Homework 3

James Eckstein

April 8, 2020

1 Theoretical Justification

1.1 Part a)

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = f, \quad x \in [0,1], t \in [0,1] \tag{1}$$

$$(0,t) = u(1,t) = 0, \quad t \in [0,1] \tag{2}$$

$$u(x,0) = u_0(x), \quad 0 \leq x \leq 1 \tag{3}$$

1.1.1 Finding the Variational Form

Multiplying (1) by a test function

$$v \in V$$

and integrating both sides over the domain we get

$$\int_\Omega u_t v dx - \int_\Omega u'' v dx = \int_\Omega f v dx \tag{4}$$

Then integrating by parts

$$\int_\Omega u_t v dx - (u'v|_{\partial\Omega} - \int_\Omega u'v' dx) = \int_\Omega f v dx \tag{5}$$

With the given boundary conditions (2), we know that the test functions vanish on the boundaries, so (5) reduces to our variational form.

$$\int_\Omega u_t v dx + \int_\Omega u'v' dx = \int_\Omega f v dx, v \in V, t \in \mathrm{T} \tag{6}$$

1.1.2 Finite Element Method

The variational form (6) can be reduced using inner products

$$(\dot{u}, v) + (u', v') = (f, v) \tag{7}$$

$$(\dot{u_h}, v) + (u_h', v') = (f, v) \tag{8}$$

And by expanding u and our test function into their hat function and nodes we get

$$\sum_{j}^{N} \dot{\xi}_j(t)(\varphi_j, \varphi_i) + \sum_{j}^{N} \xi_j(t)(\varphi_j', \varphi_i') = (f, \varphi_i), i = 1, 2, 3...n \tag{9}$$

Where n is the number of the nodes contained in the mesh. From this we derive our stiffness and mass matrices and the load vector,

$$A_{ij} = \int_\Omega \varphi_i' \varphi_j' dx$$

$$M_{ij} = \int_\Omega \varphi_i \varphi_j dx$$

$$b_i = \int_\Omega f(x,t)\varphi_i dx$$

Thus we can write (8) as

$$\dot{\xi}_j(t) M_{ij} + \xi_j(t) A_{ij} = b_i \tag{10}$$

which is the starting point for the algorithm outlined in the next subsection.

1.2 Part b)

$$\dot{u} - \Delta u = f, (x, y) \in [0,1]x[0,1], t \in [0,1] \tag{11}$$

$$u(x, y, t) = 0, (x, y) \in \partial\Omega, t \in [0,1], \tag{12}$$

$$u(x, y, 0) = u_0(x, y), 0 \leq x \leq 1, 0 \leq y \leq 1 \tag{13}$$

1.2.1 Finding the Variational Form

Multiplying (11) by test function v and integrating over the domain:

$$\int_\Omega \dot{u} v dx - \int_\Omega \Delta u v dx = \int_\Omega f v dx$$

Then we can apply Green's Formula to the second term

$$\int_\Omega \dot{u} v dx + (\int_\Omega \nabla u \cdot \nabla v dx - \int_{\partial\Omega} v \frac{\partial u}{\partial n} dx) = \int_\Omega f v dx$$

From (12), we know that u vanishes on its boundary, thus the test function will too. This means the third term is always zero, leaving us with:

$$\int_\Omega \dot{u} v dx + \int_\Omega \nabla u \cdot \nabla v dx = \int_\Omega f v dx \tag{14}$$

Leaving us with a variational form.

1.2.2 Finite Element Method

The variational form can be condensed with inner products.

$$(\dot{u_h}, v) + a(\nabla u_h, \nabla v) = (f, v)$$

By breaking u and the test functions into their components

$$u_h = \sum_i^N \xi_i(t)\varphi_i$$

a final inner product form can be obtained.

$$\sum_j^N \xi_j(\varphi_j, \varphi_i) + \sum_j^N \xi_j a(\nabla\varphi_j, \nabla\varphi_i) = (f, \varphi_i), i = 1, 2, ...N \tag{15}$$

With the same approach as in Part a, we can build stiffness and mass matrices, and a load vector.

$$A_{ij} = \int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j dx$$

$$M_{ij} = \int_\Omega \varphi_i\varphi_j dx$$

$$b_i = \int_\Omega f\varphi_i dx$$

Resulting in the final matrix form:

$$M_{ij}\dot{\xi}(t) + A_{ij}\xi(t) = b_i(t) \tag{16}$$

1.3 Algorithm

1. (a) Part a) Make f(x,t) such that

$$u = sin(2\pi x)cos(2\pi t)$$

, which can be done easily by plugging u into (1).

$$(-2\pi sin(2\pi x)sin(2\pi t)) - (-4\pi^2 sin(2\pi x)cos(2\pi t)) = f$$

$$f = 2\pi sin(2\pi x)(2\pi cos(2\pi t) - sin(2\pi t))$$

(b) Part b) $u = sin(2\pi x)sin(\pi y)cos(3\pi t)$

$$(-3\pi sin(2\pi x)sin(\pi y)sin(3\pi t)) - (-4\pi^2 sin(2\pi x)sin(\pi y)cos(3\pi t) - \pi^2 sin(2\pi x)sin(\pi y)cos(3\pi t)) = f$$

$$(-3\pi sin(2\pi x)sin(\pi y)sin(3\pi t)) + 5\pi^2 sin(2\pi x)sin(\pi y)cos(3\pi t) = f$$

$$f = sin(2\pi x)sin(\pi y)(5\pi^2 cos(3\pi t) - 3\pi sin(3\pi t))$$

2. From here populate the A and M matrices

3. Then, loop over all the nodes in time to be tested and calculate the load vector (f is dependant on time, so it must be in the loop).

4. Using the load vector, stiffness matrix, and mass matrix, one can solve for ξ using a numerical differential equations method.

   (a) Forward Euler's Method:

$$M\frac{(\xi_{l+1}-\xi_l)}{\Delta t_l}+A\xi_l=b_l$$

$$M(\xi_{l+1}-\xi_l)+\Delta t_l A\xi_l=\Delta t_l b_l$$

$$M\xi_{l+1}=M\xi_l-\Delta t_l A\xi_l+\Delta t_l b_l$$

   In this form, ξ_{l+1} can be solved for using a linear system of equations solver.

   (b) Backwards Euler's Method:

$$M\frac{(\xi_l-\xi_{l-1})}{\Delta t_l}+A\xi_l=b_l$$

$$M\xi_l-M\xi_{l-1}+\Delta t_l A\xi_l=\Delta t_l b_l$$

$$(M+\Delta t_l A)\xi_l=M\xi_{l-1}+\Delta t_l b_l$$

   Using this form, ξ_l can now be solved for using a linear system of equations solver because b_l is solved for separately. This has substantial benefits over the Forward Euler's method, as it has guaranteed stability, which is not the case with the case for Forward Euler.

   (c) Crank-Nicolson Method:

$$M\frac{(\xi_l-\xi_{l-1})}{\Delta t_l}+A\frac{(\xi_l+\xi_{l-1})}{2}=\frac{b_l+b_{l-1}}{2}$$

$$2M(\xi_l-\xi_{l-1})+\Delta t_l A(\xi_l+\xi_{l-1})=\Delta t_l(b_l+b_{l-1})$$

$$2M\xi_l+\Delta t_l A\xi_l=2M\xi_{l-1}-\Delta t_l A\xi_{l-1}+\Delta t_l(b_l+b_{l-1})$$

   Again, ξ_l can now be solved for using a linear system of equations solver. This method is also unconditionally stable, and is a good alternative to the backwards Euler method.

Table 1: Forward Euler

h	**L2(L2)**	**L2(H1)**
1/4	83.482460	584.461735
1/8	2.960e29	4.984e29
1/16	NaN	NaN
1/32	NaN	NaN
1/64	NaN	NaN
1/128	NaN	NaN

Table 2: Backwards Euler

h	**L2(L2)**	**L2(H1)**
1/4	0.494492	2.033511
1/8	0.306328	1.223929
1/16	0.171556	0.636425
1/32	0.091865	0.321490
1/64	0.047523	0.161338
1/128	0.024156	0.080798

Table 3: Crank-Nicolson

h	**L2(L2)**	**L2(H1)**
1/4	0.501844	2.070603
1/8	0.307411	1.225179
1/16	0.171674	0.636603
1/32	0.091878	0.321516
1/64	0.047524	0.161342
1/128	0.024156	0.080799

2 Results

2.1 Part 1 Convergence Data

Data for this part is on tables 1-3 (above).

Unfortunately, I was unable to make Forward Euler converge. I set Δt to $\frac{1}{2}h^2$ for both of these, because I was under the impression that if $\Delta t \leq h^2$ then it may conditionally converge, but I was not able to make that happen. Both $L2(L2)$ and $L2(H1)$ converge with h, which should only be the case for L2(H1). Again this puzzles me greatly, but of the Δt's which I tried and numerous numerical integration methods I tried, none gave me this result.

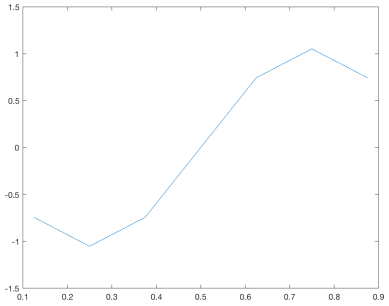
2.2 Part 2 Convergence Data

Data for this analysis is shown on Tables 4-7, sorry, I don't know how to really move them around yet.

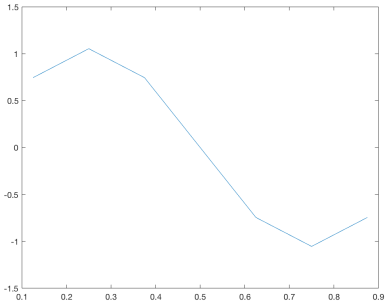
Again, Forward Euler diverged, and again, in the results seen above, I chose $\Delta t = \frac{1}{2}h^2$ in hopes that this would be frequent enough for the Forward Euler Method to produce a converging result. This test also saw similarly mixed results. Though the L2(L2) error for both Crank-Nicolson and Backward Euler continue to diminish with h^2, as they are supposed to, H1 also converges with h^2, which isn't right.

I believe this may be an issue with the Δt which I selected. I also included a 7th table with has the convergence data for Backward Euler for Part2, but instead of selecting $\Delta t = \frac{1}{2}h^2$, I choose that $\Delta t = h$. I'm sorry this is so inconclusive. I have trouble understanding why this would be the case, so I'll investigate further on my own time.

3 Images

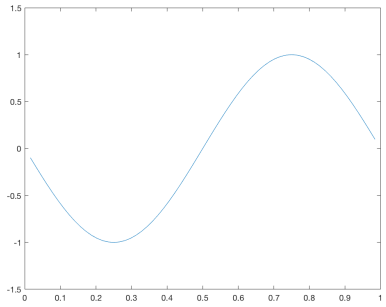


(a) $h = \frac{1}{8}$ and $t = 0.5$

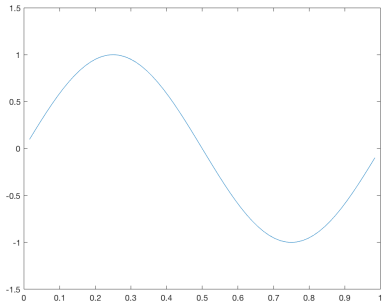


(b) $h = \frac{1}{8}$ and $t = 1$

Figure 1: Part 1: $h = \frac{1}{8}$

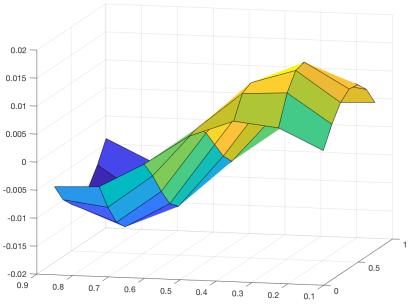


(a) $h = \frac{1}{64}$ and $t = 0.5$

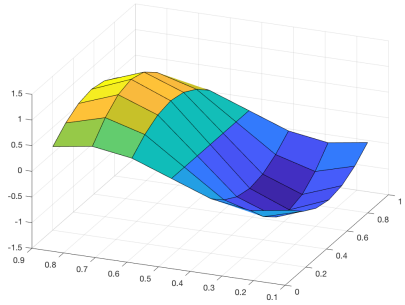


(b) $h = \frac{1}{64}$ and $t = 1$

Figure 2: Part 1: $h = \frac{1}{64}$

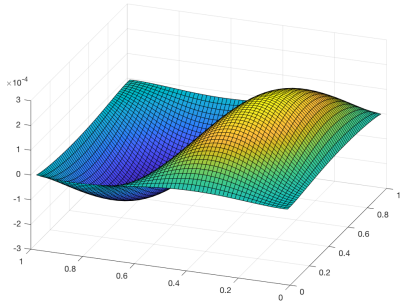


(a) $h = \frac{1}{8}$ and $t = 0.5$

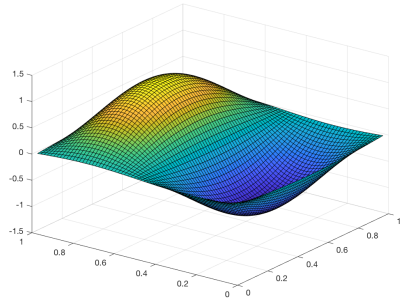


(b) $h = \frac{1}{8}$ and $t = 1$

Figure 3: Part 2: $h = \frac{1}{8}$



(a) $h = \frac{1}{64}$ and $t = 0.5$



(b) $h = \frac{1}{64}$ and $t = 1$

Figure 4: Part 2: $h = \frac{1}{64}$

Table 4: Forward Euler

h	**L2(L2)**	**L2(H1)**
1/4	1.1792e17	1.5855e18
1/8	1.1512e29	3.2606e29
1/16	NaN	NaN
1/32	NaN	NaN
1/64	NaN	NaN

Table 5: Backwards Euler

h	**L2(L2)**	**L2(H1)**
1/4	0.046637	0.358012
1/8	0.016821	0.133507
1/16	0.004362	0.032125
1/32	0.001092	0.007480
1/64	0.000272	0.001782

Table 6: Crank-Nicolson

h	**L2(L2)**	**L2(H1)**
1/4	0.043009	0.376479
1/8	0.015371	0.131267
1/16	0.004021	0.031461
1/32	0.001011	0.007345
1/64	0.000253	0.001754

Table 7: Backward Euler where $\Delta t = h$

h	**L2(L2)**	**L2(H1)**
1/4	0.059931	0.382429
1/8	0.043311	0.239447
1/16	0.020232	0.120205
1/32	0.009803	0.062815
1/64	0.004817	0.032400

Table 8: Backward Euler where $\Delta t = h$

h	**L2(L2)**	**L2(H1)**
1/4	0.059931	0.382429
1/8	0.043311	0.239447
1/16	0.020232	0.120205
1/32	0.009803	0.062815
1/64	0.004817	0.032400