hw4q1

May 9, 2020

1 HW4 Computing Problem 1

[1]:
```
%% Q1
clc; cla; clf; clear all; close all; % Initial Clean-up for multiple tries
iptsetpref('ImshowBorder', 'tight');format compact;     % Set preferences
```

[2]:
```
M = 64;
P = 64;
N = 3;
x = (0:M-1)'/M;
t = (0:P-1)/P;
```

[3]:
```
size(x)
```

```
ans =
    64     1
```

[4]:
```
size(t)
```

```
ans =
     1    64
```

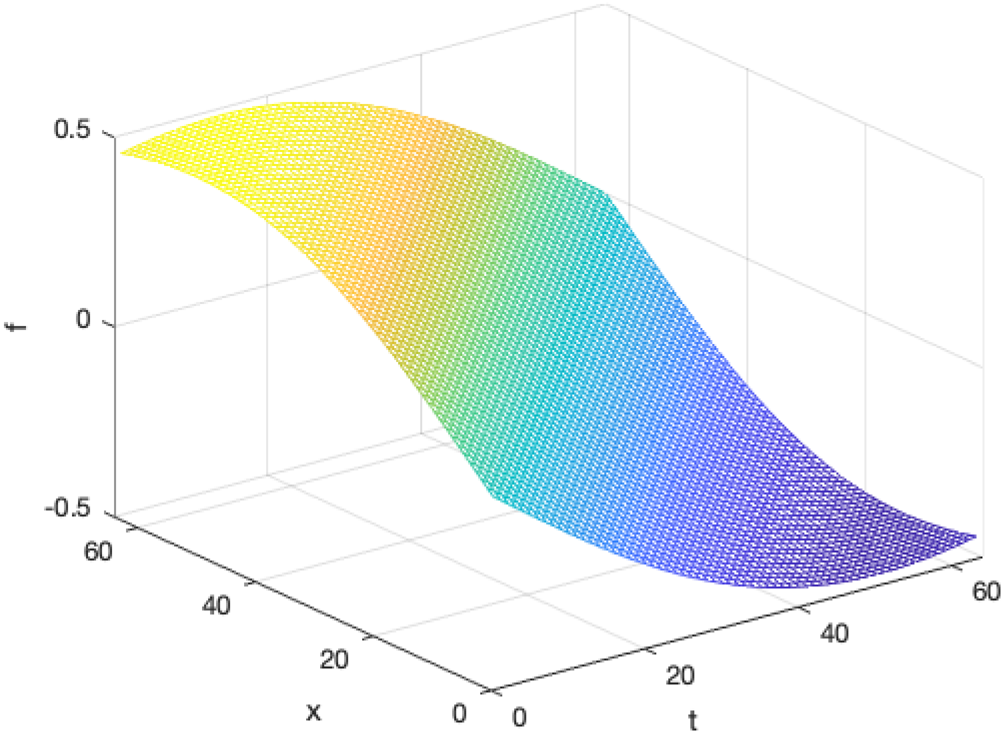
[5]:
```
f = 0;
for k = 1:N
    f = f+1/N*1/k*sin(k*(x-t));
end
```

[6]:
```
size(f)
```

```
ans =
    64    64
```

Original data

[7]:
```
mesh(f)
xlabel('t')
ylabel('x')
zlabel('f')
```



creating the mask of 10% missing

[8]:
```
m = (floor(rand(M,P)*10)>0);
```

[9]:
```
sum(m(:))/64^2
```

```
ans =
    0.9060
```

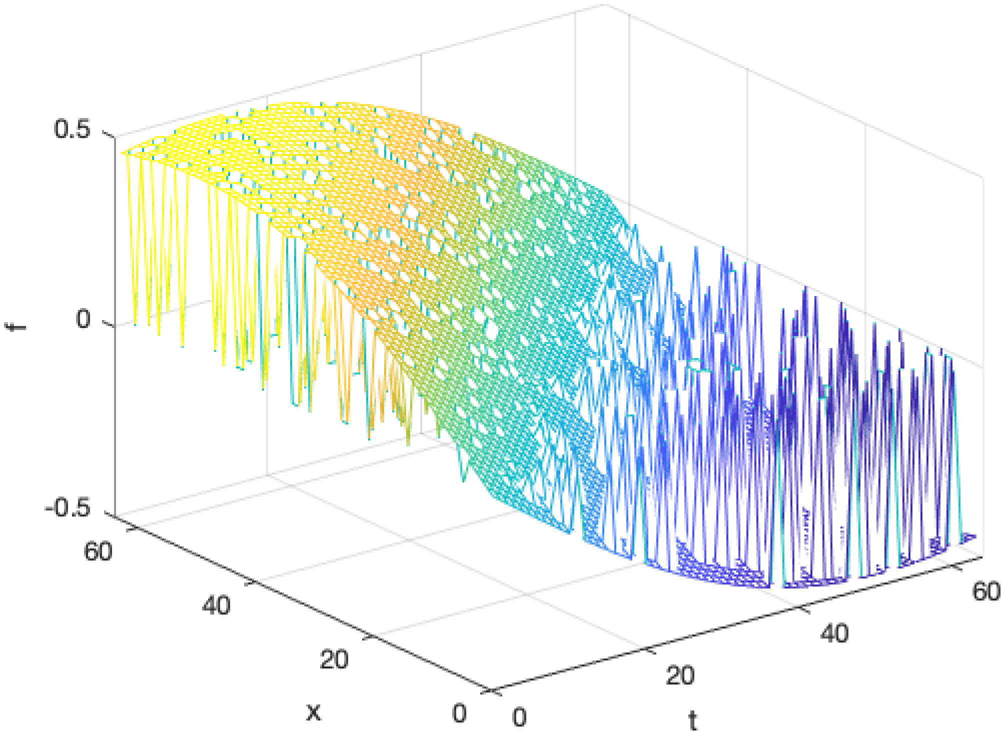
r the recovered vector. But originally gappy with 10% missing

[10]:
```
r = f.*m;
```

[11]:
```
size(r)
```

```
ans =
    64    64
```

[12]:
```
mesh(r)
xlabel('t')
ylabel('x')
zlabel('f')
```



estimate the missing by taking ensemble average

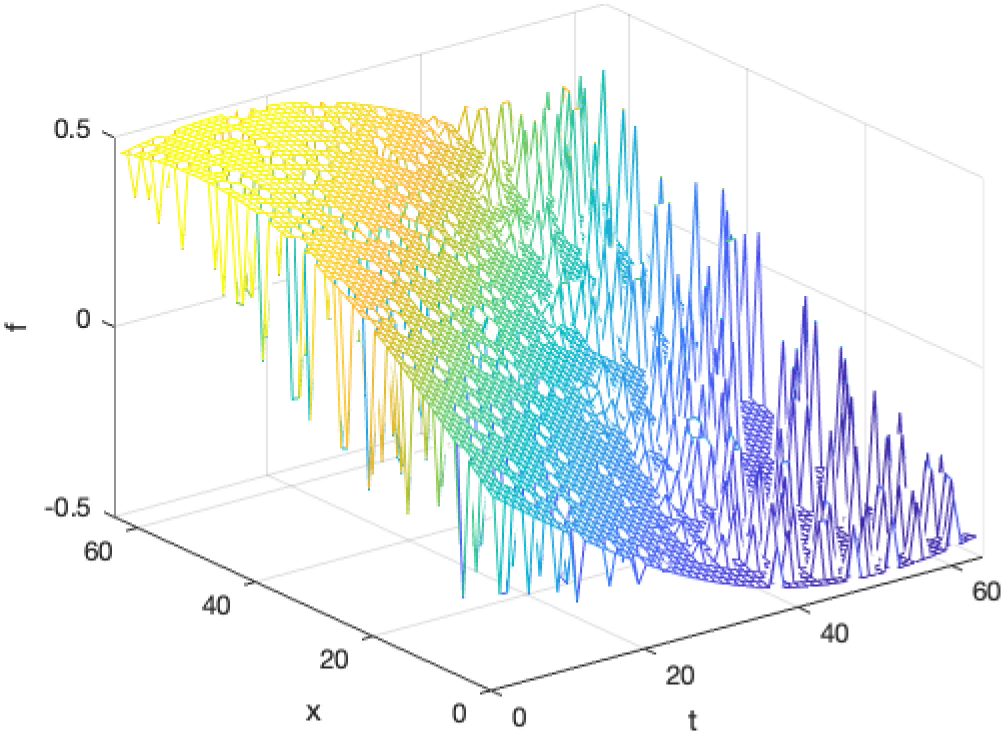
[13]:

```
for i = 1:P
    for j = 1:M
        if (m(j,i)==0)
            r(j,i) = sum(f(j,:).*m(j,:))/sum(m(j,:));
        end
    end
end
```

bring the missing values to non-zero values, but still rough

[14]:

```
mesh(r)
xlabel('t')
ylabel('x')
zlabel('f')
```



[15]:
```
r1=r;
```

[16]:
```
update = 1;
eps = 1e-3;
it = 0;
while (update>eps && it<50)

    r_old = r;
    [v, e] = eig(r*r');
    ev(it+1,:) = diag(e);

    D = 2;
    %KL basis
    v = v(:,end-D:end);
    for i = 1:P
        %Mij = (phi_i,phi_j)m
        A = (v.*m(:,i))'*(v.*m(:,i));
        %fi = (x_tilde,phi_i)m
        b = (v.*m(:,i))'*r(:,i);
        %a=M_inv f
        a = A\b;
        %XD = ai phi_i
        f(:,i) = v*a;
    end
    %fill with xD at missing places
    for i = 1:P
        for j = 1:M
            if (m(j,i)==0)
                r(j,i) = f(j,i);
            end
        end
    end

    if it == 0
        r1_iter=r;
    end
    %check the update for convergence threshold
    update = norm(r_old-r);
    it = it+1;
end
```

Looks like D=3 provides the best approximation

[17]:

```
it
```

```
it =
     8
```

[18]:

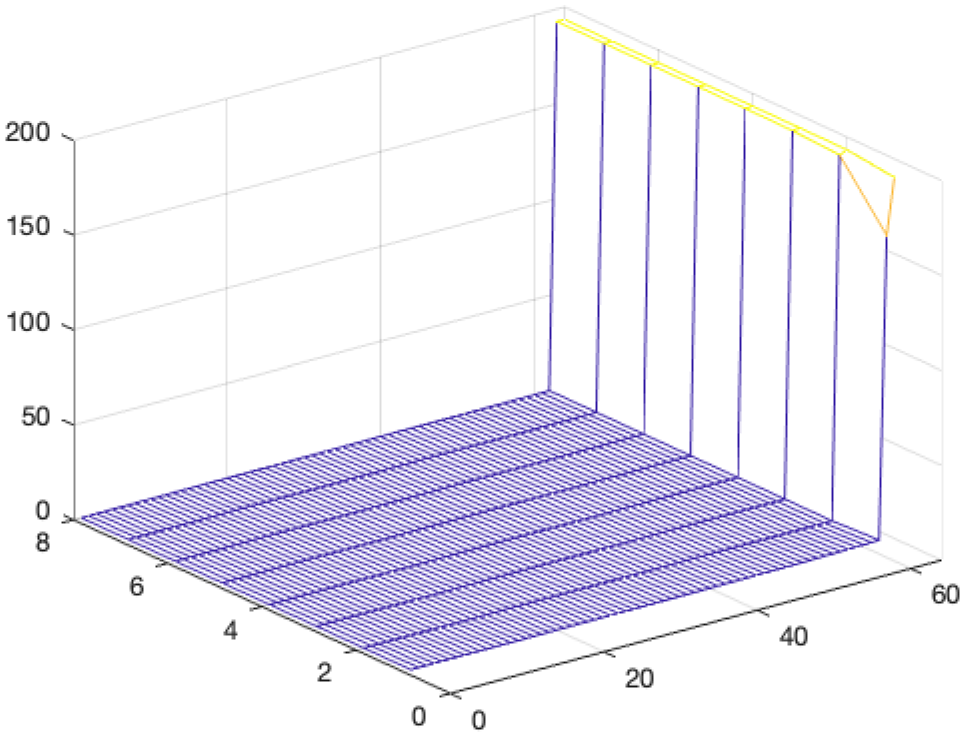
```
size(ev)
```

```
ans =
     8    64
```

Eigenvalues of the 8 iterations. Will get a better plot in matlab

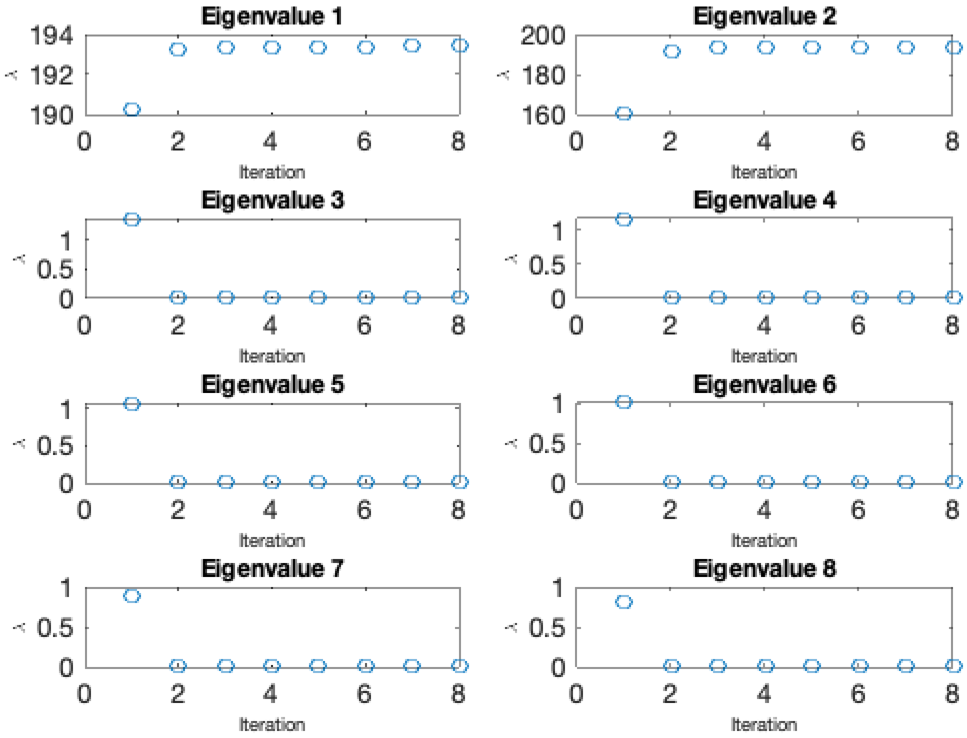
[19]:

```
mesh(ev)
```



[20]:

```
figure() ;
 for i = 1 : it
    subplot(4, 2, i) ;
    plot(ev(:,64-i+1),'o') ;
    title(strcat("Eigenvalue ",num2str(i)),'FontSize',9);
    xlabel('Iteration','FontSize',7) ;
    ylabel('\lambda','FontSize',7) ;
 end
```



[21]:

```
[v, e] = eig(r*r');
```

[22]:

```
size(v)
```

```
ans =
    64    64
```

First 10 eigenfucntions corresponding to 10 largest eigenvalues

[23]:

```
size(v(:,end-9:end))
```

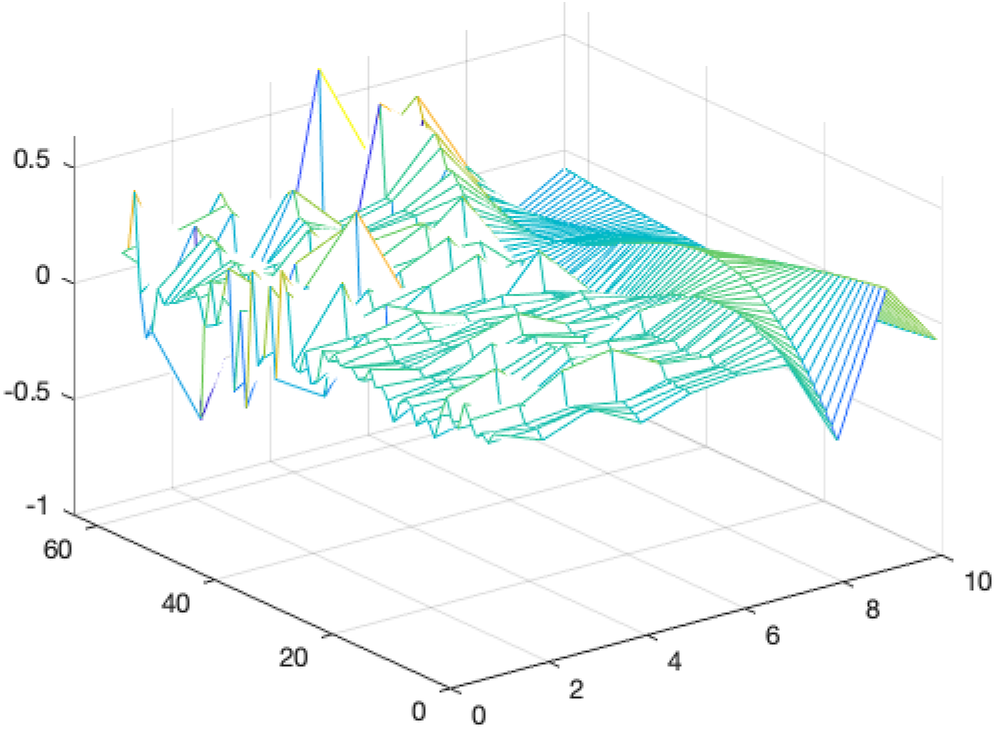
```
ans =
    64    10
```

[24]:

```
%v(:,end-9:end)
```

[25]:

```
mesh(v(:,end-9:end))
```

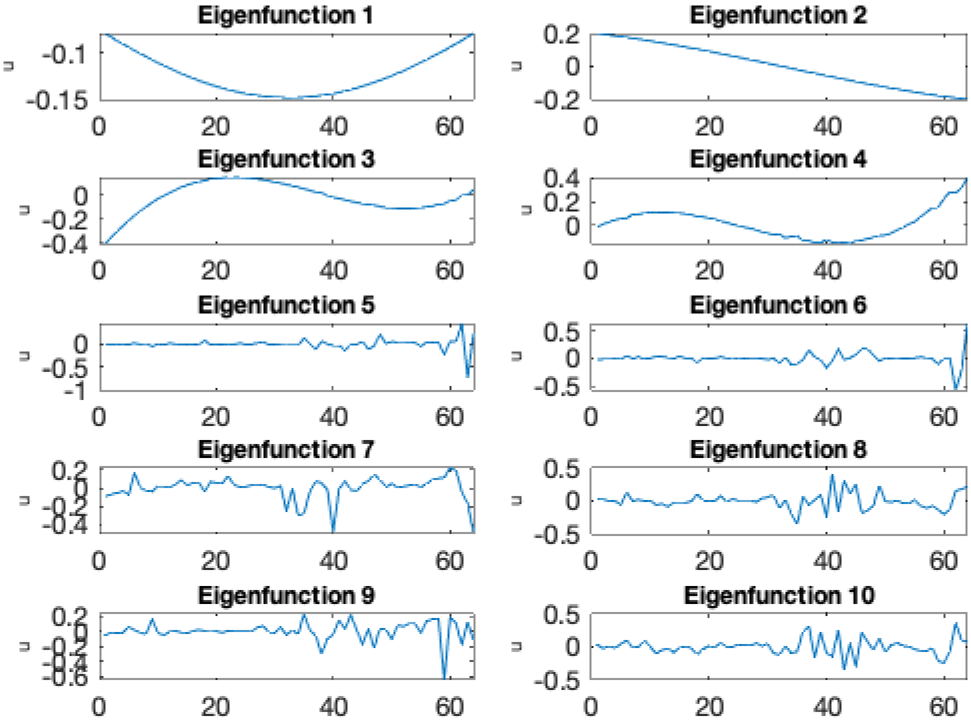


```
[26]: eig_fn=v(:,end-9:end);
```

```
[27]: size(v)
```

```
ans =
    64    64
```

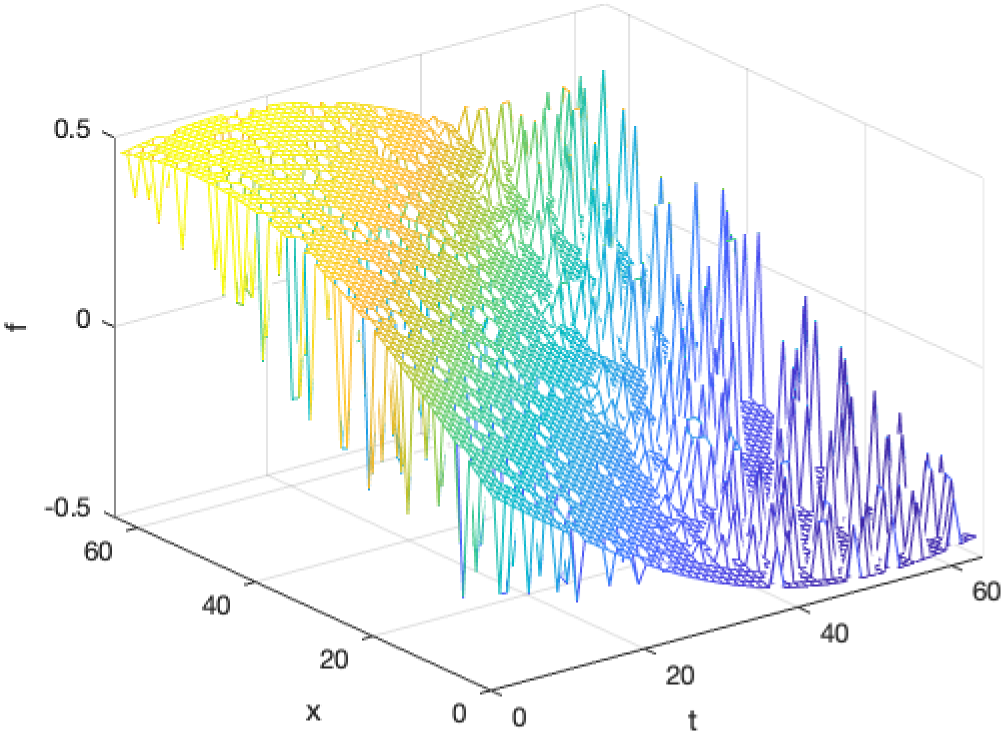
```
[28]: figure() ;
       for i = 1 : 10
          subplot(5, 2, i) ;
          plot(v(:,64-i+1)) ;
          title(strcat("Eigenfunction ",num2str(i)),'FontSize',9);
          xlabel('','FontSize',7) ;
          ylabel('u','FontSize',7) ;
       end
```



gappy data

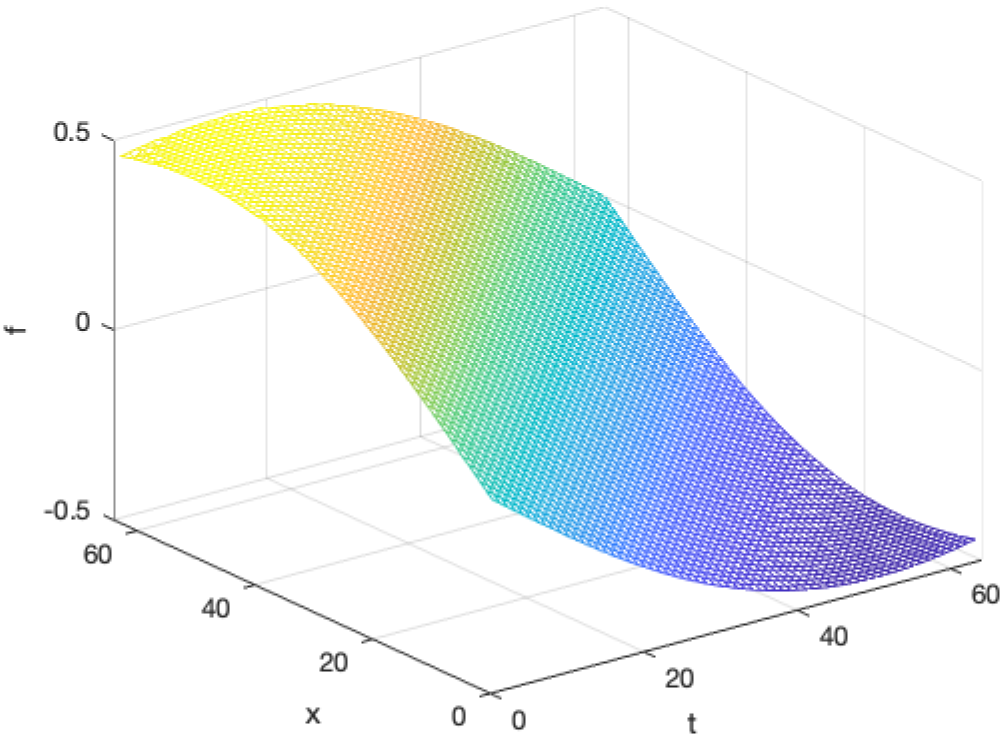
[36]:

```
mesh(r1);
xlabel('t')
ylabel('x')
zlabel('f')
```



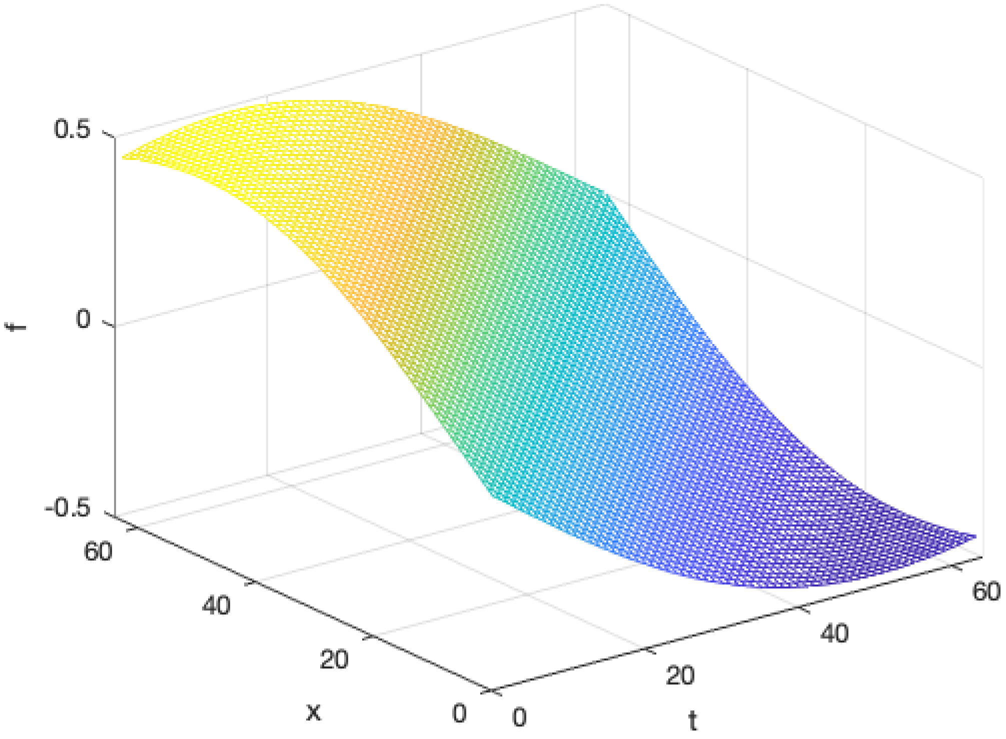
recovered data

[37]:
```
mesh(r);
xlabel('t')
ylabel('x')
zlabel('f')
```



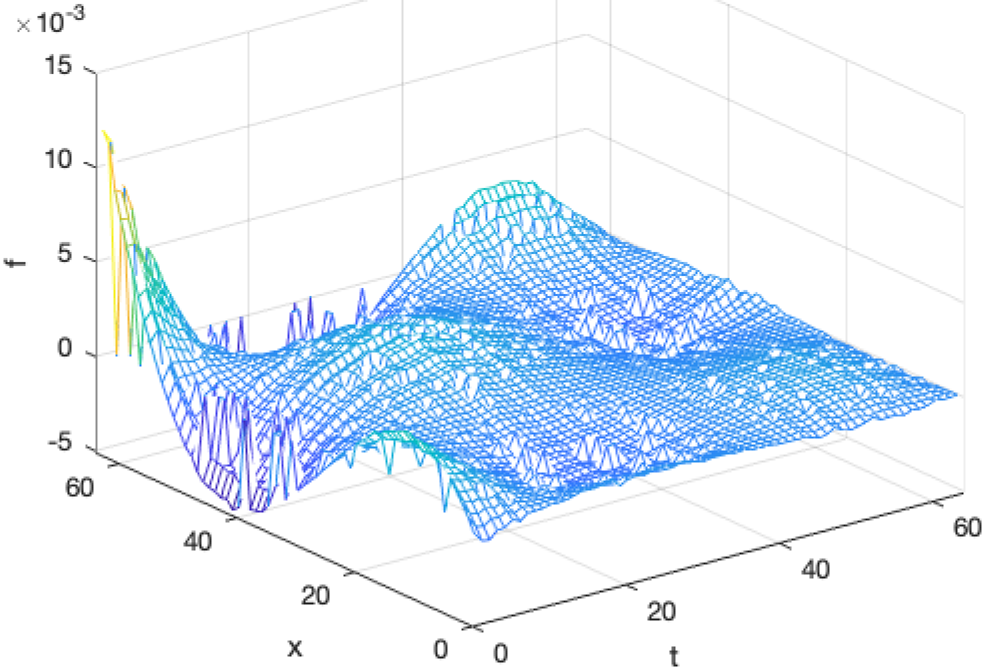
original data

[38]:
```
mesh(f)
xlabel('t')
ylabel('x')
zlabel('f')
```



error between original and recovered data

[39]:
```
mesh(r-f);
xlabel('t')
ylabel('x')
zlabel('f')
```


×10^-3
15
10
5
0
-5
f
60
40
20
0
x
0
20
40
60
t

[ ]:

[ ]: