```
function [x, t, psi, psire, psiim, psimod, prob, v] = sch_1d_cn(tmax,
 level, lambda, idtype, idpar, vtype, vpar)
% Inputs
%
% tmax: Maximum integration time
% level: Discretization level
% lambda: dt/dx
% idtype: Selects initial data type
% idpar: Vector of initial data parameters
% vtype: Selects potential type
% vpar: Vector of potential parameters
%
% Outputs
%
% x: Vector of x coordinates [nx]
% t: Vector of t coordinates [nt]
% psi: Array of computed psi values [nt x nx]
% psire Array of computed psi_re values [nt x nx]
% psiim Array of computed psi_im values [nt x nx]
% psimod Array of computed sqrt(psi psi*) values [nt x nx]
% prob Array of computed running integral values [nt x nx]
% v Array of potential values [nx]

% Define mesh and derived parameters ...
   nx = 2^level + 1;
   x = linspace(0.0, 1.0, nx);
   dx = x(2) - x(1);

   dt = lambda * dx;
   nt = round(tmax / dt) + 1;
   t = [0 : nt-1] * dt;

   % Initialize solution, and set initial data ...
   psi = zeros(nt, nx);
   if idtype == 0
      m = idpar(1);
      psi(1, :) = sin(m*pi*x);
   elseif idtype == 1
      x0 = idpar(1);
      delta = idpar(2);
      p = idpar(3);
      psi(1, :) = exp(1i*p.*x).*exp(-((x - x0)./delta).^2);
      psi(1,1) = 0.0;
      psi(1, nx) = 0.0;
   else
      fprintf('sch_1d_cn: Invalid idtype=%d\n', idtype);
      return
   end

   if vtype == 0
       v = zeros(nx, 1);
   elseif vtype == 1
```

```
    x_mn = vpar(1);  %get x_min
    index_low = round(x_mn/dx); % produce index of x_min

    x_mx = vpar(2); % get x_max
    index_high = round(x_mx/dx); %produce index of x_max

    potential = vpar(3);
    v = zeros(nx, 1);
    for xx = 1: nx
        if (xx < index_low  || xx > index_high)
             v(xx, 1) = 0;
        else
             v(xx, 1) = potential; % assign potential
        end
    end
else
   fprintf('sch_1d_cn: Invalid idtype=%d\n', idtype);
   return
end
% boundary conditions

% Initialize storage for sparse matrix and RHS ...
cplus = zeros(nx,1);
c0  = zeros(nx,1);
cminus = zeros(nx,1);
f  = zeros(nx,1);

% Set up tridiagonal system ...

Potential = - 0.5*v;
cplus = 0.5 / dx^2 * ones(nx, 1);
c0 = (1.0*1i/dt -1.0 / dx^2) * ones(nx,1) + Potential;

cminus = cplus;
% Fix up boundary cases ...
c0(1) = 1.0;
cplus(2) = 0.0;
cminus(nx-1) = 0.0;

cplus(nx) = 1.0;

% Define sparse matrix ...
A = spdiags([cminus c0 cplus], -1:1, nx, nx);
A(1, 1) = 1.0;
A(1, 2) = 0.0;
A(nx, nx-1) = 0.0;
A(nx,nx) = 1.0;

%iterate over the rest of timesteps
for n = 1 : nt-1
   % Define RHS of linear system ...

   V = 0.5*transpose(v).*psi(n, 1:nx);
```

```
      f(2:nx-1) = (1.0*1i/dt +1/dx^2)*psi(n, 2:nx-1) - 0.5 * ( psi(n,
 1:nx-2)  + psi(n, 3:nx)) / dx^2 + V(2:nx-1);

      f(1) = 0.0;
      f(nx) = 0.0;

      % Solve system, thus updating approximation to next time
      % step ...
      psi(n+1, :) = A \f;
      psi(n+1, 1) = 0.0;
      psi(n+1, nx) = 0.0;
   end
   psire = real(psi);
   psiim = imag(psi);
   psimod = abs(psi).^2;

   %Integrate probability
   prob = zeros(nt, nx);
   prob(:, 1) = psimod(:, 1);
   for m = 2 : nx
       deltax = x(m)-x(m-1);
       prob(:, m) = 0.5*(psimod(:, m) + psimod(:, m-1)).*(dx) +prob(:,
 m-1);
       if m == nx

           prob = prob ./prob(:, m);

       end
   end

end
```

```
Not enough input arguments.

Error in sch_1d_cn (line 24)
   nx = 2^level + 1;
```

Published with MATLAB® R2018a