Departamento de Matemática da Universidade de Coimbra

Análise Numérica I

Ano lectivo 1999/2000 Folha suplementar 2

—◇—

Sistemas de equações lineares

1. Aplique os métodos de Jacobi e de Gauss-Seidel na resolução dos sistemas de equações lineares,

$$\begin{cases} -x + 10y = 19 \\ 5x - y = 3 \end{cases} \quad \text{e} \quad \begin{cases} 5x - y = 3 \\ -x + 10y = 19 \end{cases}.$$

   Compare as aproximações obtidas com as soluções exactas e analise os erros cometidos.

2. Os sistemas de equações lineares

$$\begin{cases} 1.01x + y = 1 \\ x + y = 2 \end{cases} \quad \text{e} \quad \begin{cases} 1.02x + y = 1 \\ x + y = 2 \end{cases}$$

   são muito "semelhantes". Resolva-os e compare as suas soluções.

Sugestão de resolução usando o método de Gauss-Seidel:

1. Considerar a matriz $A = \begin{bmatrix} 5 & -1 \\ -1 & 10 \end{bmatrix}$ e o vector $b = \begin{bmatrix} 3 \\ 19 \end{bmatrix}$.

```
A={{5,-1},{-1,10}}; b={3,19};
```

2. Verificar se o sistema dado é possível e determinado.

```
Det[A]
```

3. Definir função para verificar a convergência do método de Gauss-Seidel para um sistema, $ax = b$.

```
convergenciaDeGaussSeidel[a_]:=(
  n=Length[a];
(* Calcular matriz de iteracao do metodo *)
  dl=Table[If[i>=j,a[[i,j]],0],{i,n},{j,n}];
  u=Table[If[i<j,-a[[i,j]],0],{i,n},{j,n}];
  inversa=Inverse[dl];
  mgs=inversa.u;
(* Calcular valores proprios dessa matriz *)
  listaDeValoresProprios=Eigenvalues[mgs];
  Print["Valores proprios da matriz de iteracao:", listaDeValoresProprios];
  If[Max[Abs[N[listaDeValoresProprios]]]<1,
    Print["O metodo de Gauss-Seidel converge"],
    Print["O metodo de Gauss-Seidel nao converge"]]
);
```

4. Aplicar a função definida ao sistema dado.

```
convergenciaDeGaussSeidel[A]
```

5. Definir funções para determinar algumas normas vectoriais.

```
normal1[b_]:=Max[Sum[Abs[b[[i]]],{i,Length[b]}]]
normaInf[b_]:=Max[Abs[b]]
norma2[b_]:=N[Sqrt[Sum[b[[i]]^2,{i,Length[b]}]]]
```

6. Definir função para aplicar o método de Gauss-Seidel a um sistema linear, $ax = b$.

```
metodoDeGaussSeidel[a_,b_]:=(
(* Inicializar variaveis *)
  n=Length[b];
  tol1=10^-5; tol2=10^-5; listaDeErro1={}; listaDeErro2={};
  itMax=50; k=0; x=Table[0,{i,n}]; continua=True;
(* Aplicar o metodo *)
  While[continua,
    antx=x;
    Do[ soma1=Sum[-a[[i,j]]/a[[i,i]] x[[j]], {j,i+1,n}];
      soma2=Sum[-a[[i,j]]/a[[i,i]] x[[j]], {j,1,i-1}];
      x[[i]]=soma1+soma2+b[[i]]/a[[i,i]],
      {i,1,n}
    ];
    erro1=norma2[x-antx]; listaDeErro1=Join[listaDeErro1,{erro1}];
    erro2=norma2[a.x-b]; listaDeErro2=Join[listaDeErro2,{erro2}];
    k=k+1;
(* Escrever aproximacao de ordem k *)
    Print["x",k,"= ",x];
(* Condicao de paragem *)
    continua=(k<=itMax && erro1>tol1 && erro2>tol2);
  ];
(* Escrever resultado da aplicacao do metodo *)
  If[k>itMax,
    Print["A solucao nao foi encontrada em ",itMax," iteracoes"],
    Print["x",k,"= ",x]
  ];
)
```

7. Aplicar a função definida ao sistema dado.

```
metodoDeGaussSeidel[A, b]
```

8. Determinar a solução exacta do sistema.

```
LinearSolve[A,b]
Solve[{5x-y==3,-x+10y==19},{x,y}]
<<LinearAlgebra`GaussianElimination`
LUSolve[LUFactor[A],b]
```

9. Representar graficamente os erros cometidos nas sucessivas iterações efectuadas.

```
graf1=ListPlot[listaDeErro1, PlotJoined->True]
graf2=ListPlot[listaDeErro2, PlotJoined->True]
Show[graf1,graf2]
```

Sugestão de implementação do método de Jacobi:

1. Definir função para verificar a convergência do método de Jacobi para um sistema, $ax = b$.

```
convergenciaDeJacobi[a_]:=(
  n=Length[a];
(* Calcular matriz de iteracao do metodo *)
  mj=Table[If[i!=j,-a[[i,j]]/a[[i,i]],0],{i,n},{j,n}];
(* Calcular valores proprios dessa matriz *)
  listaDeValoresProprios=Eigenvalues[mj];
  Print["Valores proprios da matriz de iteracao:", listaDeValoresProprios];
  If[Max[Abs[N[listaDeValoresProprios]]]<1,
    Print["O metodo de Jacobi converge"],
    Print["O metodo de Jacobi nao converge"]]
);
```

2. Definir função para aplicar o método de Jacobi a um sistema linear, $ax = b$.

```
metodoDeJacobi[a_,b_]:=(
(* Inicializar variaveis *)
  n=Length[b];
  tol1=10^-5; tol2=10^-5; listaDeErro1={}; listaDeErro2={};
  itMax=50; k=0; x=Table[0,{i,n}]; continua=True;
(* Aplicar o metodo *)
  While[continua,
    antx=x;
    Do[ soma1=Sum[-a[[i,j]]/a[[i,i]] x[[j]], {j,i+1,n}];
      soma2=Sum[-a[[i,j]]/a[[i,i]] x[[j]], {j,1,i-1}];
      aux[[i]]=soma1+soma2+b[[i]]/a[[i,i]],
      {i,1,n}
    ];
    x=N[Table[aux[i], {i,1,n}]];
    erro1=norma2[x-antx]; listaDeErro1=Join[listaDeErro1,{erro1}];
    erro2=norma2[a.x-b]; listaDeErro2=Join[listaDeErro2,{erro2}];
    k=k+1;
(* Escrever aproximacao de ordem k *)
    Print["x",k,"= ",x];
(* Condicao de paragem *)
    continua=(k<=itMax && erro1>tol1 && erro2>tol2);
  ];
(* Escrever resultado da aplicacao do metodo *)
  If[k>itMax,
    Print["A solucao nao foi encontrada em ",itMax," iteracoes"],
    Print["x",k,"= ",x]
  ];
)
```