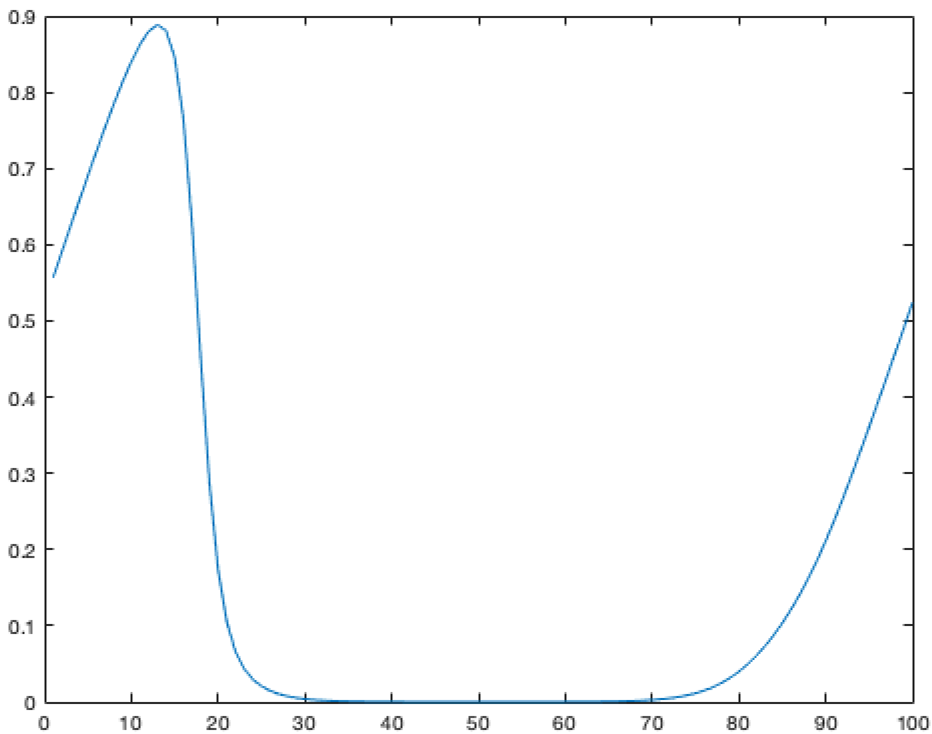
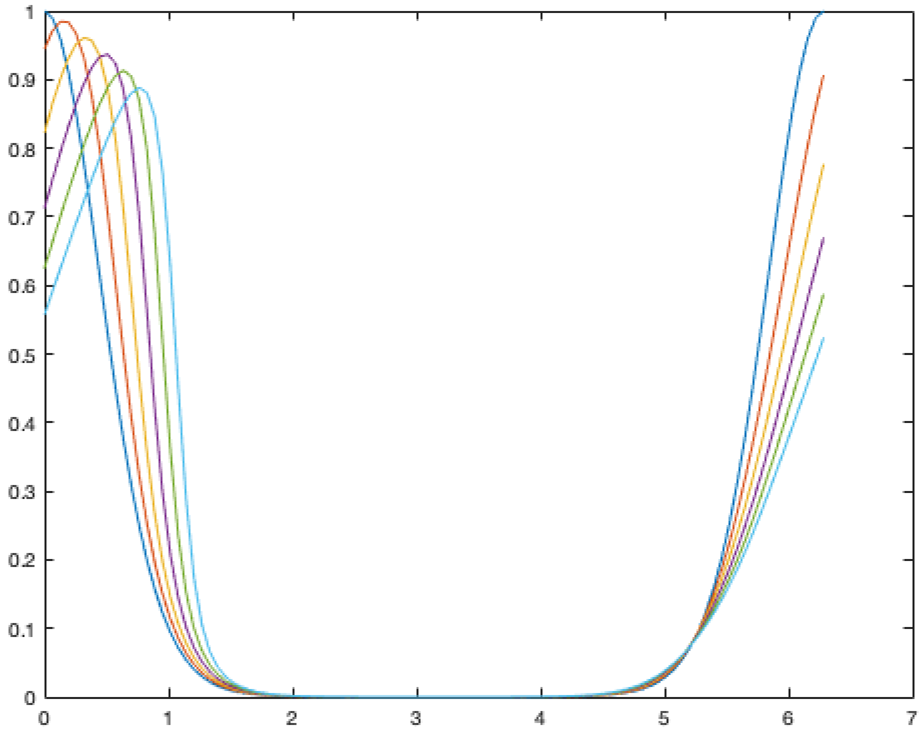
```
eps = 0.03;
N=100;
t1=0.01;

tp = linspace(0, t1, N);
D1 = zeros(N,N);
for k=1:N
    for j=1:N
       if k==j
           D1(k,j) = 0;
       else
           D1(k,j) = (((-1).^(k+j))./2).*cot((k-j).*pi./N);
       end
    end
end
D2 = zeros(N,N);
for k=1:N
    for j=1:N
       if k==j
           D2(k,j) = -N.^2./12-1./6;
       else
           D2(k,j) = -(-1).^(k+j)./2.*sin((k-j).*pi./N).^(-2);
       end
    end
end

% RKM to get initial condition at t=1. error for each step is
 O(delT^3),
% so by using delT=0.0001 and 100 steps we have error O(1e-10), which
 is
% much less than the error for Crank-Nicolson: O(delT^2) using
 delT=0.01
% and 100 steps yields truncation error O(0.01). Therefore we can use
 this
% Runge-Kutta approximate without disrupting the C-N estimate much.
delT = 0.0001;
N2 = t1./delT;
tp = linspace(0, 2.*pi, N);
ui = @(x) exp(-10.*sin(x./2).^2);
ui = ui(tp);
ti = 0;
f = @(u,t) eps.*(D2*u')-u'.*(D1*u');
for a=1:1
    k1 = delT.*f(ui,ti);
    k2 = delT.*f(ui+k1'./2,ti+delT./2);
    ui = ui+k2';
    ti = ti+delT;
end
% end RKM. initial condition at t=1 is stored in ui

delT=1./N;
ut = @(x) exp(-10.*sin(x./2).^2);
```

```
um1=ut(tp);
un=ui;
plots = [un];
for a=2:N
    rhs = um1+(eps.*delT.*D2*um1')'-(2.*delT.*un'.*(D1*un'))';
    up1 = rhs*inv(eye(N)-eps.*D2.*delT);
    um1=un;
    un=up1;
    if mod(a,20)==0
       plots = [plots; un];
    end
end
figure
plot(un)
figure
plot(tp, plots)
```





Published with MATLAB® R2018a