Reference Answers to Assignment 1 of Numerical Algorithms with Case Studies

Here are TA's reference answers, not standard answers. If you have any different ideas, please feel free to let us know. Anyone who first finds out errors in this doc will receive additional rewards(up to 5) for this assigment.

Contents

Question 1

Write your own MATLAB codes to compute the product of two matrices A and B based on the definition. Test the correctness of your codes by comparing the output of your codes with that of the MATLAB * function on randomly generated matrices.

Answer: Here are some possible answers. I will only test the correctness. If you are interested, you can test the time they use for one operation. You will find that they perform different in efficiency.

```
function[ C ] = MMLoop(A,B)
[m1,n1] = size(A);
[m2,n2] = size(B);
if n1~=m2
    fprintf('Dim error\n');
    return
end
C = zeros(m1,n2);
for i=1:m1
    for j=1:n2
        for k=1:n1
            C(i,j) = C(i,j)+A(i,k)*B(k,j);
        end
    end
end
end
```

```
function[ C ] = MMMatVec(A,B)
[m1,n1] = size(A);
[m2,n2] = size(B);
if n1~=m2
    fprintf('Dim error\n');
```

```
    return
end
C = zeros(m1,n2);
for i =1:n2
    C(:,i) = A*B(:,i);
end
end

function[ C ] = MMDaxpy(A,B)
[m1,n1] = size(A);
[m2,n2] = size(B);
if n1~=m2
    fprintf('Dim error\n');
    return
end
C = zeros(m1,n2);
for i =1:n2
    for j =1:n1
        C(:,i) = C(:,i) + A(:,j)*B(j,i);
    end
end
end

function[ C ] = MMOuterdot(A,B)
[m1,n1] = size(A);
[m2,n2] = size(B);
if n1~=m2
    fprintf('Dim error\n');
    return
end
C = zeros(m1,n2);
for i =1:n1
    C = C+A(:,i)*B(i,:);
end
end

function[ C ] = MMDot(A,B)
[m1,n1] = size(A);
[m2,n2] = size(B);
if n1~=m2
    fprintf('Dim error\n');
    return
end
C = zeros(m1,n2);
for i =1:m1
    for j =1:n2
        C(i,j) = A(i,:)*B(:,j);
```

```
    end
end
end
```

Now test the correctness.

```
error = 0;
for i=1:100
    dim = randi(256,1,3);
    A = randn(dim(1:2));
    B = randn(dim(2:3));
    C = A*B;
    C1 = MMLoop(A,B);
    C2 = MMMatVec(A,B);
    C3 = MMDaxpy(A,B);
    C4 = MMOuterdot(A,B);
    C5 = MMDot(A,B);
    loss = [norm(C-C1,'fro') norm(C-C2,'fro') norm(C-C3,'fro') ...
                                    norm(C-C4,'fro') norm(C-C5,'fro')];
    re_error= loss/norm(C,'fro');
    error = max([re_error error]);
end
fprintf('Max of the relative errors is %g.\n',error);
fprintf('Machine presicion is %g.\n',eps);
```

```
Max of the relative errors is 5.5915e-16.
Machine presicion is 2.22045e-16.
```

Question 2

Write your own MATLAB codes to compute the one-norm, two-norm and ∞ -norm of a vector x , respectively, based on the definitions. Test the correctness of your codes by comparing the output of your codes with that of the MATLAB norm function on randomly generated vectors.

Here is one possible answer:

```
function [ y ] = VecNorm( x, p)
[m,n] = size(x);
if m~=1&&n~=1
    fprintf('Dim error\n');
    return
end
if p == inf
    y = max(abs(x));
```

```
elseif p==0
    y = sum(p~=0);
else
    y = sum(abs(x).^p)^(1/p);
end
end
```

Now test the correctness.

```
error = 0;
for i=1:100
    x = randn(128,1);
    Norm_vec1 = [VecNorm(x,1) VecNorm(x,2) VecNorm(x,inf)];
    Norm_vec2 = [norm(x,1) norm(x,2) norm(x,inf)];
    re_error = abs(Norm_vec1 - Norm_vec2)./Norm_vec2;
    error = max([re_error error]);
end
fprintf('Sum of the errors is %g.\n',error);
```

```
Sum of the errors is 9.38954e-16.
```

Question 3

Write your own MATLAB codes to compute the one-norm and ∞-norm of a matrix A, respectively, based on the definitions. Test the correctness of your codes by comparing the output of your codes with that of the MATLAB norm function on randomly generated matrices.

Here is one possible answer:

```
function [ y ] = MatNorm( A, p)
if p == 1
    y = max(sum(abs(A)));
elseif p == inf
    y = max(sum(abs(A')));
else
    fprintf('Cannot compute');
    return
end
```

Now test the correctness.

```
error = 0;
for i = 1:100
    A = randn(1024);
```

```
    re_error = [abs(norm(A,1)-MatNorm(A,1))/norm(A,1) ...
        abs(norm(A,inf)-MatNorm(A,inf))/norm(A,inf)];
    error = max([error re_error]);
end
fprintf('Max of the relative errors is %g.\n',error);
```

```
Max of the relative errors is 1.95393e-15.
```