Homework 6, ME3215 Spring 2022

Table of Contents

Iterative methods for linear & nonlinear systems of equations

HW6_3: steady-state chemical concentration in tanks using Gauss-Seidel

```
%given: input concentrations, g/L
c01=1;c02=2;c03=1;c04=2;c06=1;c07=1;c08=1;
%given: flow in, L/min
r01=3; r02=3; r03=3; r04=3; r06=2; r07=2; r08=3;
%given: flow rate between tanks, L/min
r12=6;r15=2;r23=5;r26=5;r36=8;r41=5;r45=1;r48=1;
r52=1;r58=2;r59=1;r69=15;r74=4;r87=2;r89=3;
%given: flow out, L/min
r=r01+r02+r03+r04+r06+r07+r08;

c = zeros(1,9); %first guess
ea=1;es=0.0005; %5 sig figs accracy
while max(ea) > es
    oldc = c;
    %update estimate
    c(1) = (r01*c01+r41*c(4))/(r12+r15);
    c(2) = (r02*c02+r12*c(1)+r52*c(5))/(r23+r26);
    c(3) = (r03*c03+r23*c(2))/r36;
    c(4) = (r04*c04+r74*c(7))/(r41+r48+r45);
    c(5) = (r15*c(1)+r45*c(4))/(r52+r58+r59);
    c(6) = (r06*c06+r26*c(2)+r36*c(3))/r69;
    c(7) = (r07*c07+r87*c(8))/r74;
    c(8) = (r08*c08+r58*c(5)+r48*c(4))/(r87+r89);
    c(9) = (r59*c(5)+r69*c(6)+r89*c(8))/r;
    %compute errors
    ea = abs((c-oldc)./c)*100;
end

fprintf('The final concentration in each tank\n')
fprintf('tank %d = %.4f g/L\n',[1:9;c])
```

```
The final concentration in each tank
tank 1 = 1.3252 g/L
tank 2 = 1.4994 g/L
tank 3 = 1.3121 g/L
tank 4 = 1.5203 g/L
tank 5 = 1.0427 g/L
tank 6 = 1.3329 g/L
tank 7 = 1.1606 g/L
tank 8 = 1.3211 g/L
tank 9 = 1.3158 g/L
```

HW6_4 Solving a larger system of equations using Newton-Raphson

```
clear;clc;

%initialization
x=1;y=1;z=1;
es=0.005;ea=1;it=0;

%stop when largest ea is at the goal
while max(ea) > es && it < 25
    it=it+1;
    old=[x; y; z]; %save previous
    %construct f and J by pluging in x,y,z values
    f=[x^3-10*x+y-z+3;
        y^3+10*y-2*x-2*z-5;
        x+y-10*z+2*sin(z)+5];
    J = [3*x^2-10 1 -1;
         -2 3*y^2+10 -2;
         1 1 -10+2*cos(z)];
    new = old - J\f;
    ea = abs((new-old)./new)*100; %errors all at once
    x=new(1);y=new(2);z=new(3); %extract from X vector for next
 iteration
end
if it==25
    warning('convergence problem')
else
    fprintf('x = %.4f   y = %.4f   z = %.4f\n',x,y,z)
end
```

```
x = 0.2970   y = 0.6748   z = 0.7307
```

Another way to do this with function handles

```
%First set up some handles to help
%define the system of root-finding functions
%We can actually make a function handle array!

clear;clc;

f=@(x,y,z)[x^3-10*x+y-z+3;
           y^3+10*y-2*x-2*z-5;
           x+y-10*z+2*sin(z)+5];
%ditto for the Jacobian
J = @(x,y,z)[3*x^2-10 1 -1;
            -2 3*y^2+10 -2;
             1 1 -10+2*cos(z)];
```

```
%initialization
x=1;y=1;z=1;
es=0.005;ea=1;it=0;

%stop when largest ea is at the goal
while max(ea) > es && it < 25
    it=it+1;
    old=[x;y;z]; %place into array and save previous
    new = old - J(x,y,z)\f(x,y,z); %plug in x,y,z values
    ea = abs((new-old)./new)*100;  %errors all at once
    x=new(1);y=new(2);z=new(3); %extract from X vector for next
 iteration
end
if it==25
    warning('convergence problem');
else
    fprintf('x = %.4f   y = %.4f   z = %.4f\n',x,y,z)
end
```

```
x = 0.2970   y = 0.6748   z = 0.7307
```

Published with MATLAB® R2020a