UTFPR – Universidade Tecnológica Federal do Paraná – Campus Toledo

Cálculo Numérico – 2024/2

Aula computacional 2

SISTEMAS LINEARES – SISTEMAS NÃO LINEARES

1. Resolva o sistema linear:
a) Utilizando o comando solve do R;
b) Utilizando eliminação de Gauss;
c) Utilizando fatoração LU.

$$\begin{cases} x + 2y + z = 4 \\ 3x + 8y + 7z = 20 \\ 2x + 7y + 9z = 23 \end{cases}$$

Solução

```
# a)
# Definindo a matriz A
A<-matrix(c(1,2,1,3,8,7,2,7,9),nrow=3,ncol=3,byrow=T)
A
     [,1] [,2] [,3]
[1,]    1    2    1
[2,]    3    8    7
[3,]    2    7    9

# Definindo o vetor b
b<-c(4,20,23)
b
[1]  4 20 23

# Resolvendo o sistema
solve(A,b)
[1]  5 -2  3
```

b)
Definindo a matriz A

```
A<-matrix(c(1,2,1,3,8,7,2,7,9),nrow=3,ncol=3,byrow=T)
A
     [,1] [,2] [,3]
[1,]    1    2    1
[2,]    3    8    7
[3,]    2    7    9
```

Definindo o vetor b

```
b<-c(4,20,23)
b
[1]  4 20 23
```

Carregando o pacote matlib

```
require(matlib)
```

Método da Eliminação de Gauss

```
gaussianElimination(A,b,verbose=TRUE)

     [,1] [,2] [,3] [,4]
[1,]    1    0    0    5
[2,]    0    1    0   -2
[3,]    0    0    1    3
```

Dica: Utilizando o parâmetro "verbose=TRUE", é possível observar as operações elementares.

```
gaussianElimination(A,b,verbose=TRUE)
```

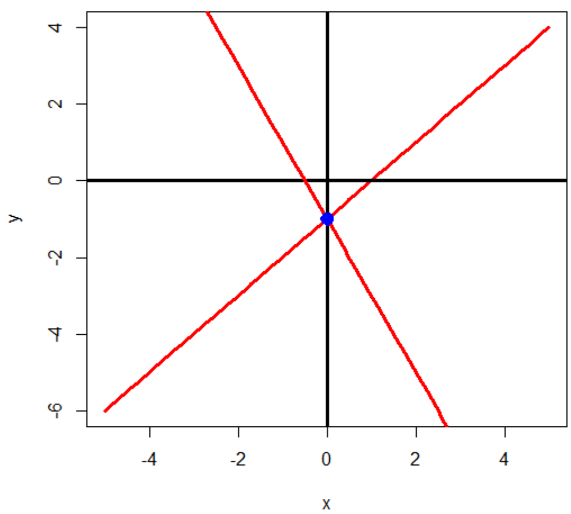
```
# c)
# Definindo a matriz A
A<-matrix(c(1,2,1,3,8,7,2,7,9),nrow=3,ncol=3,byrow=T)
A
[,1] [,2] [,3]
[1,]    1    2    1
[2,]    3    8    7
[3,]    2    7    9

# Definindo o vetor b
b<-c(4,20,23)
b
[1]  4 20 23

require(pracma)

LU<-lu(A)
LU

$L
     [,1] [,2] [,3]
[1,]    1  0.0    0
[2,]    3  1.0    0
[3,]    2  1.5    1

$U
     [,1] [,2] [,3]
[1,]    1    2    1
[2,]    0    2    4
[3,]    0    0    1

# Resolvendo Ly=b
y<-solve(LU$L,b)
y
[1] 4 8 3

# Resolvendo Ux=y
x<-solve(LU$U,y)
x
[1]  5 -2  3
```

2.Resolva graficamente o sistema .

$$\begin{cases} x - y = 1 \\ 2x + y = -1 \end{cases}$$

Solução

```
# Definindo as funções
f1<-function(x){x-1}
f2<-function(x){-2*x-1}

# Esboçando o gráfico
plot(f1,-5,5,lwd=3,col="red",ylab="y")
abline(h=0,v=0,lwd=3)
plot(f2,-5,5,lwd=3,col="red",add=T)


# Resolvendo utilizando o comando solve
A<-matrix(c(1,-1,2,1),nrow=2,ncol=2,byrow=T)
b<-c(1,-1)
solve(A,b)
[1]  0 -1

# Marcando a solução no gráfico
points(0,-1,pch=10,col="blue",lwd=6)
```



3. Considere o sistema linear $\begin{cases} 10x_1 + 5x_2 + x_3 = 14 \\ x_1 + 5x_2 + x_3 = 11 \\ 2x_1 + 3x_2 + 10x_3 = 8 \end{cases}$.

a) Considerando $X^{(0)} = [0,0,0]^T$ como aproximação inicial, calcule 10 aproximações utilizando Gauss – Jacobi.

Solução

```
# Carregando o pacote pracma
require(pracma)

# Definindo a matriz A
A<-matrix(c(10,5,1,1,5,1,2,3,10),ncol=3,nrow=3,byrow=T)

# Definindo o vetor b
b<-c(14,11,8)

# Definindo a aproximação inicial
start<-c(0,0,0)

# Construindo uma matriz para receber os resultados
resultados<-matrix(,ncol=3,nrow=10)

#esquema iterativo
for ( i in 1:10){
a<-itersolve(A,b, start, method = "Jacobi", nmax=i)
resultados[i,1]<-a$x[1]
resultados[i,2]<-a$x[2]
resultados[i,3]<-a$x[3]
}

resultados

          [,1]     [,2]        [,3]
 [1,] 1.4000000 2.200000  0.80000000
 [2,] 0.2200000 1.760000 -0.14000000
 [3,] 0.5340000 2.184000  0.22800000
 [4,] 0.2852000 2.047600  0.03800000
 [5,] 0.3724000 2.135360  0.12868000
 [6,] 0.3194520 2.099784  0.08491200
 [7,] 0.3416168 2.119127  0.10617440
 [8,] 0.3298190 2.110442  0.09593848
 [9,] 0.3351853 2.114849  0.10090368
[10,] 0.3324854 2.112782  0.09850839
```

b) Considerando $X^{(0)} = [0,0,0]^T$ como aproximação inicial, calcule 10 aproximações utilizando Gauss – Seidel.

Solução

```
# Carregando o pacote pracma
require(pracma)

# Definindo a matriz A
A<-matrix(c(10,5,1,1,5,1,2,3,10),ncol=3,nrow=3,byrow=T)

# Definindo o vetor b
b<-c(14,11,8)

# Definindo a aproximação inicial
start<-c(0,0,0)

# Construindo uma matriz para receber os resultados
resultados<-matrix(,ncol=3,nrow=10)

#esquema iterativo
for ( i in 1:10){
a<-itersolve(A,b,start,method = "Gauss-Seidel",nmax=i)
resultados[i,1]<-a$x[1]
resultados[i,2]<-a$x[2]
resultados[i,3]<-a$x[3]
}

resultados

        [,1]     [,2]        [,3]
 [1,] 1.4000000 1.920000 -0.05600000
 [2,] 0.4456000 2.122080  0.07425600
 [3,] 0.3315344 2.118842  0.09804054
 [4,] 0.3307750 2.114237  0.09957393
 [5,] 0.3329242 2.113500  0.09936505
 [6,] 0.3333133 2.113464  0.09929804
 [7,] 0.3333380 2.113473  0.09929056
 [8,] 0.3333346 2.113475  0.09929060
 [9,] 0.3333335 2.113475  0.09929075
[10,] 0.3333333 2.113475  0.09929078
```

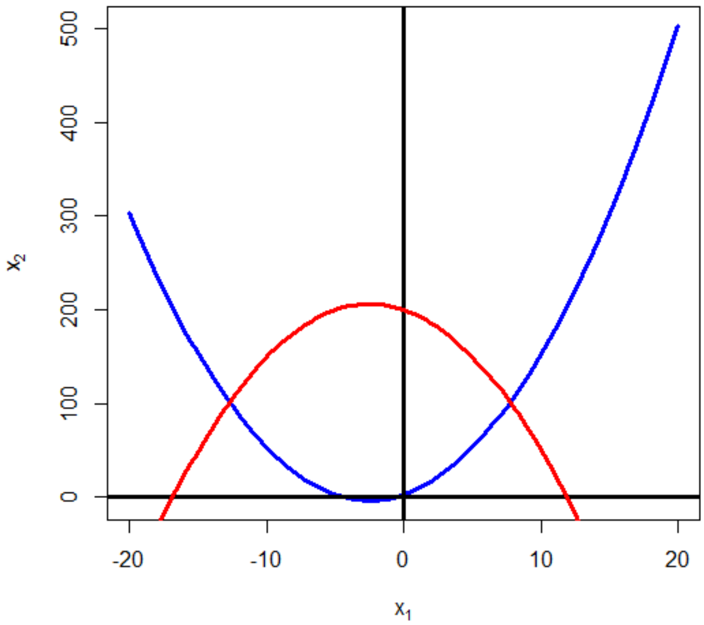
4. Considere o sistema não linear abaixo:

$$\begin{cases} x_1^2 + 5x_1 + 3 - x_2 = 0 \\ -x_1^2 - 5x_1 + 200 - x_2 = 0 \end{cases}$$

Faça um gráfico para observar as raízes.
Calcule aproximações pelo método de Newton.

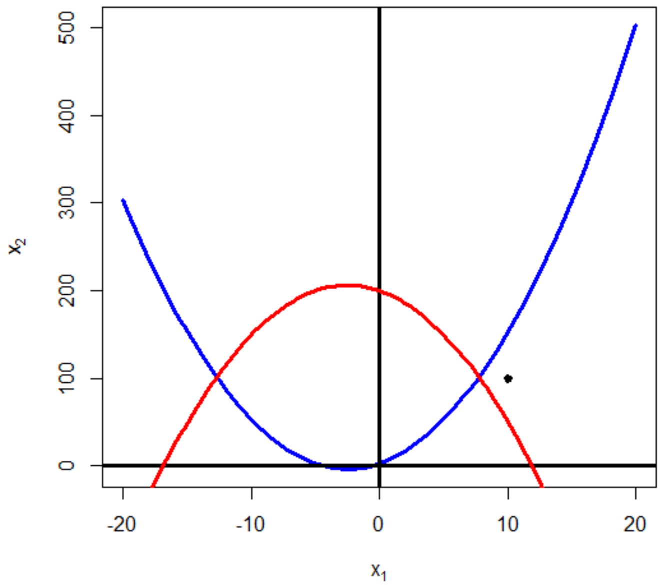
Solução

```
# Visualizando graficamente
f<-function(x){x^2+5*x+3}
plot(f,-20,20,xlab=expression('x'[1]),
ylab=expression('x'[2]),
lwd=3,col="blue")
abline(h=0,v=0,lwd=3)
g<-function(x){-x^2-5*x+200}
plot(g,-20,20,add=T,col="red",lwd=3)
```




```
# Carregando o pacote
require(nleqslv)


# Definindo o sistema
sistema <- function(x) {
y <- numeric(2)
y[1] <- x[1]^2+5*x[1]+3-x[2]
y[2] <- -x[1]^2-5*x[1]+200-x[2]
y
}
```

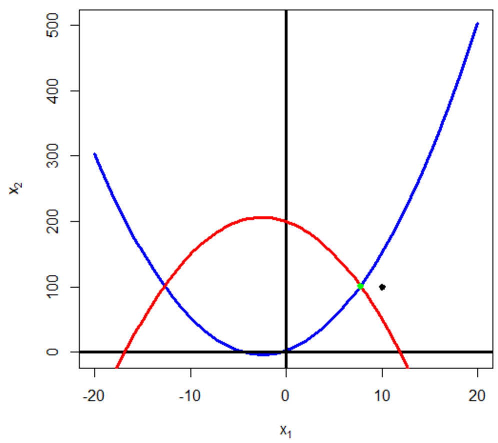
Primeira raiz
#Definindo a aproximação inicial
```
xstart <- c(10,100)
points(10,100,pch=20,lwd=3,col="black")
```



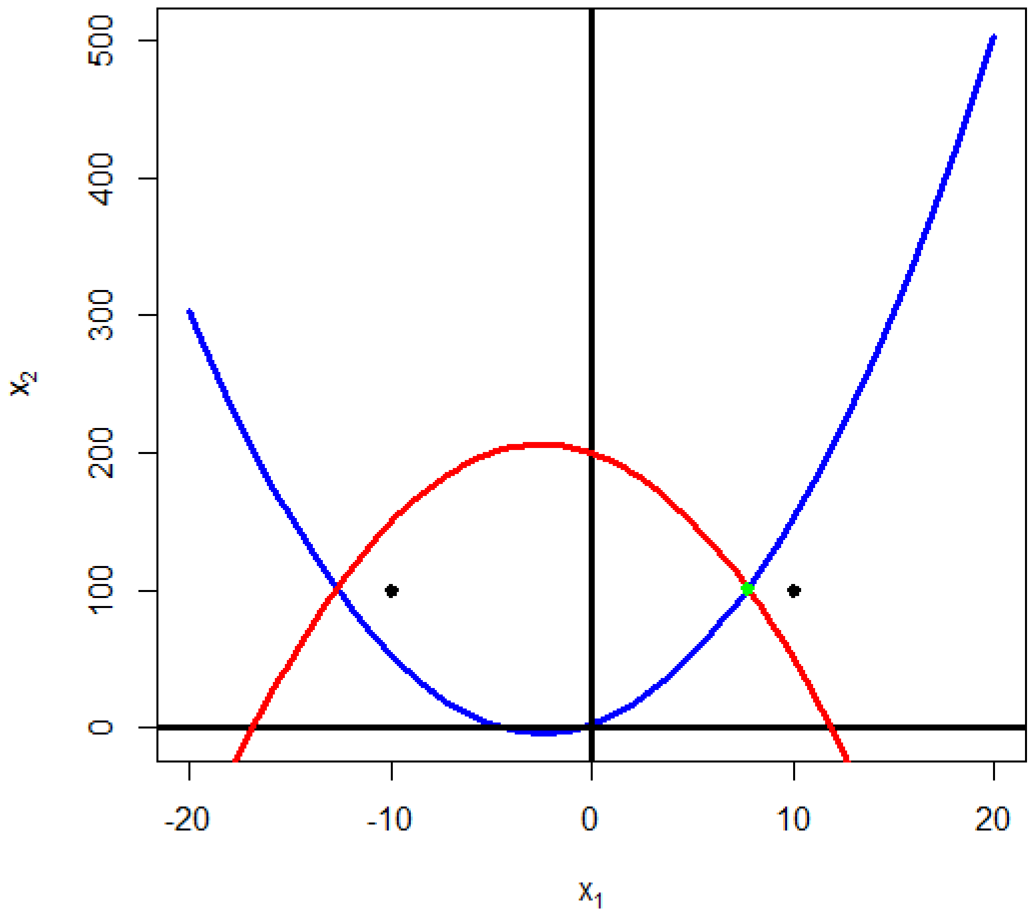
Resolvendo
```
nleqslv(xstart,sistema,method = c("Newton"),control=list(xtol=0.000001))
```

```
$x
[1]   7.734745 101.500000

$fvec
[1]  5.684342e-14 -5.684342e-14

$iter
[1] 4
```

#Plotando a solução encontrada
```
points(7.734745,101.500000,pch=20,lwd=3,col="green")
```



Segunda raiz
#Definindo a aproximação inicial
```
xstart<- c(-10,100)
points(-10,100,pch=20,lwd=3,col="black")
```



Resolvendo
```
nleqslv(xstart,sistema,method = c("Newton"),control=list(xtol=0.000001))
```

```
$x
[1] -12.73474 101.50000

$fvec
[1]  4.187939e-11 -4.187939e-11


$iter
[1] 4
```

#Plotando a solução encontrada
```
points(-12.73474,101.50000,pch=20,lwd=3,col="green")
```