Alg. Meth. i.d. Numerik Übung 3

Florian Burndorfer, Jakob Fromherz,

Luna Sandner, Franz Scharnreitner

June 28, 2021

1 Lösung des Randwertproblems

1.1 Stiff1 und Stiff2

```
function [K_h] = Stiff1(N)

ndiag=-ones((N-1)^2-1,1);
ndiag(N-1:N-1:(N-1)^2-1)=zeros(N-2,1);

i=[1:(N-1)^2,1:(N-1)^2-1,1:(N-1)^2-N+1,2:(N-1)^2,N:(N-1)^2]; %[diag,obere diag,oberste diag,
    untere diag,unterste diag]
j=[1:(N-1)^2,2:(N-1)^2,N:(N-1)^2,1:(N-1)^2-1,1:(N-1)^2-N+1];
s=[4*ones((N-1)^2,1)',ndiag',-ones((N-1)^2-N+1,1)',ndiag',-ones((N-1)^2-N+1,1)'];

K_h=N^2*sparse(i,j,s,(N-1)^2,(N-1)^2);


end
```

```
function [K_h] = Stiff2(N)

diag = 4*ones((N-1)^2,1);
ndiag=-ones((N-1)^2,1);
ndiag(N-1:N-1:(N-1)^2-1)=zeros(N-2,1);
diag2=-ones((N-1)^2,1);

B=[diag2,ndiag,diag,ndiag,diag2];
d=[-(N-1),-1,0,1,N-1];

K_h = N^2*spdiags(B,d,(N-1)^2,(N-1)^2);


end
```

1.2 RHS

```
function [f_h] = RHS(f,N)

```

```
f_h=zeros((N-1)^2,1);
for j=1:N-1
    for i=1:N-1
        f_h(i+(j-1)*(N-1)) = f(i/N,j/N);
    end
end
end
```

1.3 Testen

```
for N=[10,100,1000]
    disp('N');
    disp(N);

    K_h1=Stiff1(N);
    K_h2=Stiff2(N);

    f_h=RHS(@f,N);

    sol=RHS(@u,N); %exakte Lsg

    disp('direkte Berechnung(Stiff1):')
    tic
    u_h_direct1 = K_h1\f_h;
    toc

    disp('Fehler direkte Lsg mit Stiff1 (Maximumsnorm):')
    max(abs(u_h_direct1-sol))

    disp('direkte Berechnung(Stiff2)')
    tic
    u_h_direct2 = K_h2\f_h;
    toc

    disp('Fehler direkte Lsg mit Stiff2 (Maximumsnorm):')
    max(abs(u_h_direct2-sol));

    disp('iterative Berechnung(Stiff1)')
```

```
    tic
    u_h_pcg1 = pcg(K_h1,f_h);
    toc

    disp('Fehler iterative Lsg mit Stiff1 (Maximumsnorm):')
    max(abs(u_h_pcg1-sol))

    disp('iterative Berechnung(Stiff2)')
    tic
    u_h_pcg2 = pcg(K_h2,f_h);
    toc

    disp('Fehler iterative Lsg mit Stiff2 (Maximumsnorm):')
    max(abs(u_h_pcg2-sol))

    fprintf('--------------------------------------------\n');
end
```

Bei $N = 10$ ist der grüßte Fehler zu beobachten. Sonst ist der Fehler in der Größenordnung 10^{-4}, wobei die iterative Berechnung ungenauer ist als die direkte. Die iterative Berechnung braucht nicht mehr als 20 Iterationen. Sie ist für große N auch schnelle als die direkte Berechnung.

2 Jacobi

```
function [x,exitFlag] = Jacobi(A,b,x0,itMax,myEps)
%JACOBI Solves linear Equation via jacobi method
    if(nargin <= 4)
       myEps = 1.E-8;
    end
    if(nargin <= 3)
        itMax = size(A,1)^2;
    end
    if(nargin == 2)
        x0 = zeros(size(A,1),1);
    end

    d = diag(A);
```

```
    R= A - diag(diag(A));
    xk = x0;
    r0 = norm(A * x0 - b,2);
    for k = 1:itMax
      xk = (b-R*xk)./d;

      if (norm(A * xk - b,2) <= myEps * r0)
        exitFlag = 1;
        x=xk;
        return;
      end
    end
    x = x0;
    exitFlag = 0;
end
```

2.1 Testen

```
for N = [10,100,1000]
disp('N:')
disp(N)
K_h = Stiff1(N);
f_h = RHS(@f,N);

tic;
[u_h,err] =  Jacobi(K_h,f_h);
toc;

if(err == 0 )
    disp('Did not converge');
end
linSol = K_h\f_h;
itSol = pcg(K_h,f_h);

u_h;

max(abs(u_h-linSol))
max(abs(u_h-itSol))
```

```
end
```

Das Skript testet die Funktion für N = 10, 100, 1000. Für N = 10 braucht die Funktion 0.00298s, für N = 100 rund 9s und für N = 1000 braucht die Funktion zu lange, um das Ergebnis abzuwarten. Die Abweichung zur direkten Berechnung beträgt für N = 10,100 rund 10^{-6}. Die Abweichung zur iterative Berechnungs ist größer.