```
%Homework 3 %involved m-files : -generalized.m (base for the iteration, used by other % methods) % -sornew.m (my own SOR algorithm, using generalized.m) % -jacobinew.m (own jacobi algo, using generalized.m) % -richardnew.m (own richardson algo, use generalized.m) % -cg.m (own CG algorithm) % -specrad.m (return the spectral radius of a matrix) % -lowerpart.m/upperpart.m (return the lower or upper % part of a matrix, with a minus sign in front of the % coeff, according to the lecture) % % % Warning : to obtain the same result as presented in the report, it is % necessary to compute ONLY the concerning part of the code (in the other close all;clear all; %===initialization==== A=[62,24,1,8,15 23,50,7,14,16 4,6,58,20,22 10,12,19,66,3 11,18,25,2,54]; b=[110;110;110;110;110]; n=length(b); w=0.6; %omega parameter k=500; %number of iterations %Building algorithms if 1%======SOR algorithm in action==== [thesort,t]=sornew(A,b,w,k); diff_sor=norm(thesort(end,:)'-A\b) %it works !! %======Jacobi algorithm in action==== thejack=jacobinew(A,b,k); diff_jac=norm(thejack(end,:)'-A\b) %it works too !! %=======CG algorithm in action==== %A1=rand(30); %b1=rand(30,1); thecard=cg(A,b,k); diff_cg=norm(thecard(end,:)'-A\b) %it also works ! Marvellous ! %======Richardson algorithm in action==== P=diag(diag(A)); alpha=0.8; therich=richardnew(A,b,P,k,alpha); diff_rich=norm(therich(end,:)'-A\b) %for P=diag(diag(A)), it works perfectly. For P=eye(n), I need a very small %alpha to have the convergence. end%======================== if 1%==========1) Convergence properties of SOR========== %====initialization i=0; j=0; step=0.01; nstep=2/step; rad=zeros(nstep,1); mdiff=rad; %====spectral radius + diff with matlab solution for a given #of steps k for w=0+step:step:2 i=i+1; x(i)=w; P=1/w*diag(diag(A))-lowerpart(A); rad(i)=specrad(P\(P-A)); rep=sornew(A,b,w,k); diff_sor(i)=norm(rep(end,:)'-A\b); end for w=0.7+step:step:1.5 j=j+1; y(j)=w; [rep,t,numb]=sornew(A,b,w,k); nit(j)=numb; end figure subplot(3,1,1),plot(x,rad,'mo-'); grid('on'); subplot(3,1,2),semilogy(x,diff_sor,'bo-'); grid('on'); subplot(3,1,3),plot(y,nit,'ko-'); end%choosen w with: 1.07 if 1%============2) Richardson Method============= P=eye(n); alpha=0.018; [repi,ti,numbi]=richardnew(A,b,P,k,alpha); diff_rich=norm(repi(end,:)'-A\b); j=0; step=0.0002; for alpha=0+step:step:0.018 j=j+1; y(j)=alpha; [repi,ti,numbi]=richardnew(A,b,P,k,alpha); nit(j)=numbi; end figure subplot(2,1,1),plot(y,nit,'b-'); %best alpha=0.0092 P=diag(diag(A)); i=0; step=0.02; for alpha=0+step:step:1.04 i=i+1; z(i)=alpha; [repd,td,numbd]=richardnew(A,b,P,k,alpha); nitd(i)=numbd; end subplot(2,1,2),plot(z,nitd,'k-'); %best alpha=0.82 end%it works, but not for every alpha, and not with eye(n) as P if 1%============3) Jacobi, Richardson, CG and SOR comparison========== %evolution of the error with the number of iterations: P=diag(diag(A)); sol=(A\b)'; m=500; best_alpha=0.82; %Ok best_w=1.07; % ok for k=1:m [repcg,tcg]=cg(A,b,k); timecg(k)=tcg; [repsor,tsor,numbsor]=sornew(A,b,best_w,k); timesor(k)=tsor; [reprich,trich,numbrich]=richardnew(A,b,P,k,best_alpha); timerich(k)=trich; [repjac,tjac,numbjac]=jacobinew(A,b,k); timejac(k)=tjac; errcg(k)=norm(repcg(k,:)-sol); errsor(k)=norm(repsor(k,:)-sol); errrich(k)=norm(reprich(k,:)-sol); errjac(k)=norm(repjac(k,:)-sol); end ab=(1:m)'; figure subplot(2,1,1), loglog(ab,errcg,'m+-',ab,errsor,'ko-.',ab,errrich,'r*:',ab,errjac,'b--'); xlabel('# of iterations'); ylabel('error'); legend('cg method','sor method','richardson method','jacobi method'); grid('on'); %time needed subplot(2,1,2), loglog(ab,timecg,'m-.',ab,timesor,'k-.',ab,timerich,'r-.',ab,timejac,'b--'); xlabel('# of iterations'); ylabel('time needed'); legend('cg method','sor method','richardson method','jacobi method'); grid('on'); end if 1%=====Random Matrices A========== n=10; m=500; best_alpha=0.002; %Ok best_w=0.25; % ??? A=rand(n); A=A*A'; b=rand(n,1); sol=(A\b)'; P=eye(n); for i=1:n A(i,i)=A(i,i)+max(A(i,:))*(n+1-i); end %A=A*10 for k=1:m [repcg,tcg]=cg(A,b,k); timecg(k)=tcg; [repsor,tsor,numbsor]=sornew(A,b,best_w,k); timesor(k)=tsor;
```

```
[reprich,trich,numbrich]=richardnew(A,b,P,k,best_alpha); timerich(k)=trich;
[repjac,tjac,numbjac]=jacobinew(A,b,k); timejac(k)=tjac; errcg(k)=norm(repcg(k,:)-sol);
errsor(k)=norm(repsor(k,:)-sol); errrich(k)=norm(reprich(k,:)-sol); errjac(k)=norm(repjac(k,:)-sol);
end ab=(1:m)'; figure loglog(ab,errcg,'m+-',ab,errsor,'ko-.',ab,errrich,'r*:',ab,errjac,'b--'); xlabel('#
of iterations'); ylabel('error'); legend('cg method','sor method','richardson method','jacobi
method'); grid('on'); %time needed errcg(m) errsor(m) errrich(m) errjac(m) pcg(A,b)'-sol
%b=A'*b; end %IT DOESN'T WORK ��� %F*** THIS, I'M OUTTA HERE
```