OC680 Homework #1 Due Tuesday Jan. 16

1. **First derivative matrix**
   Let

$$\begin{aligned} f'_1 &= Af_1 + Bf_2 + Cf_3 \\ f'_N &= Af_N + Bf_{N-1} + Cf_{N-2} \end{aligned}$$

   (where the constants $A - C$ have different values in each formula).

   (a) Find expressions for the constants in each formula so that the error is proportional to Δ^2.

   (b) Combine your results from part (a) with the example from class to define a matrix D such that

$$f'_i = D_{ij} f_j \; ; \quad i = 1, 2, ..., N.$$

   (c) Type in the Matlab function **ddz** printed below. Verify that is corresponds to the finite difference approximation to the first derivative that you defined in part (b). Then type in the script **ddz_err** (in a separate m-file). This script tests the accuracy of **ddz** for a given function ($f = z^5$ in this case). Run the script to demonstrate that the error is second order in Δ.

   (d) Try it with a few other functions to see if the result is generally valid. (Two is enough.) Now try it for the case $f = z^2$. Can you make sense of the result?

2. **Second derivative matrix**

   Repeat the analyses above for the second derivative. To begin with, assume that:

$$\begin{aligned} f''_i &= Af_{i-1} + Bf_i + Cf_{i+1} \quad ; \quad i = 1, 2, ..., N \\ f''_1 &= Af_1 + Bf_2 + Cf_3 + Df_4 \\ f''_N &= Af_N + Bf_{N-1} + Cf_{N-2} + Df_{N-3} \end{aligned}$$

   (a) Find expressions for the constants A, B ... in each formula so that the error is proportional to Δ^2 .

   (b) Write a Matlab function called **ddz2**, similar to **ddz**, that computes a second derivative matrix using your results from part (a). Modify the script **ddz_test** so that it computes the second derivative using your function **ddz2** and tests its accuracy. Demonstrate that your approximation is accurate to second order.

3. **Differential eigenvalue problem**

   (a) Analytically determine the values of the constant λ for which the following boundary value problem has solutions:

$$f'' = \lambda f \; ; \quad f(0) = f(\pi) = 0. \tag{A.0.1}$$

   (b) Now do the same thing numerically. Start by defining a vector of equally-spaced z values z_i ; $i = 1, 2, ..., N$, such that z_0 and z_{N+1} , if they were included, would be equal to 0 and π. Use your subroutine **ddz2** from project 2 to compute the second derivative matrix for z , then replace the top and bottom rows so as to be consistent with the boundary conditions $f_0 = f_{N+1} = 0$. Set $N = 10$. The eigenvalues of your matrix should now correspond to the values of λ that you found in part (a), (at least inasmuch as the finite difference derivative you derived is accurate). Check this by using the Matlab routine **eig** to find the eigenvalues, then the routine **sort** to sort them from smallest to largest. Plot the eigenvectors corresponding to the smallest and largest[1] eigenvalues. You should find that the former is a smooth, well-resolved function, whereas the latter has a lot of poorly-resolved small-scale structure. Correspondingly, the smallest eigenvalues should match the analytical solution closely, whereas the largest will not.

   [At the end of this assignment is a sample script that you can use as you wish.]

   (c) Repeat part (b) using one-sided derivatives for the top and bottom rows instead of boundary conditions. What difference does this make to the result?

[1] in absolute value

Matlab code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function d=ddz(z)
% First derivative matrix for independent variable z.
% 2nd order centered differences.
% Use one-sided derivatives at boundaries.

% check for equal spacing
if abs(std(diff(z))/mean(diff(z)))>.000001
    disp(['ddz: values not evenly spaced!'])
    d=NaN;
    return
end

del=z(2)-z(1);N=length(z);

d=zeros(N,N);
for n=2:N-1
    d(n,n-1)=-1.;
    d(n,n+1)=1.;
end
d(1,1)=-3;d(1,2)=4;d(1,3)=-1.;
d(N,N)=3;d(N,N-1)=-4;d(N,N-2)=1;
d=d/(2*del);
return
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Script ddz_err
% Example script for OC680 Hmwk #1.
% This script tests the first derivative matrix computed in ddz. The result
% shows that the method is 2nd order in the time step grid increment.
% The assignment is to do the same for the second derivative.
%
NN=[10:10:100];
% compute error at each N
for i=1:length(NN);
    N=NN(i);

    % 0<z<1
    del(i)=1/N;
    z=[0:1:N-1]'*del(i);

    % specify test function f(z) and its (exact) derivative fp(z)
    f=z.^5;
    fp=5*z.^4;

    d=ddz(z);       % compute derivative matrix
    df=d*f;         % compute finite-difference approximation to the derivative
    err(i)=sqrt(mean((fp-df).^2)); % compute root-mean-squared error
end
```

```
% plot error vs. N
figure
loglog(del,err,'*')
xlabel('\Delta')
ylabel('ERROR')
hold on

% regress to find power law and plot
p=polyfit(log(del),log(err),1)
err_th=exp(p(2))*del.^p(1);
plot(del,err_th,'-')
ttle=sprintf('ERROR = %.2f\\Delta^{ %.2f}',exp(p(2)),p(1))
title(ttle)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% HMWK 1 Part 3
%
clear
close all

% define z values
N=10;
z=pi*[1:N]'/(N+1);

% compute derivative matrix
d=ddz2(z);
dz=z(2)-z(1);

% To use 1-sided derivatives, comment out the next two lines.
d(N,:)=0;d(N,N-1)=1/dz^2;d(N,N)=-2/dz^2;
d(1,:)=0;d(1,1)=-2/dz^2;d(1,2)=1/dz^2;

% compute eigvals & eigvecs
[v ee]=eig(d);e=diag(ee);

% sort
[~,ind] =sort(abs(e),'ascend');
e=e(ind);v=v(:,ind);

% Plot eigenvalues /i^2
% If the eigfn is well-resolved, this will be close to -1.
figure
plot([1:N],e./[1:N].^2','*','markersize',10)
xlabel('i');
ylabel('\lambda_i / i^2')
title('Is \lambda_i / i^2 = -1 ?')

% Plot first and last eigvecs.
% The first is well-resolved, and its eigval is close to -i^2.
% The last is poorly-resolved, and the eigval is not close to -i^2.
figure
subplot(1,2,1)
```

```
plot(v(:,1),z,'b*'); hold on
plot(v(:,1),z,'b')
ylabel('z')
title('First eigvec (smallest abs(eigval))')
subplot(1,2,2)
plot(v(:,end),z,'r*'); hold on
plot(v(:,end),z,'r')
title('Last eigvec (largest abs(eigval))')
```