Conor Curry - MATH 1080 - Homework 6

February 13, 2017

1 Function for Computing FOR; Jacobi Iteration

In [1]:
```
function jacobi(N, x0, xf, g0, gf, f, itermax, tol=0)
    h = (xf-x0) / (N+1)
    u = [f(i*h) for i in 0:(N+1)]
    u[1] = g0
    u[end] = gf
    unew = [0. for _ in u]
    res = [0. for _ in u]
     = [0. for _ in u]
    b = [0. for _ in u]

    for iter in 1:itermax
        for i in 2:N+1
            au = u[i-1] + u[i+1]
            b[i] = f((i-1)*h) * (h^2)
            unew[i] = 1/2 * (b[i] + au)
            res[i] = b[i] + au - (2u[i])
             = unew[i] - u[i]
        end
        if tol > 0 && norm(res) < tol * norm(b) && norm() < tol * norm(unew)
            println("Solution converged after $iter iterations")
            return true, u
        end
        #swap references to avoid copying data/busying garbage collector
        u, unew = unew, u
    end
    if tol > 0
        println("Convergence Failed")
    end
    return false, u
end
```

Out[1]:
```
jacobi (generic function with 2 methods)
```

1.1 Checking example solution...

```
In [2]: N=4
        x0=0
        xf=1
        g0=gf=0
        conv, u = jacobi(N, x0, xf, g0, gf, (x)->x/5, 50)
        u
```

```
Out[2]: 6-element Array{Float64,1}:
         0.0
         0.00640143
         0.0112029
         0.0128023
         0.00960181
         0.0
```

Note: I'm pretty sure there's an error in the example. making the solutions off by a factor of 25^2. So we'll do a small correction to verify the example's solution.

```
In [3]: u .* 25^2
```

```
Out[3]: 6-element Array{Float64,1}:
         0.0
         4.00089
         7.00183
         8.00145
         6.00113
         0.0
```

2 Running code with convergence verification

This utilizes the optional argument for tol, set to 10e-8

```
In [4]: N = 4
        conv, u = jacobi(N, x0, xf, g0, gf, (x)->x/5, 1000, 10e-8)
        u
```

```
Solution converged after 89 iterations


Out[4]: 6-element Array{Float64,1}:
         0.0
         0.0064
         0.0112
         0.0128
         0.0096
         0.0
```

2.0.1 So we can see that it took 89 iterations to converge with a 10e-8 tolerance!

3 Running again with h=1/200, f(x) = x/200

```
In [5]: #Setting N so we get an h=1/200
        N = ((xf-x0) * 200) - 1
        conv, u = jacobi(N, x0, xf, g0, gf, (x)->x/200, 1000000, 10e-8)
        u
```

```
Solution converged after 149614 iterations
```

```
Out[5]: 201-element Array{Float64,1}:
         0.0
         4.16656e-6
         8.3325e-6
         1.24972e-5
         1.666e-5
         2.08203e-5
         2.49775e-5
         2.91309e-5
         3.328e-5
         3.74241e-5
         4.15625e-5
         4.56947e-5
         4.982e-5

         8.42428e-5
         7.71875e-5
         7.00134e-5
         6.272e-5
         5.53066e-5
         4.77725e-5
         4.01172e-5
         3.234e-5
         2.44403e-5
         1.64175e-5
         8.27094e-6
         0.0
```

3.0.1 Yikes, that took 149614 iterations. This is much worse, and we didn't even change the step size by very much!

4 Computing the Relative Two Norm

```
In [6]: anal_soln(x) = ((-x^3)/1200) + (1/1200)x
```

```
Out[6]: anal_soln (generic function with 1 method)
```

```
In [7]: h = (xf-x0) / (N+1)
        xs = [i*h for i in 0:(N+1)]
        diffs = [anal_soln(xs[i]) - u[i] for i in 1:N+2]
        norm(diffs) / norm([anal_soln(xs[i]) for i in 1:N+2])

Out[7]: 8.466924187410467e-8
```

4.0.1 Looks pretty small!