Unsteady Heat Equation 1D with Galerkin Method

Nurul Farahain Mohammad

October 23, 2012

Consider a heat transfer problem

$$u_t - u_{xx} = f(x,t), \qquad (x,t) \in (0,1) \times (0,\pi), \tag{1}$$

with initial and boundary conditions

$$\begin{aligned} &u(x,0) = 0, \\ &u(0,t) = \sin t, \;\; u(1,t) = e^{-1}\sin t. \end{aligned} \tag{2}$$

and function $f(x,t) = e^{-x^2}(\cos t - 2(2x^2-1)\sin t)$.

This problem has been solved analytically and the exact solution is given below

$$u(x,t) = e^{-x^2}\sin t. \tag{3}$$

Now, we will try to solve this problem by using Galerkin Method. We will discretize the space x with Finite Element Method and the time t with Forward Euler Method.

Since we are dealing with non-zero boundary conditions, a function $\tilde{u}(x,t) = u(x,t) - w(x,t)$ will be introduced in order to fit our problem with existing Finite Element scheme of this problem in zero boundary conditions (refer Suli, Finite Element Methods for Partial Differential Equations, 2012).

We start with making ansatz on function $w(x,t)$. Let

$$w(x,t) = e^{-x}\sin t. \tag{4}$$

Verify function $w(x,t)$,

$$\begin{aligned} &w(x,0) = e^{-x}\sin 0 = 0 = u(x,0), \\ &w(0,t) = e^{0}\sin t = \sin t = u(0,t), \\ &w(1,t) = e^{-1}\sin t = u(1,t). \end{aligned} \tag{5}$$

Re-state the problem in terms of $\tilde{u}(x,t)$,

$$\tilde{u}_t - \tilde{u}_{xx} = f(x,t) - w_t + w_{xx}, \tag{6}$$

with initial and boundary conditions

$$\begin{aligned} &\tilde{u}(x,0) = 0, \\ &\tilde{u}(0,t) = 0,\ \tilde{u}(1,t) = 0. \end{aligned} \tag{7}$$

1 Weak Form

Multiply (6) by $v \in H_0^1(\Omega)$,

$$\tilde{u}_t v - \tilde{u}_{xx} v = fv - w_t v + w_{xx} v, \tag{8}$$

Integrate (8),

$$\int_0^1 \tilde{u}_t v \,\mathrm{d}x - \int_0^1 \tilde{u}_{xx} v \,\mathrm{d}x = \int_0^1 fv \,\mathrm{d}x - \int_0^1 w_t v \,\mathrm{d}x + \int_0^1 w_{xx} v \,\mathrm{d}x, \tag{9}$$

Applying the technique of integration by-part on a term below,

$$\begin{aligned} \int_0^1 \tilde{u}_{xx} v \,\mathrm{d}x &= \tilde{u}' v|_0^1 - \int_0^1 \tilde{u}' v' \,\mathrm{d}x, \\ &= \tilde{u}'(1)v(1) - \tilde{u}'(0)v(0) - \int_0^1 \tilde{u}' v' \,\mathrm{d}x, \\ &\text{since v(0) = v(1) = 0,} \\ &= -\int_0^1 \tilde{u}' v' \,\mathrm{d}x. \end{aligned} \tag{10}$$

Updating Equation (9), we will obtain the weak form of the problem. Find $u \in H_0^1(\Omega)$ ∋

$$\int_0^1 \tilde{u}_t v \,\mathrm{d}x + \int_0^1 \tilde{u}' v' \,\mathrm{d}x = \int_0^1 fv \,\mathrm{d}x - \int_0^1 w_t v \,\mathrm{d}x + \int_0^1 w'' v \,\mathrm{d}x, \tag{11}$$

or

$$(\tilde{u}_t, v) + (\tilde{u}', v') = (f, v) - (w_t, v) + (w'', v), \tag{12}$$

with $\tilde{u} = 0$ on $\partial\Omega$.

Next, we will define our problem in finite space.

2 Galerkin Method

Find $\tilde{u}_h \in \langle \varphi_1, \varphi_2, \cdots, \varphi_n \rangle$ which φ_i is shape function ∋

$$((\tilde{u}_h)_t, v) + (\nabla(\tilde{u}_h), \nabla v) = (f, v) - (w_t, v) - (w'', v), \tag{13}$$

Remark: $\tilde{u}_h$ can also be written as

$$\begin{aligned}\tilde{u}_h &= C_1\varphi_1 + \cdots + C_n\varphi_n, \\ &= \sum_{i=1}^{n} C_i\varphi_i.\end{aligned} \tag{14}$$

Write Equation (13) as a system of linear equations,

$$\begin{aligned}\sum \frac{dC_i}{dt}(\varphi_i,\varphi_1) + \sum C_i(\nabla\varphi_i,\nabla\varphi_1) &= (f_1,\varphi_1) - ((w_t)_1,\varphi_1) + (w_1'',\varphi_1) \\ &\vdots \\ \sum \frac{dC_i}{dt}(\varphi_i,\varphi_n) + \sum C_i(\nabla\varphi_i,\nabla\varphi_n) &= (f_n,\varphi_n) - ((w_t)_n,\varphi_n) + (w_n'',\varphi_n)\end{aligned} \tag{15}$$

which

$$\varphi_i = \begin{cases} \dfrac{x - x_{i-1}}{x_i - x_{i-1}} & x \in [x_{i-1}, x_i], \\ \dfrac{x_{i+1} - x}{x_{i+1} - x_i} & x \in [x_i, x_{i+1}], \\ 0 & \text{elsewhere.} \end{cases}$$

We can also write system of equations (15) in matrix form

$$M\frac{dC}{dt} + AC = F - W_t + W'', \tag{16}$$

which

$$M = \frac{h}{6}\begin{pmatrix} 4 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 4 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 4 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 1 & 4 \end{pmatrix}$$

and

$$A = \frac{1}{h}\begin{pmatrix} 2 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & -1 & 2 \end{pmatrix}$$

After this, we will go through a suggested numerical integration method to compute load vector named Mid-Point rule before we cover discretization of time by using the Forward or Backward Euler Method.

3 Mid-point rule

In order to obtain the inner products of (f_i, φ_i), $((w_t)_i, \varphi_i)$ and (w_i'', φ_i), we apply the mid-point rule on these terms.

Let say we have two functions $p(x)$ and $q(x)$ dependent on x. To find the inner product of p and q,

$$(p_i, q_i) = \int_{x_i}^{x_i+h} p_i q_i \,\mathrm{d}x \approx hf\Big(x_i + \frac{h}{2}\Big). \tag{17}$$

In example, for our case study

$$f_i\varphi_i = \begin{cases} e^{-x^2}(\cos t - 2(2x^2-1)\sin t)\dfrac{x - x_{i-1}}{x_i - x_{i-1}} & x \in [x_{i-1}, x_i], \\ e^{-x^2}(\cos t - 2(2x^2-1)\sin t)\dfrac{x_{i+1} - x}{x_{i+1} - x_i} & x \in [x_i, x_{i+1}], \\ 0 & \text{elsewhere.} \end{cases}$$

By using the mid-point rule,

$$\begin{aligned} \int_{x_{i-1}}^{x_{i+1}} f_i\varphi_i \,\mathrm{d}x &= \int_{x_{i-1}}^{x_i} f_i\varphi_i \,\mathrm{d}x + \int_{x_i}^{x_{i+1}} f_i\varphi_i \,\mathrm{d}x, \\ &\approx hf\Big(x_{i-1} + \frac{h}{2}\Big)\varphi\Big(x_{i-1} + \frac{h}{2}\Big) + hf\Big(x_i + \frac{h}{2}\Big)\varphi\Big(x_i + \frac{h}{2}\Big), \\ &= \frac{h}{2}\Big[e^{-(x_{i-1}+\frac{h}{2})^2}\cos t^n - 2[2(x_{i-1})^2 - 1]\sin t^n\Big] \\ &\quad + \frac{h}{2}\Big[e^{-(x_i+\frac{h}{2})^2}\cos t^n - 2[2(x_i)^2 - 1]\sin t^n\Big]. \end{aligned} \tag{18}$$

$$\int_{x_{i-1}}^{x_{i+1}} (w_t)_i\varphi_i \,\mathrm{d}x = \frac{h}{2}\Big[e^{-x_{i-1}-\frac{h}{2}}\cos t^n\Big] + \frac{h}{2}\Big[e^{-x_i-\frac{h}{2}}\cos t^n\Big]. \tag{19}$$

$$\int_{x_{i-1}}^{x_{i+1}} (w'')_i\varphi_i \,\mathrm{d}x = \frac{h}{2}\Big[e^{-x_{i-1}-\frac{h}{2}}\sin t^n\Big] + \frac{h}{2}\Big[e^{-x_i-\frac{h}{2}}\sin t^n\Big]. \tag{20}$$

4 Forward / Backward Euler Method

Let

$$\frac{dC}{dt} = \frac{F - W_t - W'' - AC}{M}, \tag{21}$$

Apply Forward Euler Method on Equation (21),

$$C^{m+1} = \triangle t\Big(\frac{F^m - W_t^m + W''^m - AC^m}{M}\Big) + C^m, \tag{22}$$

or apply Backward Euler Method on Equation (21),

$$C^{m+1} = \frac{F^{m+1} - W_t^{m+1} + W''^{m+1} + \triangle tMC^m}{\triangle tM + A}. \tag{23}$$

Don't get too excited. We just finished solving the problem in terms of $\tilde{u}$.

5 Full Solution u

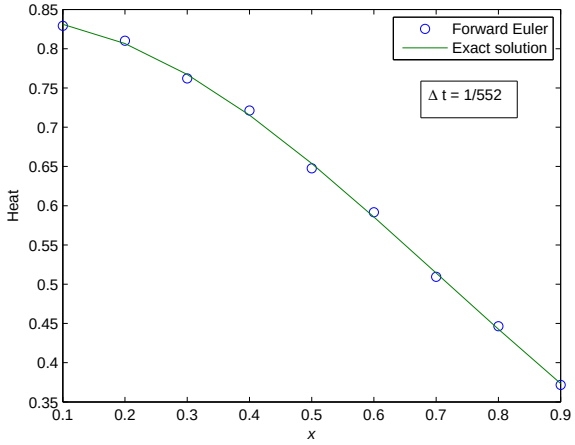
Finally, we can obtain $u = \tilde{u} + w$ by computing

$$U^m = C^m + W^m \tag{24}$$

6 Results and Discussion

For the 1st trial, this problem is solved using the Forward Euler method. But this method is conditionally stable, meaning at certain circumstances it is not stable and it's solution will 'go crazy' and give you 'headache'.

Figure 1: Final solution for $\triangle t = 1/552$



At $\triangle x = 0.1$ and $\triangle t = 1/552$, as you can see from Figure 1, the approximate solution is relatively close with the exact solution. As the $\triangle t$ grows slightly bigger ($\triangle t = 1/551$), we can see clearly from Figure 2 that the final solution of the approximate solution no longer relatively close to the exact solution. Based on Figure 3 and 4, when $\triangle t = \triangle x = 0.1$, the approximate solution at $t \geq 0.3$ starts to fluctuate.

It is advised or encouraged to use Backward Euler Method since it is **unconditionally** stable. Figure 5 shows the final solution for $\triangle t = \triangle x = 0.1$.

Figure 2: Final solution for $\triangle t = 1/551$

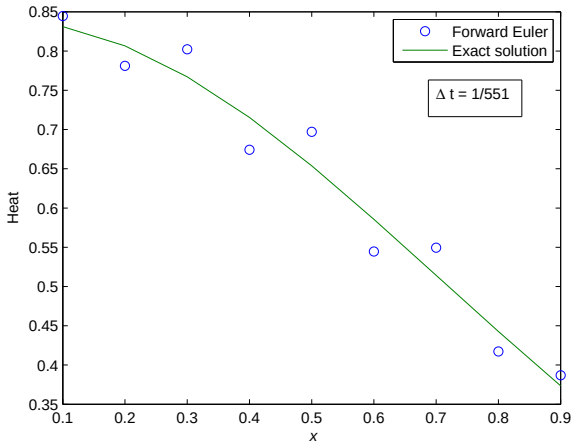


Figure 3: Final solution for $\triangle t = 0.1$

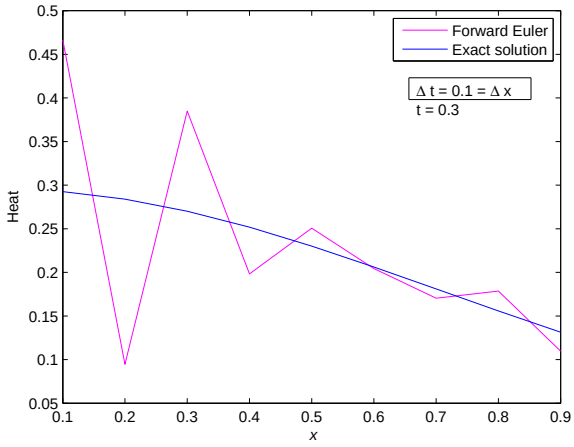


Figure 4: Final solution for $\triangle t = 0.1$

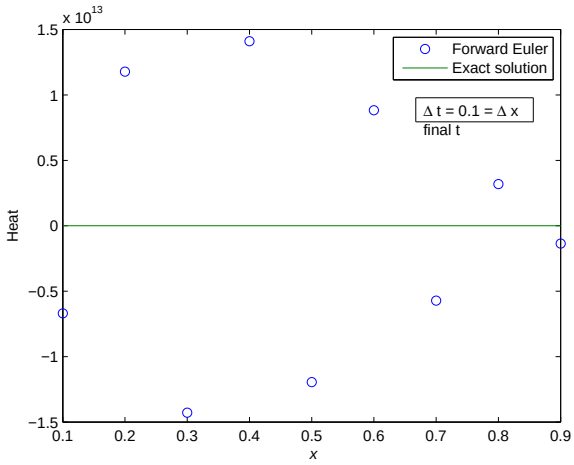
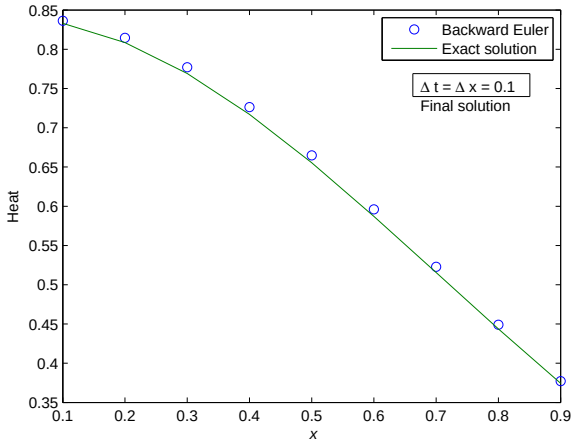


Figure 5: Final solution for $\triangle t = 0.1$ by using Backward Euler method



7 MATLAB program

```
clear all;

nt = 551; % number of time levels
delt = 1/552; nx = 11; delx = 0.1;


for n = 1:nt
    if n == 1
        t(1) = 0.0;
    else
        t(n) = t(n-1) + delt;
    end
end

x(1) = 0.0; x(nx) = 1.0;

for i = 2:nx-1
    x(i) = x(i-1) + delx;
end

BC = zeros(1,nx-2);

%compute A and M
m = full(blktridiag(4,1,1,nx-2)); M = (delx/6)*m;

a = full(blktridiag(2,-1,-1,nx-2)); A = (1/delx)*a;

C = zeros(nx-2,1,1);

%compute F, WT, WXX, W
 for n = 1:nt
     for i = 2:nx-1

         F(i,1,n) = (delx/2)*(exp(-((x(i-1)+delx/2))^2)*(
             cos(t(n)) -2*(2*(x(i-1)+delx/2)^2-1)*sin(t(n)
             ))) + (delx/2)*(exp(-((x(i)+delx/2))^2)*(cos(
             t(n)) -2*(2*(x(i)+delx/2)^2-1)*sin(t(n))));

         WT(i,1,n) = exp(-x(i-1)-delx/2)*cos(t(n))*(delx
             /2) +exp(-x(i)-delx/2)*cos(t(n))*(delx/2);

         WXX(i,1,n) = (delx/2)*exp(-x(i-1)-delx/2)*sin(t(
             n)) + (delx/2)*exp(-x(i)-delx/2)*sin(t(n));
```

```
39
40              G(i,1,n) = F(i,1,n) - WT(i,1,n) + WXX(i,1,n);
41
42              W(i,1,n) = exp(-x(i))*sin(t(n));
43
44              exact(i,1,n) = exp(-x(i)*x(i))*sin(t(n));
45          end
46 %
47  end
48
49 F(1,:,:) = []; WT(1,:,:) = []; WXX(1,:,:) = []; G(1,:,:)
       = [];
50 W(1,:,:) = []; exact(1,:,:) = [];
51
52 %%Forward Euler on time discretisation%%
53 for n = 1:nt-1
54      C(:,1,n+1) = M \ (delt*G(:,1,n)-delt*A*C(:,1,n)) +C
            (:,1,n);
55 end
56
57 %%Backward Euler on time discretisation%%
58 for n = 1:nt-1
59      C(:,1,n+1) = (delt*M + A)\(G(:,1,n+1)+ (delt*M*C(:,1,
            n)));
60 end
61
62 %%Compute U
63 for n = 1:nt
64      U(:,1,n) = C(:,1,n) +W(:,1,n);
65 end
66
67 plot(x(2:nx-1),U(:,1,n),'o',x(2:nx-1),exact(:,1,n));
```