Nick Largey

Scientific Computing

X7

using LinearAlgebra

using MatrixMarket

using Random

# 5.1.2

function Jacobi(A, b, tol)

j\_t = time()

iter = 0

x = zeros(size(b,1))

n = size(A,1)

x\_new = zeros(size(b,1))

t = tol

while n > t && iter < 50

t = tol\*norm(x, Inf)

x\_new .= Diagonal(A)\(-(tril(A,-1)+triu(A,1))\*x + b)

n = norm(x\_new - x, Inf)

x .= x\_new

iter += 1

end

J\_elap = time()-j\_t

return x, iter, J\_elap

end

# 5.1.4

A = [4 1 -1;

2 7 1;

1 -3 12]

b = [19; 3; 31]

jacobi\_x, jacobi\_iter, j\_time = Jacobi(A, b, 1e-6)

display(jacobi\_x)

display(jacobi\_iter)

# 5.1.6

function GaussSeidel(A, b, tol)

gst = time()

iter = 0

x = zeros(size(b,1))

n = size(A,1)

x\_new = zeros(size(b,1))

t = tol

while n > t && iter < 50

t = tol\*norm(x, Inf)

x\_new .= (Diagonal(A) +tril(A,-1))\(-triu(A,1)\*x + b)

n = norm(x\_new - x, Inf)

x .= x\_new

iter += 1

end

gs\_elap = time()-gst

return x, iter, gs\_elap

end

GS\_x, GS\_iter, gs\_time = GaussSeidel(A, b, 1e-6)

display(GS\_x)

display(GS\_iter)

println("Jacobi time: ", j\_time, " Gauss-Seidel time: ", gs\_time)

"""

The tic() and toc() methods have been depreciated in Julia. It is now suggested to use the time() function.

When I ran it on both the Jacobi and Gauss-Seidel functions, Gauss-Seidel performed about 10x faster.

"""

# 5.1.9

function SOR(A, b, w, tol)

SORt = time()

iter = 0

x = zeros(size(b,1))

n = size(A,1)

x\_new = zeros(size(b,1))

t = tol

while n > t && iter < 50

t = tol\*norm(x, Inf)

x\_new .= (Diagonal(A) + (w\*tril(A,-1))) \ (((1-w)\*Diagonal(A)-(w\*triu(A,1)))\*x + (w\*b))

n = norm(x\_new - x, Inf)

x .= x\_new

iter += 1

end

SOR\_elap = time()-SORt

return x, iter, SOR\_elap

end

SOR\_x, SOR\_iter, SOR\_t = SOR(A, b, .5, 1e-6)

display(SOR\_x)

display(SOR\_iter)

println(SOR\_t)

"""

time() for SOR seems to be having trouble displaying acurate measurements, but the most likely I saw

was about another 10x speed up from Gauss-Seidel.

"""

# 5.2.4 # Doesn't work....

function steepestdescent(A, b, x0, tol)

iter = 0

r = size(A,1)

t = tol\*norm(r, Inf)

n = size(A,1)

x = x0

alpha = size(A,1)

while n > t && iter < 50

t = tol\*norm(r, Inf)

r .= b - (A\*x)

alpha .= (r.\*transpose(r))\((A\*r).\*transpose(r))

x\_new .= x + alpha\*r

n = norm(x - r, Inf)

x .= x\_new

iter += 1

end

return x, iter

end

SD\_x, SD\_iter = steepestdescent(A, b, b, 1e-6)

display(SD\_x, SD\_iter)

M = MatrixMarket.mmread("can\_\_\_24.mtx")

b = rand(-10.0:10.0, size(M,1))

jacobi\_x, jacobi\_iter, j\_time = Jacobi(M, b, 1e-6)

GS\_x, GS\_iter, gs\_time = GaussSeidel(M, b, 1e-6)

SOR\_x, SOR\_iter, SOR\_t = SOR(M, b, .5, 1e-6)

println("Jacobi x: ", jacobi\_x, "\nGS x: ", GS\_x, "\nSOR x: ", SOR\_x)

println("Jacobi iterations: ", jacobi\_iter, "\nGS iterations: ", GS\_iter, "\nSOR x: ", SOR\_iter)

println("Jacobi time: ", j\_time, "\nGS time: ", gs\_time, "\nSOR time: ", SOR\_t)