2 Sistemas de equações lineares

2.1 Representação matricial

Um sistema linear de ordem n é um sistema com n variáveis $x_1, x_2, \ldots, x_n$, que satisfazem n equações lineares:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{21}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned} \tag{2.1}$$

Se as equações são linearmente independentes, existe sempre uma única solução. A **matriz aumentada** do sistema é:

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{array}\right] \tag{2.2}$$

2.2 Método de eliminação de Gauss

A qualquer equação no sistema pode somar-se outra das equações, multiplicada por uma constante k, e a solução do sistema não se altera. Na matriz aumentada, essa operação corresponde a substituir uma linha L_i por $L_i + kL_j$. Essa propriedade pode ser usada para obter uma matriz **triangular superior**, em que a_{ij} é zero, se $i < j$:

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ 0 & a'_{22} & \cdots & a'_{2n} & b'_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a'_{nn} & b'_n \end{array}\right] \tag{2.3}$$

que corresponde a outro sistema de equações com a mesma solução do sistema original (as constantes a'_{ij} e b'_j não são as mesmas a_{ij} e b_j da matriz inicial).

A última equação nesse sistema permite obter facilmente o valor da variável x_n

$$a'_{nn}x_n = b'_n \quad \Rightarrow \quad x_n = \frac{b'_n}{a'_{nn}} \tag{2.4}$$

Esse resultado pode ser substituído na penúltima equação para obter o valor de x_{n-1},

$$a'_{n-1,n-1}x_{n-1} + a'_{n-1,n}x_n = b'_{n-1} \quad \Rightarrow \quad x_n = \frac{b'_n - 1}{a'_{n-1,n-1}} - \frac{a'_{n-1,n}b'_n}{a'_{n-1,n-1}a'_{nn}} \tag{2.5}$$

e assim sucessivamente, até se obter o valor de todas as variáveis.

Exemplo 2.1
Resolva o sistema de equações:

$$\begin{aligned} 8x + 4.5z - 12.4 &= 3.2y \\ 2x - 0.8y &= 7.6 + 6.1z \\ x + 3y + 10.2z &= 8.4 \end{aligned}$$

usando o método de eliminação de Gauss.

Resolução. Usando o Maxima, convém guardar-se as 3 equações em 3 variáveis

```
(%i1) eq1: 8*x + 4.5*z - 12.4 = 3.2*y$
(%i2) eq2: 2*x - 0.8*y = 7.6 + 6.1*z$
(%i3) eq3: x + 3*y + 10.2*z = 8.4$
```

Este sistema pode ser escrito na mesma forma do sistema geral (2.1) para obter a matriz aumentada. A função `augcoefmatrix` realiza esse procedimento dando como resultado a matriz aumentada; é necessário dar dois argumentos a `augcoefmatrix`, a lista das equações e a lista das variáveis, na ordem que se quiser. Neste caso, a matriz aumentada pode ser guardada na variável *m* com o seguinte comando:

```
(%i4) m: float (augcoefmatrix ([eq1, eq2, eq3], [x, y, z]));
                    [ 8.0   - 3.2    4.5    - 12.4 ]
                    [                              ]
(%o4)               [ 2.0   - 0.8  - 6.1    - 7.6  ]
                    [                              ]
                    [ 1.0    3.0    10.2    - 8.4  ]
```

optou-se por usar a função `float` para evitar que a matriz seja escrita com números racionais.

Há que ter em conta que os números na última coluna não são b_i mas sim $-b_i$ (ou seja, no sistema (2.1) todas as equações foram igualadas a zero).

Antes de começar com o processo de eliminação podem fixar-se um número reduzido de algarismos significativos, por exemplo 4, para os resultados que serão apresentados a seguir, de modo a evitar que as matrizes tenham muitos algarismos:

```
(%i5) fpprintprec: 4$
```

Como está a ser usado o formato de vírgula flutuante de dupla precisão (8 bytes), internamente os cálculos continuarão a ser feitos com 16 algarismos significativos, mas cada vez que se apresentem os resultados, serão arredondados para 4 algarismos significativos.

Para reduzir a matriz à forma triangular, começa-se por substituir a segunda linha, L_2, por $L_2 - (1/4)L_1$, onde L_1 representa toda a primeira linha da matriz. O objetivo é fazer com que a_{21} fique igual a zero (note-se que 1/4 foi calculado dividindo a_{21} por a_{11}).

No Maxima, a função `rowop`(m, i, j, k) produz uma nova matriz, em que a linha L_i da matriz m é substituída por $L_i - kL_j$ e as restantes linhas ficam iguais. Assim sendo, a substituição $L_2 - (1/4)L_1 \longrightarrow L_2$ pode ser feita com o seguinte comando:

```
(%i6) m: rowop (m, 2, 1, 1/4);
                    [ 8.0  - 3.2    4.5    - 12.4 ]
                    [                             ]
(%o6)               [ 0.0   0.0   - 7.225  - 4.5  ]
                    [                             ]
                    [ 1.0   3.0    10.2    - 8.4  ]
```

A seguir, para eliminar $a_{31} = 1.0$ usando novamente $a_{11} = 8.0$, deverá ser feita a substituição $L_3 - kL_1 \longrightarrow L_3$, onde $k = a_{11}/a_{31} = 1/8$. Isto é,

```
(%i7) m: rowop (m, 3, 1, 1/8);
                    [ 8.0  - 3.2    4.5    - 12.4 ]
                    [                             ]
(%o7)               [ 0.0   0.0   - 7.225  - 4.5  ]
                    [                             ]
                    [ 0.0   3.4    9.637   - 6.85 ]
```

Neste momento deveria usar-se a'_{22} para eliminar a'_{23}. No entanto, como $a'_{22} = 0.0$, isso não é possível ($k = a'_{23}/a'_{22}$ não existe). Em casos como este pode-se trocar a ordem das segunda e terceira linhas, que equivale a alterar a ordem das equações, sem que por isso se altere a solução. A função `rowswap` do Maxima permite trocar a ordem de duas linhas:

```
(%i8) m: rowswap (m, 2, 3);
                    [ 8.0  - 3.2    4.5    - 12.4 ]
                    [                             ]
(%o8)               [ 0.0   3.4    9.637   - 6.85 ]
                    [                             ]
                    [ 0.0   0.0   - 7.225  - 4.5  ]
```

ficando assim uma matriz diagonal superior, que permite passar à segunda fase do método.

Para resolver as equações associadas com cada linha da matriz é conveniente definir uma lista com as variáveis, na mesma ordem em que foram usadas para encontrar a matriz aumentada, seguidas por 1:

```
(%i9) vars: [x, y, z, 1]$
```

Multiplicando a terceira linha da matriz m pela lista de variáveis, obtém-se a última equação, na qual só aparece a variável z; resolve-se essa equação para encontrar o valor de z, que será guardado para ser usado nos passos seguintes

```
(%i10) sol: float (solve (m[3].vars));
(%o10)                        [z = - .6228]
```

Multiplicando a segunda linha da matriz pela lista de variáveis e substituindo o valor que já foi encontrado para z, obtém-se uma equação que depende unicamente de y. A resolução dessa equação conduz ao valor de y e é conveniente acrescentar esse resultado na mesma lista em que foi armazenado o valor de z

```
(%i11) sol: append (float (solve ( subst (sol, m[2].vars))), sol);
(%o11)                  [y = 3.78, z = - .6228]
```

Finalmente, basta repetir o passo anterior, mas agora multiplicando pela primeira linha da matriz, para determinar o valor de x

```
(%i12) sol: append (float (solve ( subst (sol, m[1].vars))), sol);
(%o12)           [x = 3.412, y = 3.78, z = - .6228]
```

2.3 Método de Gauss-Jordan

O método de eliminação de Gauss tem a vantagem de não alterar o valor do determinante da matriz com coeficientes a_{ij} e, como tal, o determinante pode ser calculado facilmente multiplicando os valores a'_{ii} na diagonal da matriz triangular obtida.

O método de Gauss-Jordan consiste em transformar a matriz aumentada até os coeficientes a'_{ij} corresponder à matriz identidade, ou seja, iguais a 1 se $i = j$ ou zero caso contrário. A grande vantagem é que os valores finais de b'_i são os valores das variáveis, sem ser preciso realizar mais contas. Em contrapartida, a matriz final não permite calcular o determinante de a_{ij}.

Para conseguir transformar a matriz na matriz identidade, começa-se por dividir a primeira equação por a_{11}, para que a'_{11} fique igual a 1, e a seguir usa-se essa primeira linha para eliminar o primeiro elemento em todas as outras linhas. A seguir divide-se a segunda linha por a'_{22}, para que este fique igual a 1 e usa-se essa segunda linha para eliminar o segundo elemento em todas as outras linhas, incluindo a primeira.

Não existe um comando específico do Maxima para dividir uma linha de uma matriz por uma constante; no entanto, a divisão por k pode ser feita com a função `rowop`(m, i, i, $1-1/k$), que substitui a linha L_i da matriz m, por $L_i-(1-1/k)\,L_i = L_i/k$.

Exemplo 2.2
Determine a solução do seguinte sistema, pelo método de Gauss-Jordan

$$\begin{aligned} 2x_1 - 1.2x_2 - 4x_3 &= 5 \\ -3x_1 + 3x_2 + 5x_3 &= 12 \\ 2.5x_1 + 4x_2 - 3x_3 &= -6 \end{aligned}$$

Resolução. Para mostrar como é possível obter soluções exatas, na resolução deste exemplo só será usada a função `float` no fim e os dois números de vírgula flutuante 1.2 e 2.5 na primeira e terceira equações serão escritos como as fracções 12/10 e 25/10.

Em vez de se usar a função `augcoefmatrix`, a matriz aumentada pode também ser inserida diretamente usando a função `matrix`

```
(%i13) m: matrix([2,-12/10,-4,5],[-3,3,5,12],[25/10,4,-3,-6]);
                        [         6           ]
                        [  2    - -  - 4    5  ]
                        [         5           ]
                        [                     ]
(%o13)                  [ - 3    3     5    12 ]
                        [                     ]
                        [  5                  ]
                        [  -     4    - 3  - 6 ]
                        [  2                  ]
```

Começa-se por dividir a primeira linha pelo seu primeiro elemento (ou seja 2):

```
(%i14) m: rowop (m, 1, 1, 1 - 1/m[1][1]);
                        [         3         5  ]
                        [  1    - -  - 2    -  ]
                        [         5         2  ]
                        [                     ]
(%o14)                  [ - 3    3     5    12 ]
                        [                     ]
                        [  5                  ]
                        [  -     4    - 3  - 6 ]
                        [  2                  ]
```

A seguir, subtrai-se a primeira linha, multiplicada pelo primeiro elemento da segunda linha, para eliminar o primeiro elemento de segunda linha:

```
(%i15) m: rowop (m, 2, 1, m[2][1]);
                        [       3         5  ]
                        [ 1   - -  - 2    -  ]
                        [       5         2  ]
                        [                    ]
                        [      6         39  ]
(%o15)                  [ 0    -    - 1   --  ]
                        [      5         2   ]
                        [                    ]
                        [ 5                  ]
                        [ -    4    - 3  - 6 ]
                        [ 2                  ]
```

e faz-se o mesmo com a primeira e a terceira linhas, para eliminar o primeiro elemento da terceira linha:

```
(%i16) m: rowop (m, 3, 1, m[3][1]);
                        [       3         5   ]
                        [ 1   - -  - 2    -   ]
                        [       5         2   ]
```

```
                          [                    ]
                          [     6          39  ]
(%o16)                    [ 0   -   - 1    --  ]
                          [     5          2   ]
                          [                    ]
                          [    11           49 ]
                          [ 0  --    2    - -- ]
                          [    2            4  ]
```

O mesmo procedimento repete-se para a segunda coluna; isto é, divide-se a segunda linha pelo seu segundo elemento,

```
(%i17) m: rowop (m, 2, 2, 1 - 1/m[2][2]);
                          [      3         5   ]
                          [ 1  - -   - 2   -   ]
                          [      5         2   ]
                          [                    ]
                          [            5   65  ]
(%o17)                    [ 0   1    - -   --  ]
                          [            6   4   ]
                          [                    ]
                          [    11           49 ]
                          [ 0  --    2    - -- ]
                          [    2            4  ]
```

e usa-se essa nova linha para eliminar o segundo elemento na primeira e na terceira linhas:

```
(%i18) m: rowop (rowop (m, 1, 2, m[1][2]), 3, 2, m[3][2]);
                          [          5   49    ]
                          [ 1  0   - -   --    ]
                          [          2   4     ]
                          [                    ]
                          [          5   65    ]
(%o18)                    [ 0  1   - -   --    ]
                          [          6   4     ]
                          [                    ]
                          [        79      813 ]
                          [ 0  0   --    - --- ]
                          [        12      8   ]
```

Finalmente, repete-se o mesmo procedimento para a terceira coluna:

```
(%i19) m: rowop (m, 3, 3, 1 - 1/m[3][3]);
                          [          5    49   ]
                          [ 1  0   - -    --   ]
                          [          2    4    ]
                          [                    ]
                          [          5    65   ]
(%o19)                    [ 0  1   - -    --   ]
                          [          6    4    ]
```

```
                          [                   ]
                          [              2439 ]
                          [ 0  0   1   - ---- ]
                          [              158  ]
(%i20) m: rowop (rowop (m, 1, 3, m[1][3]), 2, 3, m[2][3]);
                           [             2081 ]
                           [ 1  0   0  - ---- ]
                           [              79  ]
                           [                  ]
                           [            535   ]
(%o20)                     [ 0  1   0   ---   ]
                           [            158   ]
                           [                  ]
                           [             2439 ]
                           [ 0  0   1  - ---- ]
                           [             158  ]
```

Os três números na última coluna são os valores exatos das 3 variáveis. Para aproximar esses números por números de vírgula flutuante e colocá-los numa lista, extrai-se a quarta linha da matriz transposta e aplica-se a função `float`

```
(%i21) float (transpose(m)[4]);
(%o21)                [- 26.34, 3.386, - 15.44]
```

Note-se que se tivesse sido usada a função `augcoefmatrix` para construir a matriz, seria necessário mudar os sinais desses 3 valores obtidos.

Este algoritmo pode ser automatizado facilmente; o mesmo resultado obtido com os comandos (`%i13`) até (`%i20`) podia ter-se obtido com os seguintes comandos:

```
(%i22) m: matrix([2,-12/10,-4,5],[-3,3,5,12],[25/10,4,-3,-6])$
(%i23) for i:1 thru 3 do (
         m: rowop (m, i, i, 1 - 1/m[i][i]),
         for j:1 thru 3 do (
           if (i # j) then m: rowop (m, j, i, m[j][i])));
(%o23)                          done
```

onde # é o operador "diferente de".

Se em alguma das iterações `m[i][i]` for nulo, `1/m[i][i]` produz um erro. Para evitar esse tipo de erro seria necessário acrescentar instruções para conferir que `m[i][i]` não seja nulo e, caso contrário, trocar a linha L_i com alguma das linhas L_j com $j > i$.