Department of Applied Mathematics and Statistics
COLORADO SCHOOL OF MINES
MATH484: Capstone - Mathematical and Computational Modeling

Assignment #6, Spring 2014
SOLUTIONS

1. Let $\eta = \frac{1}{4}, \alpha = \frac{2}{10}$, and $\beta = 2$. Use Fourier Series with $N = 15$ to solve the inverse Electron Beam Lithography problem (i.e., find and plot $D_{15}(x)$) on the spatial interval $[-1, 1]$ with $dx = 10^{-3}$ and

$$E(x) = \begin{cases} 1 & \text{if } |x| < \frac{1}{2} \\ 0 & \text{else.} \end{cases}$$

Also compute and plot $E_{15}(x)$ - the approximation to $E(x)$ given by the first 15 terms of the Fourier Series expansion - in the same figure as $D_{15}(x)$.
Hint: To check your code for $D_{15}(x)$, try it with $N = 10$ and compare to Figure 3.9.

Figure 1 displays a comparison between the original desired shape, the dosing Fourier approximation, and the resulting desired shape Fourier approximation. Below is the associated code.

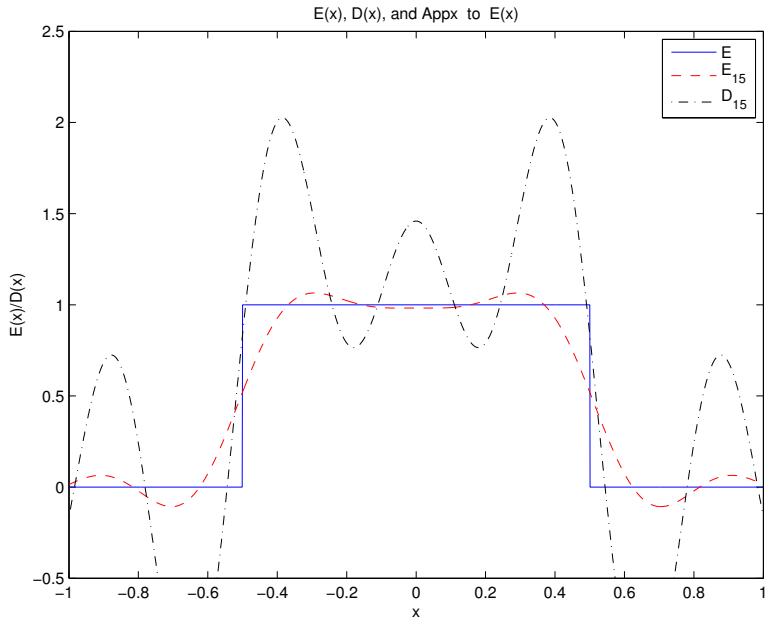

Figure 1: Graphs for Problem 1

MATLAB Code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% Function used for the initial desired shape - E(x)
function out = Shape(x)
out = zeros(1, length(x));
for j = 1:length(x)
    if ((abs(x(j)) <= 0.5))
        out(j) = 1;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%% Other code
clear; clc;

dx = 1e-3;
a = -1;
b = 1;
x = a:dx:b;
N = 15;

%Constants
alpha = 0.2;
beta = 2;
eta = 0.25;
T1 = alpha^2/4;
T2 = beta^2/4;

E_target = @(s) Shape(s);
D = zeros(1,N);
E = D;
Ex = (1/pi)*integral(E_target, 0, pi)*ones(1,length(x));
Dx = Ex;

for n = 1:N
    f = @(s) E_target(s).*cos(n*s);
    E(n) = (2/pi)*integral(f, 0, pi);
    % Similarly, you could just use
    % E(n) = (2/pi)*integral(@(s) cos(n*s), 0, 0.5);
    D(n) = ((1+eta)/(exp(-n^2*T1) + eta*exp(-n^2*T2)))*E(n);
    Dx = Dx + D(n)*cos(n*x);
    Ex = Ex + E(n)*cos(n*x);
end
figure;
plot(x, E_target(x), x, Ex,'--r', x, Dx, '-.k'),
title('E(x), D(x), and Appx to E(x)'),
```

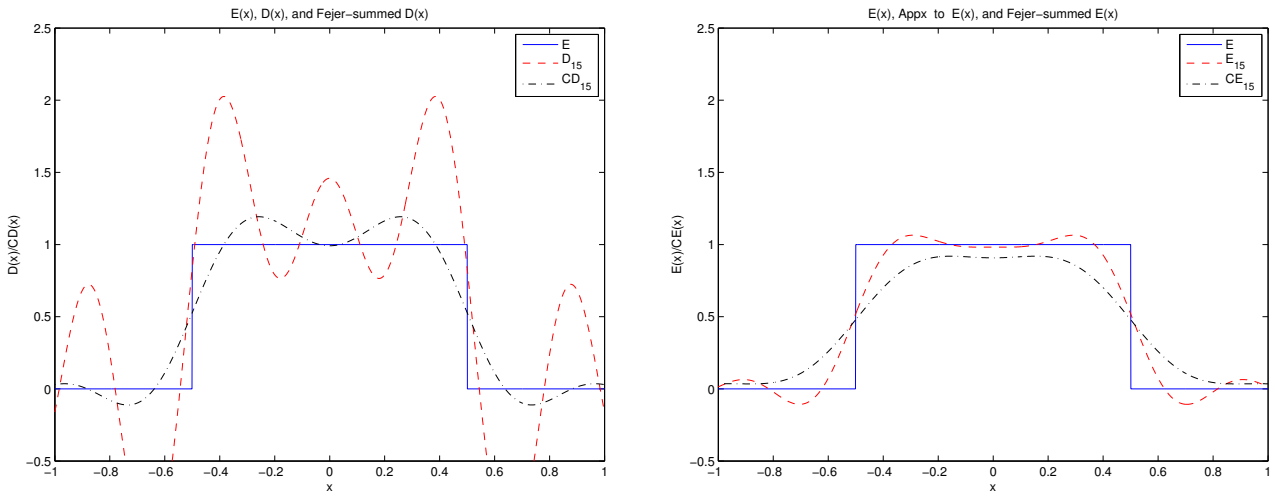

Figure 2: Graphs for Fejer Sums in Problem 2

```
axis([-1,1,-0.5,2.5]),
xlabel('x'), ylabel('E(x)/D(x)'),
legend('E', ['E_{',num2str(N),'}'], ['D_{',num2str(N),'}'])
```

2. Do the same as for Problem 1, but use the Fejer sums (i.e. $\tilde{D}_{15}(x)$ and $\tilde{E}_{15}(x)$) instead. Compare the new dosing function approximation to the old one - how are they different and why is one better?

Figure 2 displays the Fejer Fourier approximations compared to those from the previous problem - $\tilde{D}_{15}(x)$ is represented by $CD_{15}(x)$ and $\tilde{E}_{15}(x)$ is represented by $CE_{15}(x)$ on the plots. Notice that the Fejer approximations contain less oscillations than the standard Fourier series approximations and remain positive for more x values. Hence, they appear to serve our purpose much better. In fact, $\tilde{E}_{15}(x) > 0$ for all $x \in [-1, 1]$. Below is the associated code to generate the graphs.

MATLAB Code

```
clear; clc;

dx = 1e-3
a = -1;
b = 1;
x = a:dx:b;
N = 15;

%Constants
alpha = 0.2;
beta = 2;
```

```
eta = 0.25;
T1 = alpha^2/4;
T2 = beta^2/4;

E_target = @(s) Shape(s);

D = zeros(1,N);
E = D;
Dx = 1/(2*pi);
Ex = Dx;
CDx = Dx;
CEx = Ex;

for n = 1:N
    E(n) = (2/pi)*integral(@(s) cos(n*s), 0, 0.5);
    D(n) = (1+eta)*E(n)/(exp(-n^2*T1) + eta*exp(-n^2*T2));
    Dx = Dx + D(n)*cos(n*x);
    Ex = Ex + E(n)*cos(n*x);

    %Add Cesaro Sum
    CDx = CDx + (1 - (n/N))*D(n)*cos(n*x);
    CEx = CEx + (1 - (n/N))*E(n)*cos(n*x);

end

figure;
plot(x, E_target(x), x, Dx,'--r', x, CDx,'-.k'),
title('E(x), D(x), and Fejer-summed D(x)'),
axis([a,b,-0.5,2.5]),
xlabel('x'), ylabel('D(x)/CD(x)'),
legend('E', ['D_{',num2str(N),'}'], ['CD_{',num2str(N),'}'])

figure;
plot(x, E_target(x), x, Ex,'--r', x, CEx, '-.k'),
title('E(x), Appx  to  E(x), and Fejer-summed E(x)'),
axis([a,b,-0.5,2.5]),
xlabel('x'), ylabel('E(x)/CE(x)'),
legend('E', ['E_{',num2str(N),'}'], ['CE_{',num2str(N),'}'])
```

3. Friedman & Littman, p.102, Problem **5.7.3** - Implicit Method only
Define $L = 10$, $t_0 = 4$, $u_0 = 2$, as well as the functions

$$S_1(t) = 1 \qquad \text{for } t \in [0, 4]$$

and

$$S_2(t) = \begin{cases} \frac{1}{2} & \text{if } t \in [0,2] \\ \frac{3}{2} & \text{if } t \in [2,4] \end{cases}$$

where L is the length of the converter and impose the boundary condition $\frac{\partial T}{\partial x} = 0$ at the right endpoint. Using these values and the implicit method on p.98 with $dx = 0.05$ and $dt = 0.01$, solve (5.19)–(5.23) and determine which control function makes $J(S)$ smaller, S_1 or S_2. Recall that $J(S)$ is defined for the simplified problem by (5.25).

The first control, $S_1(t)$, provides a smaller value of the objective functional $J(S)$. Below is the associated code and results.

MATLAB Code

```
clear; clc;

dx = 5e-2;
dt = 1e-2;
sigma = dt/(dx^2);
u0 = 2;

a = 0;
b = 10;
x = a:dx:b;
m = length(x);

t0 = 4;
t = 0:dt:t0;
n = t0/dt;

S(1,1:n+1) = 1;
S(2,1:n+1) = 0.5 + (t(1:n+1) >= 2);

M = (1+2*sigma)*diag(ones(m-2,1))-sigma*diag(ones(m-3,1),1)...
        -sigma*diag(ones(m-3,1),-1);

%Neumann BC at x = L
M(end, end) = M(end, end) - sigma;

[L,U] = lu(M);

for p = 1:2
    T(1,:) = [S(p,1);zeros(m-1, 1)];
    u(1,:) = u0*ones(m, 1);
```

```
    for i = 1:n
        %Control and Dirichlet BCs at x = 0
        T(i+1, 1) = S(p,i+1);
        u(i+1, 1) = u0;

        NL = (u(i,1:m).*T(i,1:m))./(1 + T(i,1:m));

        %Implicit Solution for T
        b = T(i,2:m-1) + dt*NL(2:m-1);
        b(1) = b(1) + sigma*T(i+1, 1);
        y = L\b';
        T(i+1,2:m-1) = U\y;

        %Neumann BC at x = L
        T(i+1, m) = T(i+1, m-1);

        %Explcit Solution for u
        for k = 1:m-1
            u(i+1, k+1) = u(i+1,k) - dx*u(i+1,k)*T(i+1,k)/(1 + T(i+1,k));
        end
    end
    J(p) = trapz(t,u(1:n+1, m));
    clear u;
    clear T;
end
format long g
J


J =

          2.17689494366807           2.7273835653439
```