogy(1:numel(hist_jacobi), hist_jacobi, 'c-;jacobi;')
    print([id '.eps'], '-color');
end
end
% -----
function assert(p)
not_p = ~p;
if (sum(abs(not_p)) ~= 0)
    error('Assertion violated.')
end
end
% -----
function [x, itcount, resid_hist] = it_steepest(A, b, tol)
[n,dummy] = size(A);

```

```

x = zeros(n,1);

r = A*x - b;

itcount = 0;
resid_hist = [];

while 1
    d= -r;
    u = A*d; % this is where we spend most of our processing time
    itcount = itcount + 1; % that's why we increment the iteration count here

    lambda = -d'*r/(d'*u);
    x = x + lambda*d;
    r = r + lambda*u;

    resid_norm = norm(r, 2);
    resid_hist(end+1) = resid_norm;
    if (resid_norm<tol)
        return
    end
end
end
% -----
function [x, itcount, resid_hist] = it_cg(A, b, tol)
    [n,dummy] = size(A);
    x = zeros(n,1);

    r = A*x - b;
    d = -r;
    delta = r'*r;

    itcount = 0;
    resid_hist = [];

    while 1
        u = A*d; % this is where we spend most of our processing time
        itcount = itcount + 1; % that's why we increment the iteration count here

        lambda = delta/(d'*u);
        x = x + lambda*d;
        r = r + lambda*u;
        next_delta = r'*r;

        resid_hist(end+1) = sqrt(next_delta);
        if (sqrt(next_delta)<tol)
            return
        end

        alpha = next_delta/delta;
        delta = next_delta;
        d = -r + alpha*d;
    end
end
% -----
function [x, itcount, resid_hist] = it_jacobi(A, b, tol)

```

```

[dummy, n] = size(A);

L = -tril(A, -1);
U = -triu(A, 1);
D = diag(diag(A));

T = D\(L+U);
c = D\b;

[x, itcount, resid_hist] = run_iteration(A, b, T, c, zeros(n,1), tol);
end
% -----
function [x, itcount, resid_hist] = it_gauss_seidel(A, b, tol)
[dummy, n] = size(A);

L = -tril(A, -1);
U = -triu(A, 1);
D = diag(diag(A));

T = (D-L)\U;
c = (D-L)\b;

[x, itcount, resid_hist] = run_iteration(A, b, T, c, zeros(n,1), tol);
end
% -----
function [x, itcount, resid_hist] = it_sor(A, b, tol, omega)
[dummy, n] = size(A);

L = -tril(A, -1);
U = -triu(A, 1);
D = diag(diag(A));

T = (1/omega*D-L)\((1/omega-1)*D+U);
c = (1/omega*D-L)\b;

[x, itcount, resid_hist] = run_iteration(A, b, T, c, zeros(n,1), tol);
end
% -----
function [x, itcount, resid_hist] = run_iteration(A, b, T, c, xstart, tol)
x = xstart;
last_stepsize = 2*tol;

itcount = 0;

resid_hist = [];

while last_stepsize > tol
    last_x = x;
    x = T*x + c;
    last_stepsize = norm(x-last_x, 2);
    itcount = itcount + 1;

    resid_norm = norm(A*x-b);
    resid_hist(end+1) = resid_norm;
end
end

```

Results:

Problem 7.

-----  
A =

7	-3	0	0	0
-3	9	1	0	0
0	1	3	-1	0
0	0	-1	10	4
0	0	0	4	6

b =

4  
-6  
3  
7  
2

Steepest descent took 33 iterations.  
CG took 5 iterations.  
SOR took 9 iterations.  
Gauss-Seidel took 14 iterations.  
Jacobi took 27 iterations.

Problem 8.

-----  
A =

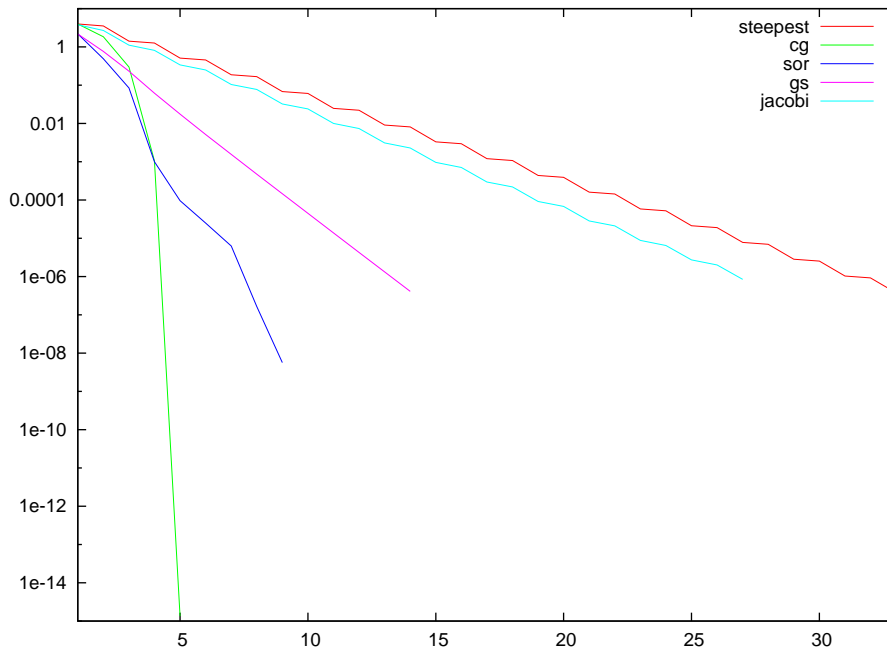
4	-1	0	-1	0	0
-1	4	-1	0	-1	0
0	-1	4	0	0	-1
-1	0	0	4	-1	0
0	-1	0	-1	4	-1
0	0	-1	0	-1	4

b =

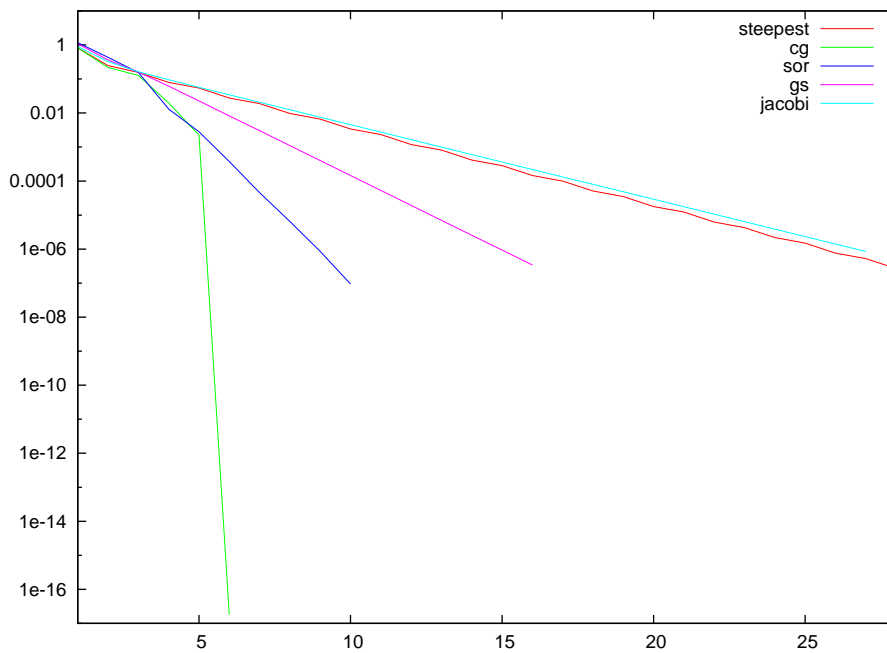
-1  
0  
1  
-2  
1  
2

Steepest descent took 28 iterations.  
CG took 6 iterations.  
SOR took 10 iterations.  
Gauss-Seidel took 16 iterations.  
Jacobi took 27 iterations.

Residual history for Problem 7:



Residual history for Problem 8:



Discussion:

Judging from the residual history plots, it is quite obvious that all methods except CG seem to produce algebraic convergence (i.e. there is an  $\alpha$  such that convergence is of  $\alpha$ th order), because their plots are roughly linear (note that the plots are semilogarithmic).

In particular, the convergence of the method of steepest descent is similar to that of the (pretty abysmal) Jacobi method, while the convergence of CG is faster than any algebraically-converging method.