Juan Alejandro Ornaza
CS375: HW6
October 6 2021

1. Explain how the component-wise update formula for Jacobi given on the course slides is equivalent to the vector-update version. That is, for the Jacobi algorithm show that the vector update version

$$x^{(k)} = D^{-1}(L+U)x^{(k-1)} + D^{-1}b$$

is the same as the component-wise update,

$$x_i^{(k)} = -\sum_{j=1, j\neq i}^{n} \left(\frac{a_{ij}}{a_{ii}}\right) x_j^{(k-1)} + \frac{b_i}{a_{ii}}$$

Let's look at the v-th component of the vector update version of Jacobi

$$x^{(k)} = \left[0 \cdots \frac{1}{a_{ii}} \cdots 0\right] \begin{bmatrix} 0 & \cdots & -a_{1n} \\ -a_{21} & & \vdots \\ -a_{i1} & 0 & -a_{in} \\ \vdots & & \vdots \\ -a_{n1} & \cdots & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1^{(k-1)} \\ \vdots \\ x_n^{(k-1)} \end{bmatrix} + \left[0 \cdots \frac{1}{a_{ii}} \cdots 0\right] \begin{bmatrix} b_1 \\ \vdots \\ b_i \\ \vdots \\ b_n \end{bmatrix}$$

$$x^{(k)} = \left[\frac{-a_{i1}}{a_{ii}}, \frac{-a_{i2}}{a_{ii}}, \underbrace{0}_{a_{ii}=0}, \cdots, \frac{-a_{in}}{a_{ii}}\right] \begin{bmatrix} x_1^{(k-1)} \\ \vdots \\ x_n^{(k-1)} \end{bmatrix} + \frac{b_i}{a_{ii}}$$

$$x^{(k)} = -\sum_{\substack{j=1 \\ j\neq i}}^{n} \left(\frac{a_{ij}}{a_{ii}}\right) x_j^{(k-1)} + \frac{b_i}{a_{ii}}$$

It is the same

Contents

Homework 6

Juan Alejandro Ormaza October 6 2021

```
clear all;clc; close all;
format LONG E
```

Problem 2.b

```
P=20;
A=gallery('poisson',P);
b = ones(P^2,1);

% Since both algorithms should be the same, I am going to use norm 2 to
% figure the difference between my_jacobi and my_vector_jacobi.

error = norm(my_jacobi(A,b,100)-my_vector_jacobi(A,b,100),2);
fprintf("the size of the differnce between my_jacobi and my_vector_jacobi is %2.8f\n",error);
```

```
the size of the differnce between my_jacobi and my_vector_jacobi is 0.00000000
```

Problem 2.c

```
% Since my_jacobi takes too long with p=20 I am going to use p=12 to
% calculate the time my algorithms take to run.

P=12;
A=gallery('poisson',P);
b = ones(P^2,1);

tic
    x1=my_vector_jacobi(A,b,100);
t1=toc;
fprintf("my vector jacobi takes %2.8f seconds\n",t1);

tic
    x2=my_jacobi(A,b,100);
t2=toc;
fprintf("my component-wise jacobi takes %10.8f seconds\n",t2);

fprintf("my_vector_jacobi is %10.8f times faster than my_jacobi\n", t2/t1);
```

```
my vector jacobi takes 0.00596608 seconds
my component-wise jacobi takes 9.96381930 seconds
my_vector_jacobi is 1670.07721884 times faster than my_jacobi
```

Problem 2.d

```
P=[ 10, 20, 40, 80, 160];
error=zeros(size(P,2),1);

for i=1:size(P,2)
    A=gallery('poisson',P(i));
    b = ones(P(i)^2,1);
    xtrue=A\b;
    x=my_vector_jacobi(A,b,100);
    error(i)=norm(xtrue-x,2)/norm(xtrue,2);
end
```

Problem 3.c

```
tot_it = [50, 100, 200, 400, 800, 1600];
error=zeros(size(tot_it,2),3);

for i=1:size(tot_it,2)
    A=gallery('poisson',160);
    b = ones(160^2,1);
    xtrue=A\b;
    x_jacobi=my_vector_jacobi(A,b,tot_it(i));
    x_CG=my_CG(A,b,tot_it(i));
    x_gauss=my_gauss_siedel(A,b,tot_it(i));
    error(i,1)=norm(xtrue-x_jacobi,2)/norm(xtrue,2);
    error(i,2)=norm(xtrue-x_gauss,2)/norm(xtrue,2);
    error(i,3)=norm(xtrue-x_CG,2)/norm(xtrue,2);
end

fprintf("Num Iterations\tError Jacobi\tError GS\tError CG\n");
for i=1:size(tot_it,2)
fprintf("%4.0f\t\t %2.5f\t%2.5f\t\t%2.5f\t\n",tot_it(i),error(i,1),error(i,2),error(i,3));
end
```

```
Num Iterations  Error Jacobi    Error GS        Error CG
  50             0.99012        0.98039         0.22322
 100             0.98039        0.96131         0.00849
 200             0.96130        0.92453         0.00000
 400             0.92452        0.85577         0.00000
 800             0.85574        0.73419         0.00000
1600             0.73415        0.54123         0.00000
```

Problem 4.a

```
P=10;
A=gallery('poisson',P);
b = ones(P^2,1);
```

```
nonZeros = nnz(A)/size(A,1);

fprintf("The number of nonzeros in A is %3.6f\n", nonZeros);
fprintf("As P increases the number of nonzeros approaches 5.")
```

```
The number of nonzeros in A is 4.600000
As P increases the number of nonzeros approaches 5.
```

Problem 4.b

```
fprintf("see in attached written solutions.\n")
```

```
see in attached written solutions.
```

Published with MATLAB® R2021a

4.b

based on the answer of part a we know

$m = 5$

thus Big-Oh is $O(m^2 n) = O(25n)$

additionally, we know that component-wise does not take advantage of sparsity because it goes through each element of the matrix.

Vector-update does take advantage of the speed that sparsity provides because it uses qlr to solve for each line/row

4.c $m = 2p = 2\sqrt{n}$

Since the solution for an m banded system takes $O(m^2 n)$ to solve.

We know that the system with $m = 2\sqrt{n}$ will take: $O(4n^2)$ or essentially $O(n^2)$

with $n = \alpha = 2$

4.d

if $n = 100$

let's assume i is the number of iterations

1) we want to find the moment when Jacobi becomes more expensive

$\frac{i_1}{2500} \geq \frac{i_2}{10000}$ $\quad 4 i_1 \geq i_2$

$i_1 \geq 0.25\, i_2$ $\quad$ at 4 times i_2 (number of iterations for c) Jacobi will become more expensive.

$i_1 \geq \frac{1}{4} i_2$