Untitled

October 5, 2017

1 10. Inversion of a triagonal matrix

1.1 a)

In [1]:
```
function thomasalgo(d,a,b,y)
    N=length(d)
    #A=a' and Y=y''
    A=Array{Float64}(N)
    Y=Array{Float64}(N)
    x=Array{Float64}(N)
    #timestep1
    A[1]=a[1]/d[1]
    Y[1]=y[1]/d[1]
    #steps 2 to N-1
    for i in 2:(N-1)
        A[i]=a[i]/(d[i]-b[i]*A[i-1])
        Y[i]=(y[i]-b[i]*Y[i-1])/(d[i]-b[i]*A[i-1])
    end
    #N-th value
    Y[N]=(y[N]-b[N]*Y[N-1])/(d[N]-b[N]*A[N-1])

    #calculating the x-Vector
    x[N]=Y[N]
    for i in (N-1):-1:1
       x[i]=Y[i]-A[i]*x[i+1]
    end
    return(x)
end
```

Out[1]:
```
thomasalgo (generic function with 1 method)
```

1.2 b)

In [2]:
```
A=[2.04 -1 0 0;
-1 2.04 -1 0;
0 -1 2.04 -1;
0 0 -1 2.04]
#vectornotation
```

```
Y=[40.8;0.8;0.8;200.8]
#array notation
d=[2.04 2.04 2.04 2.04]
a=[-1 -1 -1 0.]
b=[0. -1 -1 -1]
y=[40.8 0.8 0.8 200.8]
thomassolution=thomasalgo(d,a,b,y)
println("Thomasalgo:",thomassolution)
Xtest=inv(A)*Y
println("Matrixinversion:",Xtest)
```

```
Thomasalgo:[65.9698,93.7785,124.538,159.48]
Matrixinversion:[65.9698,93.7785,124.538,159.48]
```

1.3 c)

In [3]:
```
function thomasalgo2(gamma,kronecker,epsilon,N)
    A=Array{Float64}(N)
    Y=Array{Float64}(N)
    x=Array{Float64}(N)
    #timestep1
    A[1]=kronecker/gamma
    Y[1]=1.
    #steps 2 to N-1
    for i in 2:(N-1)
        Y[i]=(gamma^i-epsilon*Y[i-1])/(gamma-epsilon*A[i-1])
        A[i]=kronecker/(gamma-epsilon*A[i-1])
    end
    #N-th value
    Y[N]=(gamma^N-epsilon*Y[N-1])/(gamma-epsilon*A[N-1])

    #calculating the x-Vector
    x[N]=Y[N]
    for i in (N-1):-1:1
       x[i]=Y[i]-A[i]*x[i+1]
    end
    return(x)
end
```

Out[3]: thomasalgo2 (generic function with 1 method)

In [4]:
```
#create the matrix to test thomasalgo2
function diagonalmatrix(gamma,kronecker,epsilon,N)
    gammavector=ones(N)*gamma
    kroneckervector=ones(N-1)*kronecker
    epsilonvector=ones(N-1)*epsilon
    matrix=diagm(gammavector,0)+diagm(epsilonvector,-1)+diagm(kroneckervector,1)
end
```

```
Out[4]: diagonalmatrix (generic function with 1 method)
```

```
In [5]: gamma=2.
        kronecker=-3.
        epsilon=-1.
        N=10
        #generate y-vector
        Y=ones(N)
        for i in 1:N
            Y[i]=Y[i]*gamma^i
        end
        println("The Matrix: ")
        A=diagonalmatrix(gamma,kronecker,epsilon,N)
```

```
The Matrix:
```

```
Out[5]: 10×10 Array{Float64,2}:
          2.0  -3.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
         -1.0   2.0  -3.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
          0.0  -1.0   2.0  -3.0   0.0   0.0   0.0   0.0   0.0   0.0
          0.0   0.0  -1.0   2.0  -3.0   0.0   0.0   0.0   0.0   0.0
          0.0   0.0   0.0  -1.0   2.0  -3.0   0.0   0.0   0.0   0.0
          0.0   0.0   0.0   0.0  -1.0   2.0  -3.0   0.0   0.0   0.0
          0.0   0.0   0.0   0.0   0.0  -1.0   2.0  -3.0   0.0   0.0
          0.0   0.0   0.0   0.0   0.0   0.0  -1.0   2.0  -3.0   0.0
          0.0   0.0   0.0   0.0   0.0   0.0   0.0  -1.0   2.0  -3.0
          0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0  -1.0   2.0
```

```
In [6]: thomassolution=thomasalgo2(gamma,kronecker,epsilon,N)
        println("Thomasalgo:",thomassolution)
        Xtest=inv(A)*Y
        println("Matrixinversion:",Xtest)
```

```
Thomasalgo:[-1.02182e5,-68122.3,-11355.4,15134.5,13869.5,4190.81,-1850.62,-2673.35,-1250.69,-1
Matrixinversion:[-1.02182e5,-68122.3,-11355.4,15134.5,13869.5,4190.81,-1850.62,-2673.35,-1250.
```

```
In [ ]:
```

```
In [ ]:
```

c) step 1

$$a_1' = \frac{\delta}{\gamma} \qquad y_1'' = \frac{\gamma}{\gamma} = 1$$

step 2 for 2 to $n-1$

$$a_i' = \frac{\delta}{\gamma - \epsilon a_{i-1}'} \quad , \quad y_i'' = \frac{\gamma' - \epsilon y_{i-1}''}{\gamma - \epsilon a_{i-1}'}$$

step 4

$$\boxed{x_n = y_n''}$$

for $n-1$ to 1

$$x_i = y_i'' - a_i' \cdot x_{i+1}$$