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

Assignment #6, Spring 2015
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

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

In the same figure as $D_{15}(x)$, plot $E(x)$, and compute and plot $E_{15}(x)$ - the approximation to $E(x)$ given by the first 15 terms of the Fourier Series expansion. Use a solid blue line (default) for $E(x)$, a dashed red line for $E_{15}(x)$, and a dot-dashed black line for $D_{15}(x)$.
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.

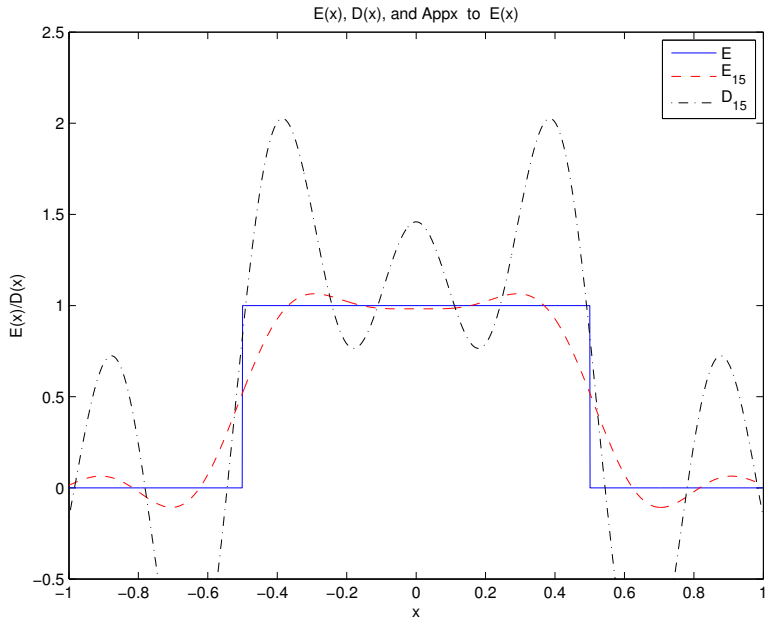

Figure 1: Graphs for Problem 1

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)'),
```

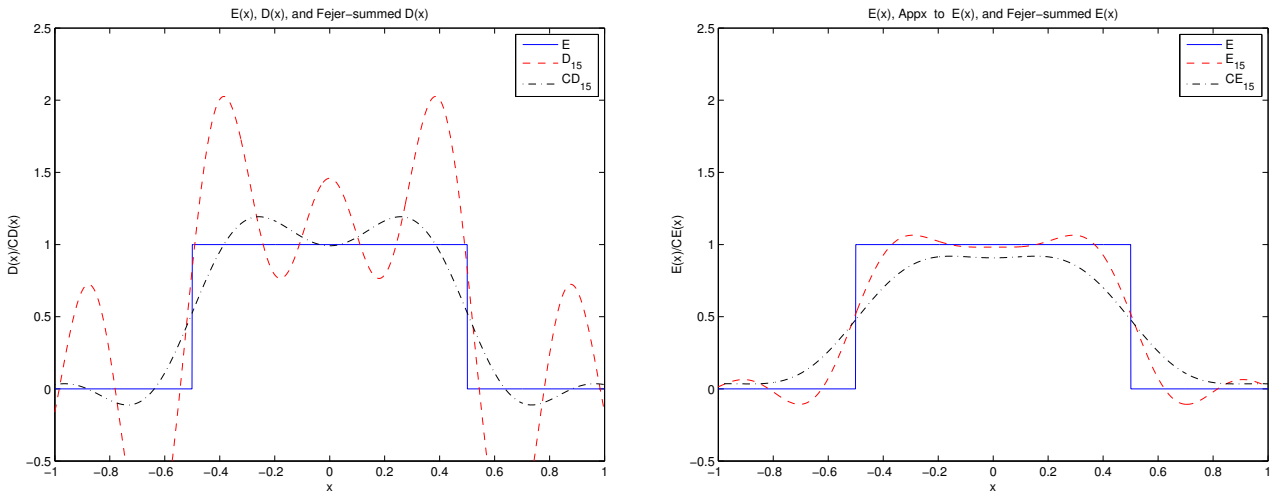

Figure 2: Graphs for Fejer Sums in Problem 2

```
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.82, Problem **4.6.1**
Use the implicit finite difference method included on p.82 above the problem statement with step sizes $dt = 1$ sec and $dx = 500\mu$m, and the stopping time $T = 180$ sec. After computing the solutions create two figures - one with $A(0, x)$, $A(90, x)$, and $A(180, x)$ and another with $B(0, x)$, $B(90, x)$, and $B(180, x)$. Within each of these figures, use a solid blue line (default)

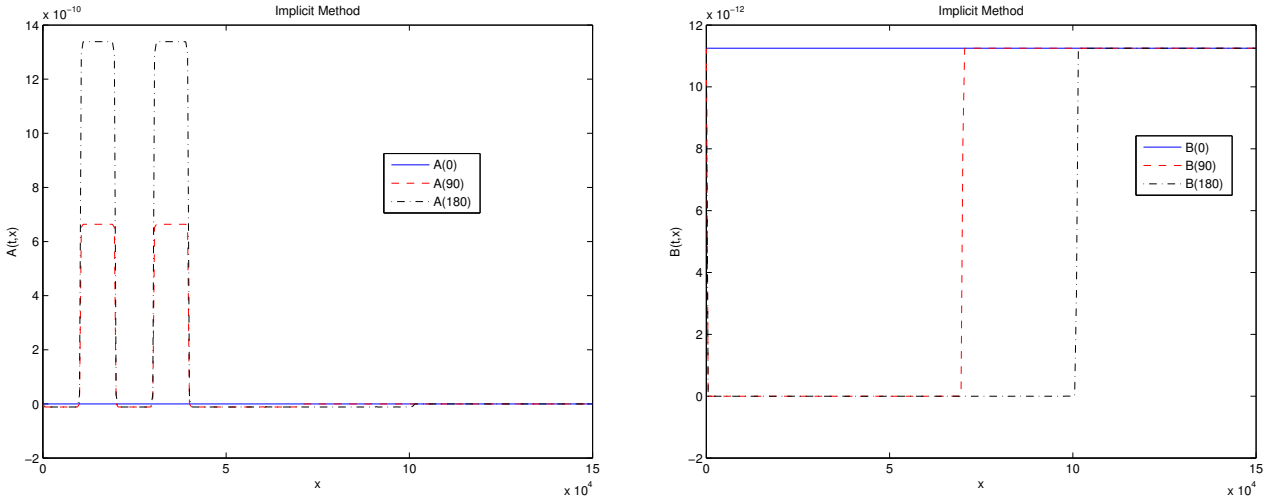

Figure 3: Graphs for $A(t,x)$ and $B(t,x)$ in Problem 3

for the $t = 0$ solution, a dashed red line for the $t = 90$ solution, and a dot-dashed black line for the $t = 180$ solution.

Figure 3 displays the representations of $A(t,x)$ and $B(t,x)$ at the three different times. Below is the associated code and results.

MATLAB Code

```
clear; clc;

dx = 5e2;
dt = 1;

a = 0;
b = 1.5e5;

T = 180;

D = 100;
k = 6.6e12;
gamma = 7.5e-12;
B0 = 1.125e-11;

sigma = D*dt/(dx^2);

x = a:dx:b;
m = length(x);
n = T/dt;
```

```
J = find( (x > 1e4 & x < 2e4) | (x > 3e4 & x < 4e4) );
E = zeros(1,m);
E(J) = 1;

A = zeros(n,m);
B(1,:) = B0*ones(1,m);

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

for i = 1:n
    for j = 1:m
        B(i+1, j) = B(i,j)/(1 + k*dt*A(i,j));
    end

    %Assuming Dirichlet BCs so that A(i,1) = A(i, m) = 0
    b = (1 - k*dt*B(i+1,2:m-1)).*A(i,2:m-1) + gamma*dt*E(2:m-1);
    A(i+1,2:m-1) = M\b';
end

figure;
plot(x, A(1, :), x, A(n/2+1, :),'--r', x, A(n+1, :),'-.k'), title('Implicit Method'),
    xlabel('x'), ylabel('A(t,x)'), legend('A(0)', 'A(90)', 'A(180)');

figure;
plot(x, B(1,:), x, B(n/2+1, :),'--r', x, B(n+1, :),'-.k'), title('Implicit Method'),
    xlabel('x'), ylabel('B(t,x)'), legend('B(0)', 'B(90)', 'B(180)');
```