Contents

Iterates over all helper functions to do each part of problem.

```
function[]=Math671_HW1_p7()

for i=['a','b','c']
    mesh_point_variation(i)
end
end
```

For each function in problem, run interpolations at multiple N.

```
function[]=mesh_point_variation(part)
% Running over N=4,8,16,32 per problem statement
N_input=[4 8 16 32];
xx = linspace(0,1,300);

mesh_points=length(N_input);
figure('Name',['Part ' part],'NumberTitle','off')
for i=1:mesh_points
    xj = linspace(0,N_input(i)-1,N_input(i))/N_input(i);
    [vx, vj] = problemstatementparts(part,xx,xj);
    [I1, I2,] = interpolation(N_input(i),xx,xj,vj);
    subplot(sqrt(mesh_points),sqrt(mesh_points),i)
    plot(xx,vx,'-',xj,vj,'o',xx,I1,'.-',xx,I2,'--')
    title(['N=' num2str(N_input(i))])
    if i==mesh_points
        legend('Original function','Mesh points','Unbalanced interpolation','Balanced interpolation','Location','southoutside');
    end
end
end
```

Give functions for each problem statement part.

```
function[vx, vj]=problemstatementparts(part, xx, xj)
% Parts of the problem
if part == 'a'
    % (a)
    vx = 4.*xx.*(1-xx);
    vj = 4.*xj.*(1-xj);
elseif part == 'b'
    % (b)
    vx = sin(pi.*xx);
    vj = sin(pi.*xj);
elseif part == 'c'
    % (c)
    vx = sin(2*pi.*xx);
    vj = sin(2*pi.*xj);
end
end
```

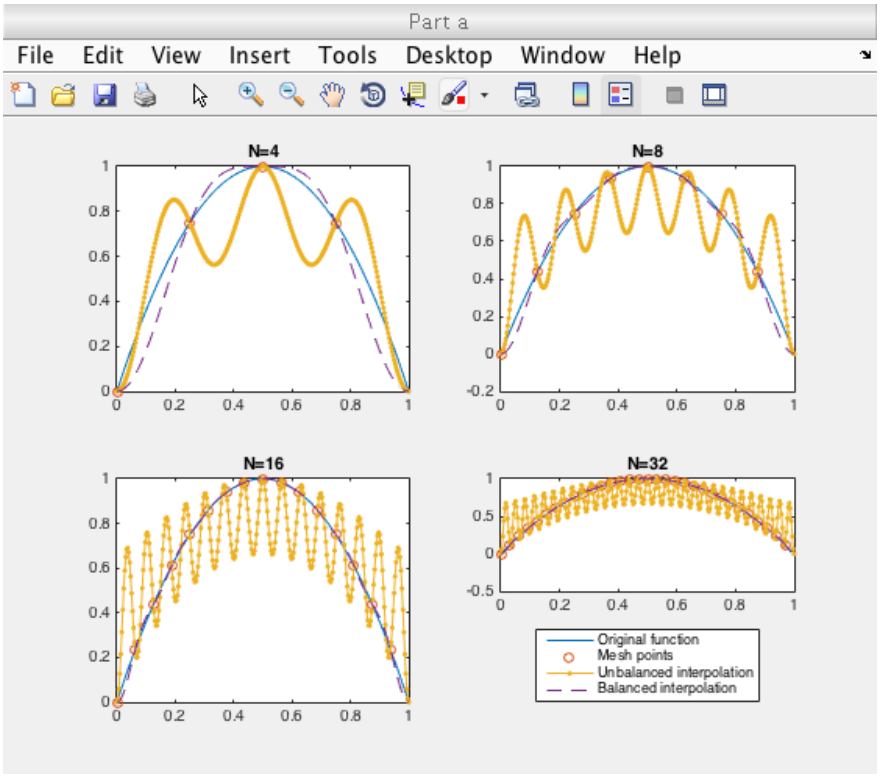
Calculate interpolations at constant # of mesh points.

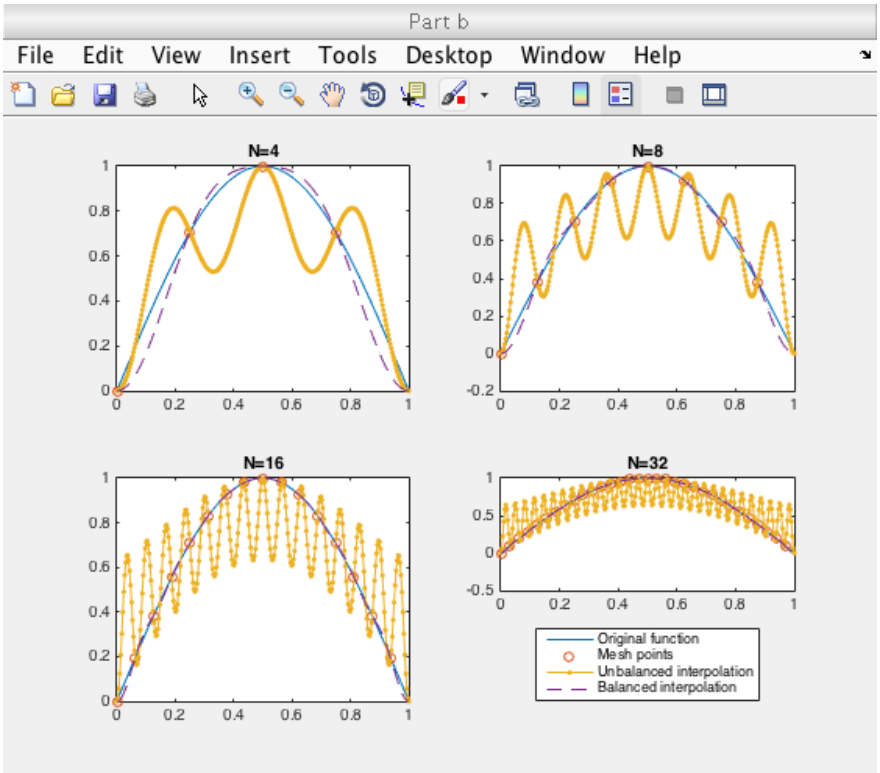
```
function [I1, I2]=interpolation(N,xx,xj,vj)
points = length(xx);

% Iterative calculation
% NOTE! The prefactor of 1/sqrt(N) is not included here
% Calculating v_hat here for interpolants
v_hat = zeros(N,N);
for j=1:N
    for n=1:N
```

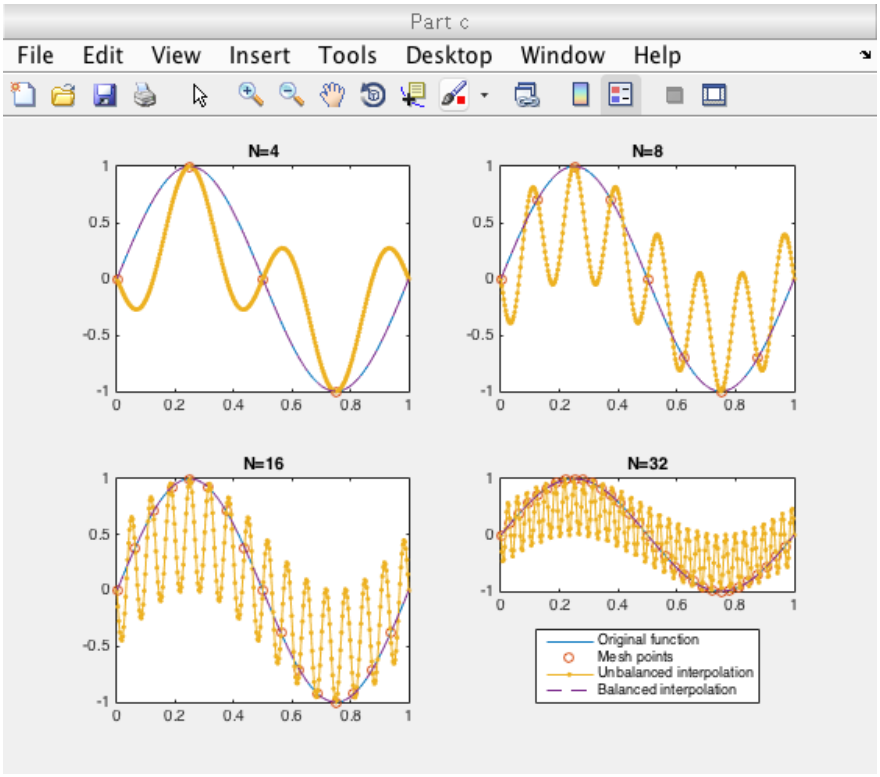
```
        v_hat(n,j) = vj(j)*exp(-2*pi*1i*(n-1)*xj(j));
    end
end
v_hat = sum(v_hat,2)/sqrt(N);

% Unbalanced interpolation
I1 = zeros(points,N);
for n=1:N
    for j=1:points
        I1(j,n) = v_hat(n)*exp(2*pi*1i*(n-1)*xx(j));
    end
end
I1 = real(sum(I1,2)/sqrt(N));

% Balanced interpolation; function for modulus defined separately below
I2 = zeros(points,N);
nspace = linspace(-N/2,N/2-1,N);
for n=1:N
    for j=1:points
        I2(j,n) = v_hat(newmod(nspace(n),N)+1)*exp(2*pi*1i*nspace(n)*xx(j));
    end
end
I2 = real(sum(I2,2)/sqrt(N));
end
```

Calculated custom n mod N method for balanced interpolation

```
function n_out=newmod(n,N)
if n>=0
    n_out=n;
else
    n_out=n+N;
end
end
```


Part a
File Edit View Insert Tools Desktop Window Help
N=4
N=8
N=16
N=32
Original function
Mesh points
Unbalanced interpolation
Balanced interpolation


Part b
File Edit View Insert Tools Desktop Window Help
N=4
N=8
N=16
N=32
Original function
Mesh points
Unbalanced interpolation
Balanced interpolation



Published with MATLAB® R2015a