W.D. Henshaw **Math 6840: Solutions for Problem Set 2**

1 *Solution:*

1(a) Here is the BE-CD2 scheme

$$
\begin{aligned}
&\frac{U_i^{n+1}-U_i^n}{\Delta t} = \kappa D_+D_- U_i^{n+1}, \qquad i=1,2,3,\ldots,N_x-1, \quad n=0,1,2,\ldots \\
&U_0^{n+1} = g_a(t^{n+1}), \qquad U_{N_x}^{n+1} = g_b(t^{n+1}), \quad n=0,1,2,\ldots \\
&U_i^0 = u_0(x_i), \qquad i=0,1,2,3,\ldots,N_x.
\end{aligned}
$$

1(b) The equation satisfied by the error is

$$
\begin{aligned}
&\frac{e_i^{n+1}-e_i^n}{\Delta t} - \kappa D_+D_- e_i^{n+1} = \tau_i^n, \qquad i=1,2,3,\ldots,N_x-1, \quad n=0,1,2,\ldots \\
&e_0^{n+1} = 0, \qquad e_{N_x}^{n+1} = 0, \qquad n=0,1,2,\ldots \\
&e_i^0 = 0, \qquad i=0,1,2,3,\ldots,N_x.
\end{aligned}
$$

where

$$
\tau_i^n = D_{+t}u(x_i,t^n) - \kappa D_+D_- u(x_i,t^{n+1}).
$$

By Taylor series

$$
\begin{aligned}
D_{+t}u(x_i,t^n) &= \partial_t u(x_i,t^n) + \frac{\Delta t}{2}\partial_t^2 u(x_i,t^n) + \mathcal{O}(\Delta t^2) \\
D_+D_- u(x_i,t^{n+1}) &= D_+D_-\Big(u(x_i,t^n) - \Delta t\partial_t u(x_i,t^n)\Big) + \mathcal{O}(\Delta t^2), \\
&= \partial_x^2 u(x_i,t^n) + \frac{\Delta x^2}{12}\partial_x^4 u(x_i,t^n) - \Delta t\partial_t\partial_x^2 u(x_i,t^n) + \mathcal{O}(\Delta x^2+\Delta t^2)
\end{aligned}
$$

whence (using $\partial_t = \kappa\partial_x^2 u$), *check me*

$$
\begin{aligned}
\tau_i^n &= \partial_t u(x_i,t^n) - \kappa\partial_x^2 u(x_i,t^n) + \frac{\Delta t}{2}\partial_t^2 u(x_i,t^n) - \kappa\Delta t\partial_t\partial_x^2 u(x_i,t^n) - \kappa\frac{\Delta x^2}{12}\partial_x^4 u(x_i,t^n) + \mathcal{O}(\Delta t^2+\Delta x^2), \\
&= \frac{\Delta t}{2}\kappa^2\partial_x^4 u(x_i,t^n) - \kappa^2\Delta t\partial_x^4 u(x_i,t^n) - \kappa\frac{\Delta x^2}{12}\partial_x^4 u(x_i,t^n) + \mathcal{O}(\Delta t^2+\Delta x^2), \\
&= -\Big(\kappa^2\frac{\Delta t}{2} + \kappa\frac{\Delta x^2}{12}\Big)\partial_x^4 u(x_i,t^n) + \mathcal{O}(\Delta t^2+\Delta x^2), \\
&= -\kappa\Big(\kappa\frac{\Delta t}{2} + \frac{\Delta x^2}{12}\Big)\partial_x^4 u(x_i,t^n) + \mathcal{O}(\Delta t^2+\Delta x^2).
\end{aligned}
$$

1(c) The error equation can be written as

$$
e_i^{n+1} = e_i^n + r\Big[e_{i+1}^{n+1} - 2e_i^n + e_{i-1}^n\Big] + \Delta t\tau_i^n
$$

where $r = \kappa\Delta t/\Delta x^2$. Whence

$$
(1+2r)e_i^{n+1} = e_i^n + r\Big[e_{i+1}^{n+1} + e_{i-1}^{n+1}\Big] + \Delta t\tau_i^n
$$

Taking absolute values, assuming $r > 0$, and using the triangle inequality gives

$$(1 + 2r)|e_i^{n+1}| \le |e_i^n| + r\Big[|e_{i+1}^{n+1}| + |e_{i-1}^{n+1}|\Big] + \Delta t|\tau_i^n|$$

Taking the maximum over i

$$(1 + 2r)E^{n+1} \le E^n + r\Big[E^{n+1} + E^{n+1}\Big] + \Delta t Y^n$$

where

$$E^n = \max_{0\le i\le N_x} |e_i^n|, \qquad Y^n = \max_{0\le i\le N_x} |\tau_i^n|.$$

Thus

$$E^{n+1} \le E^n + \Delta t Y^n$$

which gives

$$\begin{aligned} E^{n+1} &\le E^0 + \Delta t\Big(Y^0 + Y^1 + \ldots + Y^n\Big), \\ &\le (n+1)\Delta t \max_{0\le m\le n} Y^m \end{aligned}$$

Let $T = (n+1)\Delta t$ be a fixed final time. Assuming the derivatives of u are bounded, the truncation error will go to zero as $\Delta t \to 0$ and $\Delta x \to 0$ and thus

$$E^{n+1} \to 0.$$

This proves the BE-CD2 scheme is convergent as $\Delta t \to 0$ and $\Delta x \to 0$ provided $r > 0$.

2. (20 points) Mixed boundary conditions and ghost points (centered).

2(a) Here is the AB2-CD2 scheme

$$\begin{aligned} \frac{U_i^{n+1} - U_i^n}{\Delta t} &= \frac{3}{2}\big(\kappa D_+D_-U_i^n + f(x_i, t^n)\big) \\ &\quad - \frac{1}{2}\big(\kappa D_+D_-U_i^{n-1} + f(x_i, t^{n-1})\big), \qquad i = 0, 1, 2, \ldots, N_x, \quad n = 0, 1, 2, \ldots \\ \alpha_a U_0^{n+1} + \beta_a D_0 U_0^{n+1} &= g_a(t^{n+1}), \qquad \alpha_b U_{N_x}^{n+1} + \beta_b D_0 U_{N_x}^{n+1} = g_b(t^{n+1}), \quad n = 0, 1, 2, \ldots \\ U_i^0 = u_0(x_i), &\quad U_i^{-1} = u^e(x_i, -\Delta t), \qquad i = 0, 1, 2, 3, \ldots, N_x, \end{aligned}$$

2(b) The code listing is given below.

2(c) Results for the polynomial manufactured solution ...

Listing 1: Results from 2(d) AB2-CD2, second-order centered, polynomial MS.

```
>> heatMixed
Heat: AB2, centered2, poly: t=1.00: Nx= 20 Nt= 160 dt=6.250e-03 maxErr=8.88e-16
Heat: AB2, centered2, poly: t=1.00: Nx= 40 Nt= 640 dt=1.563e-03 maxErr=1.33e-15 ratio=6.67e-01
Heat: AB2, centered2, poly: t=1.00: Nx= 80 Nt= 2560 dt=3.906e-04 maxErr=1.78e-15 ratio=7.50e-01
Heat: AB2, centered2, poly: t=1.00: Nx=160 Nt=10240 dt=9.766e-05 maxErr=5.33e-15 ratio=3.33e-01
Heat: AB2, centered2, poly: t=1.00: Nx=320 Nt=40960 dt=2.441e-05 maxErr=8.88e-15 ratio=6.00e-01
```

2(d) Results for the trigonometric manufactured solution ...

Listing 2: Results from 2(d) AB2-CD2, second-order centered, trig MS.

```
>> heatMixed
Heat: AB2, centered2, trig: t=1.00: Nx= 20 Nt= 160 dt=6.250e-03 maxErr=1.07e-01
Heat: AB2, centered2, trig: t=1.00: Nx= 40 Nt= 640 dt=1.563e-03 maxErr=2.55e-02 ratio=4.18e+00
Heat: AB2, centered2, trig: t=1.00: Nx= 80 Nt= 2560 dt=3.906e-04 maxErr=6.35e-03 ratio=4.02e+00
Heat: AB2, centered2, trig: t=1.00: Nx=160 Nt=10240 dt=9.766e-05 maxErr=1.59e-03 ratio=3.99e+00
Heat: AB2, centered2, trig: t=1.00: Nx=320 Nt=40960 dt=2.441e-05 maxErr=3.97e-04 ratio=4.00e+00
```

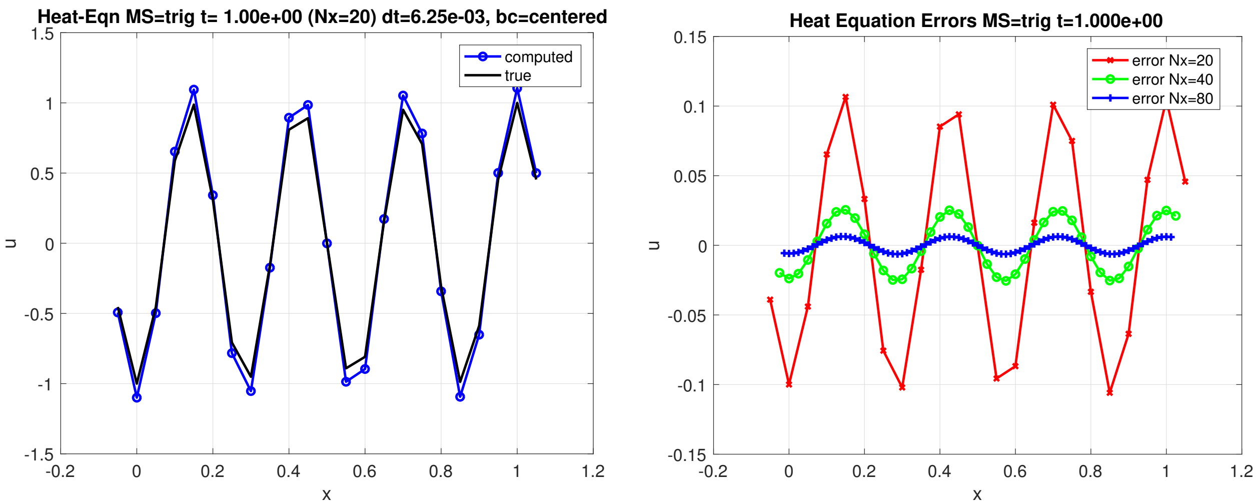

Figure 1: Results from problem 3(a).

3. (20 points) Mixed boundary conditions (one-sided).

3(a) Here is the AB2-CD2 scheme with 2nd-order accurate one-sided approximations for the boundary conditions,

$$
\begin{aligned}
\frac{U_i^{n+1}-U_i^n}{\Delta t} &= \frac{3}{2}\big(\kappa D_+D_-U_i^n + f(x_i,t^n)\big) \\
&\quad - \frac{1}{2}\big(\kappa D_+D_-U_i^{n-1} + f(x_i,t^{n-1})\big), \qquad i=1,2,\ldots,N_x-1, \quad n=0,1,2,\ldots \\
&\alpha_a U_0^{n+1} + \beta_a \mathcal{D}^+ U_0^{n+1} = g_a(t^{n+1}), \qquad n=0,1,2,\ldots \\
&\alpha_b U_{N_x}^{n+1} + \beta_b \mathcal{D}^- U_{N_x}^{n+1} = g_b(t^{n+1}), \qquad n=0,1,2,\ldots \\
&U_{-1}^{n+1} = 3U_0^{n+1} - 3U_1^{n+1} + U_2^{n+1}, \qquad n=0,1,2,\ldots \\
&U_{N_x+1}^{n+1} = 3U_{N_x}^{n+1} - 3U_{N_x-1}^{n+1} + U_{N_X-2}^{n+1}, \qquad n=0,1,2,\ldots \\
&U_i^0 = u_0(x_i), \quad U_i^{-1} = u^e(x_i,-\Delta t), \qquad i=0,1,2,3,\ldots,N_x,
\end{aligned}
$$

Listing 3: Results from 3(b) AB2-CD2, second-order one-sided, poly MS.

```
>> heatMixed
Heat: AB2, oneSided2, poly: t=1.00: Nx= 20 Nt= 160 dt=6.250e-03 maxErr=2.66e-15
Heat: AB2, oneSided2, poly: t=1.00: Nx= 40 Nt= 640 dt=1.563e-03 maxErr=6.22e-15 ratio=4.29e-01
Heat: AB2, oneSided2, poly: t=1.00: Nx= 80 Nt= 2560 dt=3.906e-04 maxErr=1.07e-14 ratio=5.83e-01
Heat: AB2, oneSided2, poly: t=1.00: Nx=160 Nt=10240 dt=9.766e-05 maxErr=1.69e-14 ratio=6.32e-01
Heat: AB2, oneSided2, poly: t=1.00: Nx=320 Nt=40960 dt=2.441e-05 maxErr=4.62e-14 ratio=3.65e-01
```

Listing 4: Results from 3(a) AB2-CD2, second-order one-sided.

```
>> heatMixed
Heat: AB2, oneSided2, trig: t=1.00: Nx= 20 Nt= 160 dt=6.250e-03 maxErr=8.31e-01
Heat: AB2, oneSided2, trig: t=1.00: Nx= 40 Nt= 640 dt=1.563e-03 maxErr=9.93e-02 ratio=8.37e+00
Heat: AB2, oneSided2, trig: t=1.00: Nx= 80 Nt= 2560 dt=3.906e-04 maxErr=1.15e-02 ratio=8.66e+00
Heat: AB2, oneSided2, trig: t=1.00: Nx=160 Nt=10240 dt=9.766e-05 maxErr=2.19e-03 ratio=5.22e+00
Heat: AB2, oneSided2, trig: t=1.00: Nx=320 Nt=40960 dt=2.441e-05 maxErr=4.71e-04 ratio=4.66e+00
```

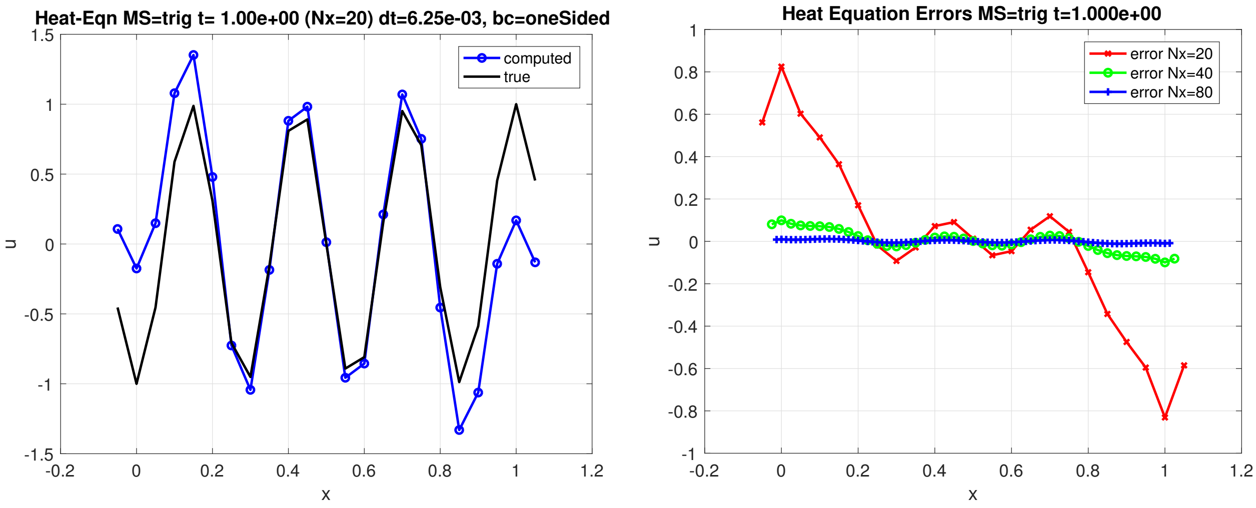

Figure 2: Results from problem 3(a).

3(b)

Listing 5: Results from 3(b) AB2-CD2, first-order one-sided, poly MS.

```
>> heatMixed
Heat: AB2, oneSided1, poly: t=1.00: Nx= 20 Nt= 160 dt=6.250e-03 maxErr=2.04e-02
Heat: AB2, oneSided1, poly: t=1.00: Nx= 40 Nt= 640 dt=1.563e-03 maxErr=8.83e-03 ratio=2.31e+00
Heat: AB2, oneSided1, poly: t=1.00: Nx= 80 Nt= 2560 dt=3.906e-04 maxErr=4.10e-03 ratio=2.15e+00
Heat: AB2, oneSided1, poly: t=1.00: Nx=160 Nt=10240 dt=9.766e-05 maxErr=1.97e-03 ratio=2.08e+00
Heat: AB2, oneSided1, poly: t=1.00: Nx=320 Nt=40960 dt=2.441e-05 maxErr=9.69e-04 ratio=2.04e+00
```

Listing 6: Results from 3(b) AB2-CD2, first-order one-sided, trig MS.

```
>> heatMixed
Heat: AB2, oneSided1, trig: t=1.00: Nx= 20 Nt= 160 dt=6.250e-03 maxErr=2.00e+00
Heat: AB2, oneSided1, trig: t=1.00: Nx= 40 Nt= 640 dt=1.563e-03 maxErr=1.02e+00 ratio=1.97e+00
Heat: AB2, oneSided1, trig: t=1.00: Nx= 80 Nt= 2560 dt=3.906e-04 maxErr=4.69e-01 ratio=2.16e+00
Heat: AB2, oneSided1, trig: t=1.00: Nx=160 Nt=10240 dt=9.766e-05 maxErr=2.22e-01 ratio=2.11e+00
Heat: AB2, oneSided1, trig: t=1.00: Nx=320 Nt=40960 dt=2.441e-05 maxErr=1.08e-01 ratio=2.06e+00
```

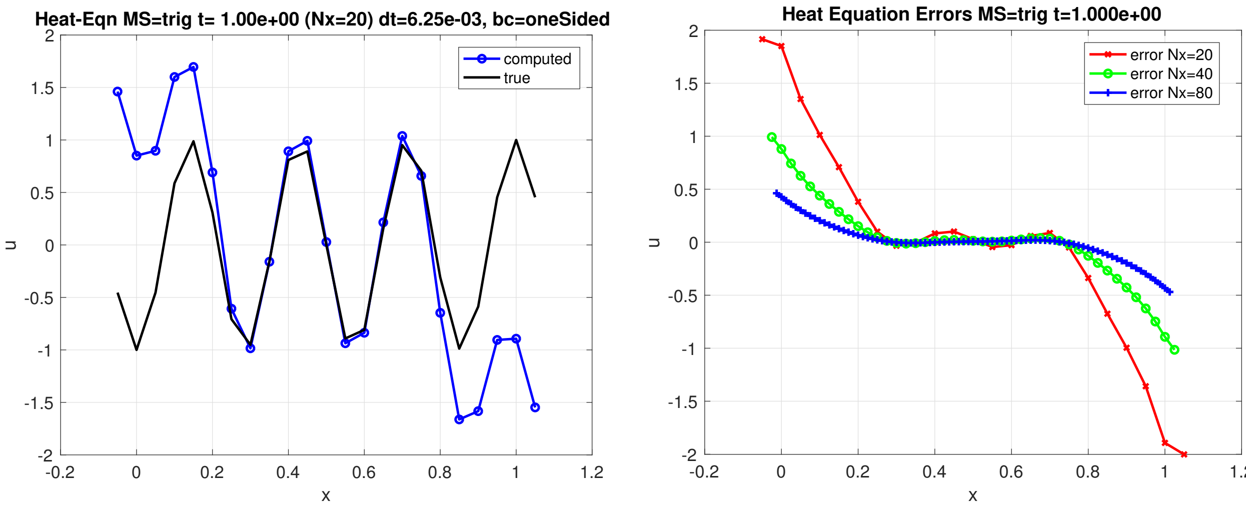

Figure 3: Results from problem 3(b).

Listing 7: heatMixed.m

```
%
% Solve the heat equation with mixed BC's and manufactured solutions
%
%   u_t = kappa * u_xx + f(x,t),  a < x < b , 0 < t <= tFinal
%   alphaA*u(a,t) + betaA*ux(a,t) = ga(t),
%   alphaB*u(b,t) + betaB*ux(b,t) = gb(t)
%   u(x,0) = u_0(x)
%
clear; clf; fontSize=14; lineWidth=2;

kappa=.1;    % coefficient of diffusion
a=0.; b=1.;  % space interval interval
tFinal=1.;   % final time
cfl=1.;

alphaA=.5; betaA=-1.; % coefficients of the mixed BC
alphaB=.25; betaB=+2.;

% Time-stepping
ts = 'FE'; % forward Euler
ts = 'AB2'; % Adams-Bashforth order 2

% Choose the manufactured solution:
ms = 'poly';
ms = 'trig';  % un-comment for trig MS

% Choose BC option
bcOption = 'centered'; bcOrder=2;
bcOption = 'oneSided'; bcOrder=2;
bcOption = 'oneSided'; bcOrder=1;

if( strcmp(ms,'poly') )
  % Polynomial manufactured solution:
  % b0=.5; b1=.7; b2=.0; % time coefficients
  b0=.5; b1=.7; b2=.9; % time coefficients
```

```
  c0=1.; c1=.8; c2=.6; % space coefficients
  % c0=1.; c1=.8; c2=.0; % space coefficients
  ue   = @(x,t) (b0+b1*t+b2*t^2)*(c0+c1*x+c2*x.^2);
  uet  = @(x,t) (b1+2.*b2*t  )*(c0+c1*x+c2*x.^2);
  uex  = @(x,t) (b0+b1*t+b2*t^2)*(c1+2.*c2*x);
  uexx = @(x,t) (b0+b1*t+b2*t^2)*(2.*c2);
else
  % Trigonometric manufactured solution:
  kt = pi; kx=7*pi;
  ue   = @(x,t) cos(kt*t)*cos(kx*x);
  uet  = @(x,t) -kt*sin(kt*t)*cos(kx*x);
  uex  = @(x,t) -kx*cos(kt*t)*sin(kx*x);
  uexx = @(x,t) (-kx^2)*ue(x,t);
end

ga = @(t) alphaA*ue(a,t) + betaA*uex(a,t); % BC RHS at x=a
gb = @(t) alphaB*ue(b,t) + betaB*uex(b,t); % BC RHS at x=b
u0 = @(x) ue(x,0); % initial condition function
f  = @(x,t) uet(x,t) - kappa*uexx(x,t); % forcing


numResolutions=5;
for m=1:numResolutions
  Nx=10*2^m;      % number of space intervals
  dx=(b-a)/Nx; % grid spacing
  numGhost=1;
  Nd = Nx + 1 + 2*numGhost; % total number of points
  x = linspace(a-numGhost*dx,b+numGhost*dx,Nd)'; % spatial grid
  if( abs(x(2)-x(1) -dx) > 1.e-12*dx ) fprintf('ERROR in dx!\n'); pause; pause; end;

  % allocate space for the solution at two levels
  um   = zeros(Nd,1);   % holds U_i^{n-1}
  un   = zeros(Nd,1);   % holds U_i^n
  unp1 = zeros(Nd,1);   % holds U_i^{n+1}

  dt = cfl*.25*dx^2/kappa;  % time step (adjusted below)
  Nt = round(tFinal/dt); % number of time-steps
  % fprintf('dt=%9.3e, adjusted=%9.3e, diff=%8.2e\n',dt,tFinal/Nt,abs(dt-tFinal/Nt));
  dt = tFinal/Nt;      % adjust dt to reach tFinal exactly

  ia=numGhost+1;  % index of boundary point at x=a
  ib=ia + Nx;     % index of boundary point at x=b
  i1=ia+1;        % first interior pt
  i2=ib-1;        % last interior pt
  I  = i1:i2; % interior points
  Ib = ia:ib; % interior + boundary points

  % Define D+D- operator
  DpDm = @(u,I) (u(I+1)-2.*u(I)+u(I-1))/(dx^2);

  t=0.;
  un = u0(x); % initial conditions
  um = ue(x,t-dt); % set old time to exact
  % --- Start time-stepping loop ---
  for( n=1:Nt )

    tm = (n-1)*dt; % old time
    t = n*dt;      % new time

```

```
  if( strcmp(ts,'FE') )
    % Forward Euler in time:
    unp1(Ib) = un(Ib) + (kappa*dt)*DpDm(un,Ib) + dt*f(x(Ib),tm);
  else
    % AB2
    unp1(Ib) = un(Ib) + dt*( 1.5*( kappa*DpDm(un,Ib) + f(x(Ib),t-dt ) ) ...
                             -.5*( kappa*DpDm(um,Ib) + f(x(Ib),t-2*dt) ) );
  end

  if( strcmp(bcOption,'centered') )
    % Centered BC: assign ghost:
    unp1(ia-1)=unp1(ia+1) + (2.*dx/betaA)*( alphaA*unp1(ia) - ga(t) ); % set left ghost from BC at x=a
    unp1(ib+1)=unp1(ib-1) - (2.*dx/betaB)*( alphaB*unp1(ib) - gb(t) ); % set right ghost from BC at x=b
  else
    % one-sided BC
    if( bcOrder==1 )
      % first-order one-sided
      unp1(ia) = ( ga(t) - betaA*( unp1(ia+1) )/dx )/( alphaA - betaA/dx );
      unp1(ib) = ( gb(t) - betaB*( -unp1(ib-1) )/dx )/( alphaB + betaB/dx );
    else
      % second-order one-sided
      unp1(ia) = ( ga(t) - betaA*( 4*unp1(ia+1) - unp1(ia+2) )/(2.*dx) )/( alphaA - 3.*betaA/(2.*dx) );
      unp1(ib) = ( gb(t) - betaB*( -4*unp1(ib-1) + unp1(ib-2) )/(2.*dx) )/( alphaB + 3.*betaB/(2.*dx) );
    end
    % extrapolate ghost
    unp1(ia-1)=3*unp1(ia)-3*unp1(ia+1)+unp1(ia+2);
    unp1(ib+1)=3*unp1(ib)-3*unp1(ib-1)+unp1(ib-2);
  end;


  um=un;  % set um <- un for next step
  un=unp1; % Set un <- unp1 for next step

end;
% --- End time-stepping loop ---

uexact = ue(x,t);        % eval exact solution:
err = un-uexact;         % error
errMax(m) = max(abs(err)); % max-norm error
fprintf('Heat:␣%s,␣%s%d,␣%s:␣t=%4.2f:␣Nx=%3d␣Nt=%5d␣dt=%9.3e␣maxErr=%8.2e',ts,bcOption,bcOrder,ms,t,Nx,Nt,dt,errMax(m));
if( m==1 ) fprintf('\n'); else fprintf('␣ratio=%8.2e\n',errMax(m-1)/errMax(m)); end;

% plot results
figure(1);
plot( x,un,'b-o', x,uexact,'k-','Linewidth',lineWidth);
legend('computed','true');
title(sprintf('Heat-Eqn␣MS=%s␣t=%9.2e␣(Nx=%d)␣dt=%8.2e,␣bc=%s',ms,t,Nx,dt,bcOption));
xlabel('x'); ylabel('u'); set(gca,'FontSize',fontSize); grid on;
if( m==1 )
  print('-depsc2',sprintf('heatMixedMS%s%s%d.eps',ms,bcOption,bcOrder)); % save as an eps file
end;

% plot error
figure(2);
```

```
  plot(x,err,'b-x','Linewidth',lineWidth);
  legend('Error');
  title(sprintf('Heat Equation Error t=%9.3e bc=%s',t,bcOption));
  xlabel('x'); ylabel('u'); set(gca,'FontSize',fontSize); grid on;

  % save grid and errors in a data structure
  data{m}.x=x; data{m}.err=err;

  pause(1);

end; % end for m


% plot errors
figure(2);
% plot(x1,err1,'r-x', x2,err2,'g-o',x3,err3,'b-+','Linewidth',lineWidth);
plot(data{1}.x,data{1}.err,'r-x', data{2}.x,data{2}.err,'g-o',data{3}.x,data{3}.err,'b-+','
    Linewidth',lineWidth);
legend('error Nx=20','error Nx=40','error Nx=80','Location','best');
title(sprintf('Heat Equation Errors MS=%s t=%9.3e',ms,t));
xlabel('x'); ylabel('u'); set(gca,'FontSize',fontSize); grid on;
fileName=sprintf('heatMixedMS%s%s%dErrors.eps',ms,bcOption,bcOrder);
print('-depsc2',fileName); % save as an eps file
fprintf('Wrote file=[%s] with errors\n',fileName);
```

4. (20 points) (Implicit time-stepping)

4(a) Substituting $u(x,t) = \hat{u}(t)\,\sin(k\pi x)$ into the PDE leads to the ODE

$$\frac{d\hat{u}}{dt} = -(\kappa(k\pi)^2 + \alpha)\hat{u}$$

which gives the exact solution

$$u(x,t) = e^{-(\kappa(k\pi)^2+\alpha)\,t}\,\sin(k\pi x)$$

4(b) Here is the theta-CD2 scheme

$$\begin{aligned}
\frac{U_i^{n+1} - U_i^n}{\Delta t} &= \theta\left(\kappa D_+ D_- U_i^{n+1} - \alpha U^{n+1}\right) \\
&\quad + (1-\theta)\left(\kappa D_+ D_- U_i^n - \alpha U^n\right), \qquad i = 1,2,3,\ldots,N_x - 1, \quad n = 0,1,2,\ldots \\
U_0^{n+1} &= g_a(t^{n+1}), \qquad U_{N_x}^{n+1} = g_b(t^{n+1}), \quad n = 0,1,2,\ldots \\
U_i^0 &= u_0(x_i), \qquad i = 0,1,2,3,\ldots,N_x.
\end{aligned}$$

4(c) The equation-truncation error (ETE) for the theta-CD2 scheme is

$$\tau_i^n = D_{+t}u(x_i,t^n) - (\kappa D_+ D_- - \alpha I)\big(\theta u(x_i,t^{n+1}) - (1-\theta)u(x_i,t^n)\big)$$

Using (if not specified, u and it's derivatives are evaluated at (x_i,t^n)),

$$\begin{aligned}
D_{+t}u(x_i,t^n) &= u_t + \frac{\Delta t}{2}u_{tt} + \frac{\Delta t^2}{6}u_{ttt} + \mathcal{O}(\Delta t^3), \\
D_+ D_- u &= u_{xx} + \frac{\Delta x^2}{12}u_{xxxx} + \mathcal{O}(\Delta x^4), \\
u(x_i,t^{n+1}) &= u + \Delta t u_t + \frac{\Delta t^2}{2}u_{ttt} + \mathcal{O}(\Delta t^3)
\end{aligned}$$

Thus

$$\begin{aligned}
D_+ D_- u(x_i,t^{n+1}) &= u_{xx} + \frac{\Delta x^2}{12}u_{xxxx} + \Delta t\, u_{txx} + \frac{\Delta t^2}{2}u_{tttxx} + \mathcal{O}(\Delta t\Delta x^2 + \Delta x^4) \\
\theta u(x_i,t^{n+1}) + (1-\theta)u &= u + \theta\Delta t u_t + \theta\frac{\Delta t^2}{2}u_{ttt} + \mathcal{O}(\Delta t^3)
\end{aligned}$$

Substituting these into the expression for τ_i^n and using $u_t = \kappa u_{xx} - \alpha u$ (to convert everything to time-derivatives of u) gives (*check me*)

$$\tau_i^n = \Delta t(\frac{1}{2} - \theta)u_{tt} - \kappa\frac{\Delta x^2}{12}u_{xxxx} + \frac{\Delta t^2}{6}\Big((1-3\theta)u_{ttt} + 6\theta\alpha^3 u\Big) + \mathcal{O}(\Delta t^3 + \Delta t\Delta x^2 + \Delta x^4)$$

We see the scheme is second-order accurate in time if $\theta = \frac{1}{2}$ and first order in time otherwise. The scheme is second-order accurate in space.

4(d) The code listing is given below.

Case 1. $\theta = 0.5$:

Listing 8: Results from 4(d), $\theta = 0.5$

```
>> heatThetaMethod
Solve the modified equation heat: theta=0.50
theta=0.5: t=5.0000e-01: Nx= 20 Nt= 10 dt=5.000e-02 maxErr=1.43e-03
theta=0.5: t=5.0000e-01: Nx= 40 Nt= 20 dt=2.500e-02 maxErr=3.58e-04 ratio=4.00e+00
theta=0.5: t=5.0000e-01: Nx= 80 Nt= 40 dt=1.250e-02 maxErr=8.83e-05 ratio=4.05e+00
theta=0.5: t=5.0000e-01: Nx=160 Nt= 80 dt=6.250e-03 maxErr=2.20e-05 ratio=4.01e+00
```

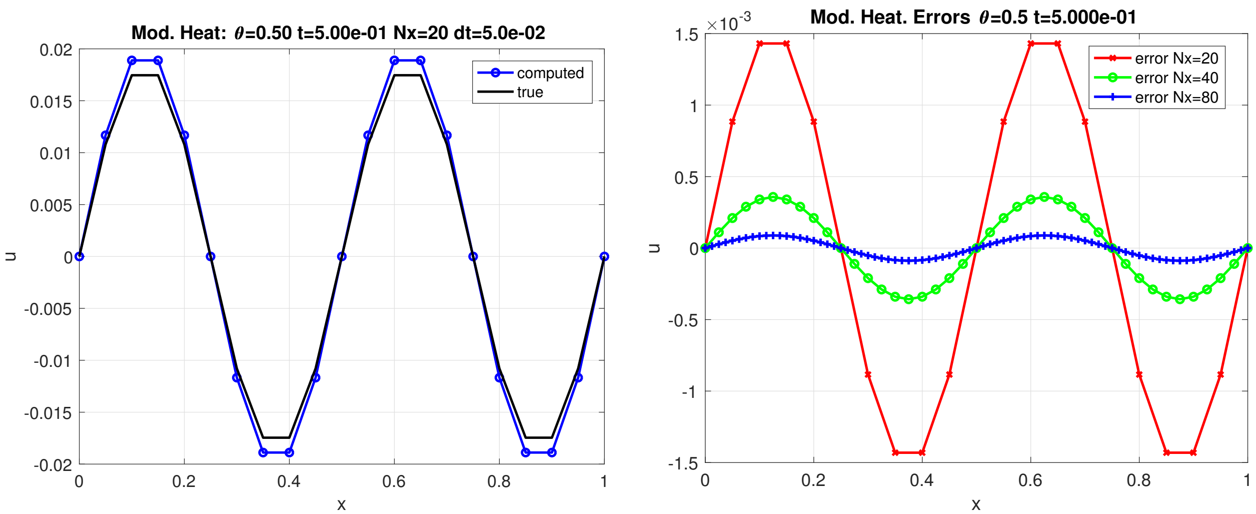

Figure 4: Results from problem 4(d), $\theta = 0.5$.

Case 2. $\theta = 1.0$:

Listing 9: Results from 4(d), $\theta = 1.00$

```
>> heatThetaMethod
Solve the modified equation heat: theta=1.00
theta=1.0: t=5.0000e-01: Nx= 20 Nt= 10 dt=5.000e-02 maxErr=1.86e-02
theta=1.0: t=5.0000e-01: Nx= 40 Nt= 20 dt=2.500e-02 maxErr=8.49e-03 ratio=2.20e+00
theta=1.0: t=5.0000e-01: Nx= 80 Nt= 40 dt=1.250e-02 maxErr=3.95e-03 ratio=2.15e+00
theta=1.0: t=5.0000e-01: Nx=160 Nt= 80 dt=6.250e-03 maxErr=1.90e-03 ratio=2.07e+00
```

Case 3. $\theta = 0.25$:

Listing 10: Results from 4(d), $\theta = 0.25$

```
>> heatThetaMethod
Solve the modified equation heat: theta=0.25
theta=0.2: t=5.0000e-01: Nx= 20 Nt= 10 dt=5.000e-02 maxErr=5.20e-03
theta=0.2: t=5.0000e-01: Nx= 40 Nt= 20 dt=2.500e-02 maxErr=3.17e-03 ratio=1.64e+00
theta=0.2: t=5.0000e-01: Nx= 80 Nt= 40 dt=1.250e-02 maxErr=9.18e-03 ratio=3.45e-01
theta=0.2: t=5.0000e-01: Nx=160 Nt= 80 dt=6.250e-03 maxErr=1.71e+17 ratio=5.38e-20
```

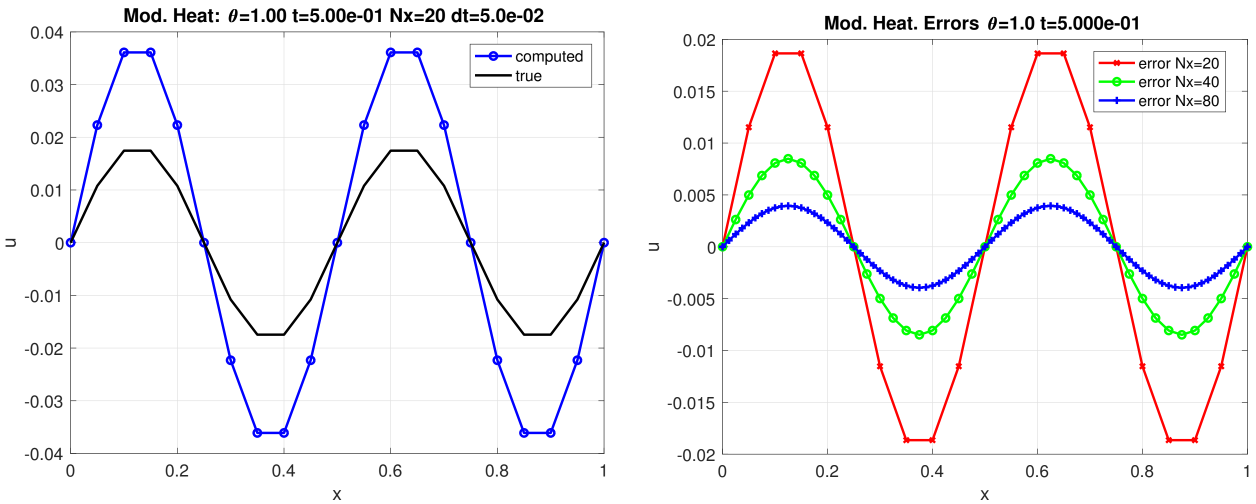

Figure 5: Results from problem 4(d), $\theta = 1.0$.

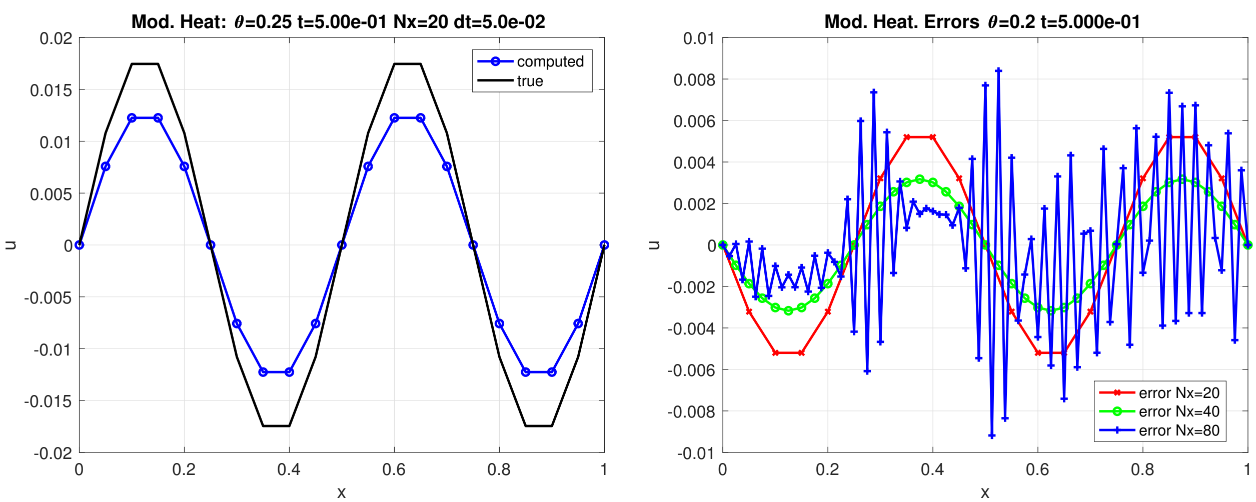

Figure 6: Results from problem 4(d), $\theta = 0.25$.

Listing 11: heatThetaMethod.m

```
%
%  Solve the heat equation with the theta-method
%
%    u_t = kappa * u_xx -alpha*u a < x < b , 0 < t <= tFinal
%    u(a,t) = ga(t), u(b,t)=gb(t)
%    u(x,0) = u_0(x)
%
clear; clf; fontSize=14; lineWidth=2;

theta=.5; % BE: theta=1, FE: theta=0, CN theta=.5
theta=1;
theta=.25;
% method Name : (replace '.' by 'p' so we can use in file names)
methodName = regexprep(sprintf('Theta%3.2f',theta),'\.','p'); % theta0p5 or theta1p0 etc.

kappa=.05;   % coefficient of diffusion
alpha=.1;
kx=4.*pi;    % wave number in the IC and exact solution
a=0.; b=1.;  % space interval interval
tFinal=.5;   % final time

ga = @(t) 0 ; % BC RHS at x=a
gb = @(t) 0 ; % BC RHS at x=b
u0 = @(x) sin(kx*x); % initial condition function
% exact solution function:
uexact = @(x,t) sin(kx*x)*exp((-kappa*kx^2-alpha)*t);

fprintf('Solve the modified equation heat: theta=%4.2f\n',theta);
numResolutions=4;
for m=1:numResolutions
  Nx=10*2^m;               % number of space intervals
  dx=(b-a)/Nx;             % grid spacing
  x = linspace(a,b,Nx+1)'; % spatial grid

  % allocate space for the solution at two levels
  un = zeros(Nx+1,1);   % holds U_i^n
  unp1 = zeros(Nx+1,1); % holds U_i^{n+1}

  dt=dx;  % time step (adjusted below)
  Nt = round(tFinal/dt); % number of time-steps
  dt = tFinal/Nt;      % adjust dt to reach tFinal exactly

  ia=1;   % index of boundary point at x=a
  ib=Nx+1; % index of boundary point at x=b
  i1=ia+1; % first interior pt
  i2=ib-1; % last interior pt
  I = i1:i2;

  % Form the implicit matrix
  rhs=zeros(Nx,1);
  A=sparse(Nx+1,Nx+1);
  for i=i1:i2
    A(i,i-1)=  -theta*dt*( kappa*( 1./dx^2) );
    A(i,i) = 1 -theta*dt*( kappa*(-2./dx^2) -alpha );
    A(i,i+1)=  -theta*dt*( kappa*( 1./dx^2) );
  end;
  A(ia,ia)=1.; % BC at x=a
```

```
  A(ib,ib)=1.; % BC at x=b

  % Define D+D- operator
  DpDm = @(u,I) (u(I+1)-2.*u(I)+u(I-1))/(dx^2);

  t=0.;
  un = u0(x); % initial conditions
  % --- Start time-stepping loop ---
  for( n=1:Nt )

    t = n*dt;       % new time
    % Theta-method
    rhs(I) = un(I) + ((1.-theta)*dt)*( kappa*DpDm(un,I) - alpha*un(i1:i2) );
    rhs(ia)=ga(t); % BC at x=a
    rhs(ib)=gb(t); % BC at x=b

    unp1 = A\rhs;

    un=unp1; % Set un <- unp1 for next step
  end;
  % --- End time-stepping loop ---

  ue = uexact(x,t); % eval exact solution:
  err = un-ue;
  errMax(m) = max(abs(err)); % max-norm error
  fprintf('theta=%3.1f: t=%10.4e: Nx=%3d Nt=%4d dt=%9.3e maxErr=%8.2e',theta,t,Nx,Nt,dt,errMax(m));
  if( m==1 ) fprintf('\n'); else fprintf(' ratio=%8.2e\n',errMax(m-1)/errMax(m)); end;

  % plot results
  figure(1);
  plot( x,un,'b-o', x,ue,'k-','Linewidth',lineWidth);
  legend('computed','true');
  title(sprintf('Mod. Heat: \\theta=%4.2f t=%8.2e Nx=%d dt=%7.1e',theta,t,Nx,dt));
  xlabel('x'); ylabel('u'); set(gca,'FontSize',fontSize); grid on;

  if( m==1 )
    print('-depsc2',sprintf('heatEquation%s.eps',methodName)); % save as an eps file
  end;

  % plot error
  figure(2);
  plot(x,err,'b-x','Linewidth',lineWidth);
  legend('Error');
  title(sprintf('Mod. Heat: Error: \\theta=%3.1f t=%8.2e Nx=%d dt=%7.1e',theta,t,Nx,dt));
  xlabel('x'); ylabel('u'); set(gca,'FontSize',fontSize); grid on;

  % save grid and errors in a data structure
  data{m}.x=x; data{m}.err=err;

  pause(1);

end; % end for m


% plot errors
figure(2);
% plot(x1,err1,'r-x', x2,err2,'g-o',x3,err3,'b-+','Linewidth',lineWidth);
plot(data{1}.x,data{1}.err,'r-x', data{2}.x,data{2}.err,'g-o',data{3}.x,data{3}.err,'b-+','Linewidth',lineWidth);
```

```
legend('error Nx=20','error Nx=40','error Nx=80','Location','best');
title(sprintf('Mod. Heat. Errors \\theta=%3.1f t=%9.3e',theta,t));
xlabel('x'); ylabel('u'); set(gca,'FontSize',fontSize); grid on;

fileName = sprintf('heatEquation%sErrors.eps',methodName);
print('-depsc2',fileName); % save as an eps file
fprintf('Wrote file=[%s]\n',fileName);
```