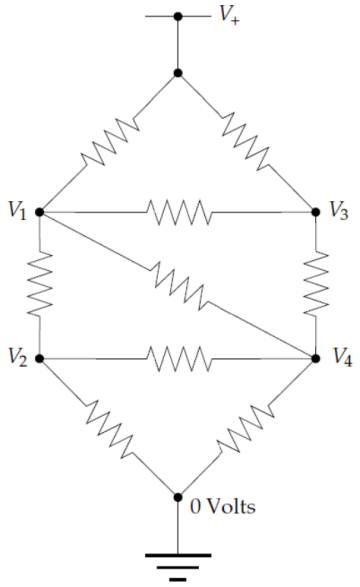
Solución Tarea 7
FISI 6510

Guillermo Fidalgo

2 de abril de 2021

Exercise 6.1: A circuit of resistors

Consider the following circuit of resistors:



All the resistors have the same resistance R. The power rail at the top is at voltage $V_+ = 5\,\mathrm{V}$. What are the other four voltages, V_1 to V_4?

To answer this question we use Ohm's law and the Kirchhoff current law, which says that the total net current flow out of (or into) any junction in a circuit must be zero. Thus for the junction at voltage V_1, for instance, we have

$$\frac{V_1 - V_2}{R} + \frac{V_1 - V_3}{R} + \frac{V_1 - V_4}{R} + \frac{V_1 - V_+}{R} = 0,$$

or equivalently

$$4V_1 - V_2 - V_3 - V_4 = V_+.$$

a) Write similar equations for the other three junctions with unknown voltages.

b) Write a program to solve the four resulting equations using Gaussian elimination and hence find the four voltages (or you can modify a program you already have, such as the program `gausselim.py` in Example 6.1).

a)

Sabemos que por la la ley de corriente de Kirchhoff, la suma de las corrientes en una malla debe ser 0. Ya tenemos la ecuacion para la unión a voltaje V_1, asi que para el resto tenemos:

$$\frac{V_2 - V_1}{R} + \frac{V_2}{R} + \frac{V_2 - V_4}{R} = 0 \quad \rightarrow \quad -V_1 + 3V_2 - V_4 = 0 \tag{1}$$

$$\frac{V_3 - V_1}{R} + \frac{V_3 - V_4}{R} + \frac{V_3 - V_+}{R} = 0 \quad \rightarrow \quad -V_1 + 3V_3 - V_4 = V_+ \tag{2}$$

$$\frac{V_4 - V_1}{R} + \frac{V_4 - V_2}{R} + \frac{V_4}{R} + \frac{V_4 - V_3}{R} = 0 \quad \rightarrow \quad -V_1 - V_2 - V_3 + 4V_4 = 0 \tag{3}$$

[1]:
```
from numpy import array,empty,copy
import numpy as np
```

[2]:
```
def gausselim(A,v,N,pivot=False):


    # Gaussian elimination
    for m in range(N):
        #pivoting
        if pivot == True:
            if A[m,m]==0:
                B=copy(A)
                maxm= np.max(abs(B),axis=0)
                maxi=np.argmax(maxm)
                A[m,:],A[maxi,:]=B[maxi,:],B[m,:]
                v2=copy(v)
                v[m],v[maxi]=v2[maxi],v2[m]

        # Divide by the diagonal element
        div = A[m,m]
        A[m,:] /= div
        v[m] /= div

        # Now subtract from the lower rows
        for i in range(m+1,N):
            mult = A[i,m]
            A[i,:] -= mult*A[m,:]
            v[i] -= mult*v[m]

    # Backsubstitution
    x = empty(N,float)
    for m in range(N-1,-1,-1):
        x[m] = v[m]
        for i in range(m+1,N):
            x[m] -= A[m,i]*x[i]
```

```
    return x
```

b)

[3]:
```
 A = array([[ 4, -1, -1, -1 ],
            [ -1, 3,  0, -1 ],
            [ -1, 0,  3, -1 ],
            [ -1, -1, -1, 4 ]],float)
v= array([5 , 0 ,5 ,0],float)
N = len(v)
print("A= \n",A)
print("\nv =",v)
```

```
A=
 [[ 4. -1. -1. -1.]
 [-1.  3.  0. -1.]
 [-1.  0.  3. -1.]
 [-1. -1. -1.  4.]]

v = [5. 0. 5. 0.]
```

[4]:
```
x=gausselim(A,v,N)
print("El vector x es \nx=",x)
```

```
El vector x es
x= [3.         1.66666667 3.33333333 2.        ]
```

Exercise 6.2:

a) Modify the program `gausselim.py` in Example 6.1 to incorporate partial pivoting (or you can write your own program from scratch if you prefer). Run your program and demonstrate that it gives the same answers as the original program when applied to Eq. (6.1)

b) Modify the program to solve the equations in (6.17) and show that it can find the solution to these as well, even though Gaussian elimination without pivoting fails.

a)

[5]:
```
 A = array([[ 4, -1, -1, -1 ],
            [ -1, 3,  0, -1 ],
            [ -1, 0,  3, -1 ],
            [ -1, -1, -1, 4 ]],float)
v= array([5 , 0 ,5 ,0],float)
N = len(v)
print("A= \n",A)
print("\nv =",v)
```

```
A=
 [[ 4. -1. -1. -1.]
 [-1.  3.  0. -1.]
 [-1.  0.  3. -1.]
 [-1. -1. -1.  4.]]

v = [5. 0. 5. 0.]
```

[6]:
```
x=gausselim(A,v,N,pivot=True)
print("x con pivot\n",x)
```

```
x con pivot
 [3.         1.66666667 3.33333333 2.        ]
```

b)

Ecuación 6.17 es

$$\begin{pmatrix} 0 & 1 & 4 & 1 \\ 3 & 4 & -1 & -1 \\ 1 & -4 & 1 & 5 \\ 2 & -2 & 1 & 3 \end{pmatrix} \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix} = \begin{pmatrix} -4 \\ 3 \\ 9 \\ 7 \end{pmatrix}$$

[7]:
```
 A = array([[ 0, 1, 4, 1 ],
            [ 3, 4,-1,-1 ],
            [ 1,-4, 1, 5 ],
            [ 2,-2, 1, 3 ]],float)
v= array([-4 , 3 ,9 ,7],float)
N = len(v)
print("A= \n",A)
print("\nv =",v)
```

```
A=
 [[ 0.  1.  4.  1.]
 [ 3.  4. -1. -1.]
 [ 1. -4.  1.  5.]
 [ 2. -2.  1.  3.]]

v = [-4.  3.  9.  7.]
```

[8]:
```
x=gausselim(A,v,N,pivot=True)
print("x con pivot\n",x)
```

```
x con pivot
 [ 1.61904762 -0.42857143 -1.23809524  1.38095238]
```

Exercise 6.4: Write a program to solve the resistor network problem of Exercise 6.1 on page 220 using the function `solve` from `numpy.linalg`. If you also did Exercise 6.1, you should check that you get the same answer both times.

[9]:
```
 A = array([[ 4, -1, -1, -1 ],
            [ -1, 3,  0, -1 ],
            [ -1, 0,  3, -1 ],
            [ -1, -1, -1, 4 ]],float)
v= array([5 , 0 ,5 ,0],float)
N = len(v)
print("A= \n",A)
print("\nv =",v)
```

```
A=
 [[ 4. -1. -1. -1.]
 [-1.  3.  0. -1.]
 [-1.  0.  3. -1.]
 [-1. -1. -1.  4.]]

v = [5. 0. 5. 0.]
```

[10]:
```
from numpy.linalg import solve
x = solve(A,v)
print("x con solve\n",x)
```

```
x con solve
 [3.         1.66666667 3.33333333 2.        ]
```