Project 3: Time dependent Heat Equation

Wataru Hashimoto

4/25/2019

MEE 615

1. **Strong form and Inputs**
2. **Weak form derivation**
3. Convert to Einstein Notation
4. Multiply by weight function
5. Take Integration over domain
6. Integration by parts formula
7. Substitute to equation
8. Divergence theorem formula
9. Apply Divergence theorem and split the boundary to essential and natural
10. Weight function at essential boundary condition goes to 0

1. Using dot product property, using w as a scalar we can rewrite as
2. Substitute Neumann Boundary Condition eq. (3) and algebra
3. Substitute eq. (4) to
4. Use simplified notation
5. Weak form
6. **Discretization**
7. Discretized form
8. Substitute
9. Semi-discrete form (time is still continuous)

1. **Galerkin’s Approximation**
2. Shape functions
3. Substitute
4. Simplify
5. Matrix Formula
6. **Solving for spatially and temporary varying heat source q(x,t)**
7. Finding the q(x,t) for general case, start from governing equation
8. Exact solution is
9. Take partial differential with respect to time
10. Gradient formula
11. Plug in to gradient formula
12. Found gradient of exact solution
13. Perform matrix multiplication with tensor, (2x2) (2x1) = (2x)
14. Divergent formula
15. Plug in to the divergent
16. Found the heat source for general case
17. Finding the q(x,t=0) for initial condition, for this case, we won’t use p, but rather, we use which is
18. Notice that the term become zero because is independent of the time, hence
19. **Boundary Condition Derivation**
20. Drichlet Boundary Condition
21. Neumann Boundary Condition

Normal Vector visualization

1. **Global Coordinate Matrix**
2. This matrix stores the information of the (x,y) coordinates for the each nodes, left to right and bottom to top
3. For example, let’s say we have 2 by 2 elements for total of 4 elements for domain from [0,1], then the coordinates will be

(1,1)

(0,1)

(0.5,1)

(0.5,1)

(0.5,0.5)

(1,0.5)

(1,0)

(0,0)

(0.5,0)

1. **Elemental Node matrix**
2. This matrix stores the information about the location of the nodes for each element, the node for each element is counted starting from bottom left corner and proceed counter clock wise.
3. For example, quadrilateral element with 4 local nodes will be.

4

3

2

1

1. Notice if you would make an elemental node matrix then it will be something like this for 2 by 2 element

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Number of element | | | |
| Local Nodes | 1 | 2 | 4 | 5 |
| 2 | 3 | 5 | 6 |
| 5 | 6 | 8 | 9 |
| 4 | 5 | 7 | 8 |

1. Which can be treated as this table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Number of element | | | | |
| Local Nodes | 1 | 2 | 3 | 4 | 5 |
| 2 | 3 | 4 | 5 | 6 |
| 5 | 6 | 7 | 8 | 9 |
| 4 | 5 | 6 | 7 | 8 |

1. And remove each number of element+1 column, and you will get elemental node matrix

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Number of element | | | | |
| Local Nodes | 1 | 2 | 3 | 4 | 5 |
| 2 | 3 | 4 | 5 | 6 |
| 5 | 6 | 7 | 8 | 9 |
| 4 | 5 | 6 | 7 | 8 |

1. **Boundary Matrix**
2. This matrix is to store the information about which nodes we should apply boundary conditions
3. **Assembly Theory**
4. Regular Integration form for local matrix
5. Forcing term components

Neumann Boundary Condition =

Dirichlet Boundary Condition =

1. Finding the initial d (or T) and v
2. Finding the final temperature after time loop
3. **Results - Convergence Table**
4. Backward Euler (alpha = 1)
   1. Spatial convergence

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Cycle | Time step | Mesh size | # cells | # dofs | L2 error | convergence |
| 1 | 5e-05 | 8 | 64 | 81 | 0.0049801 | 1.996 |
| 2 | 5e-05 | 16 | 256 | 289 | 0.001287 | 1.996 |
| 3 | 5e-05 | 32 | 1024 | 1089 | 0.00032342 | 1.996 |
| 4 | 5e-05 | 64 | 4096 | 4225 | 7.8427e-05 | 1.996 |

* 1. Temporal convergence

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Cycle | Time step | Mesh size | # cells | # dofs | L2 error | convergence |
| 1 | 0.1 | 64 | 4096 | 4225 | 0.012666 | 1.0008 |
| 2 | 0.05 | 64 | 4096 | 4225 | 0.006401 | 1.0008 |
| 3 | 0.025 | 64 | 4096 | 4225 | 0.0031989 | 1.0008 |
| 4 | 0.0125 | 64 | 4096 | 4225 | 0.0015805 | 1.0008 |

1. Crank-Nicolson (alpha = 0.5)
   1. Spatial convergence

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Cycle | Time step | Mesh size | # cells | # dofs | L2 error | convergence |
| 1 | 5e-05 | 8 | 64 | 81 | 0.0049839 | 1.9750 |
| 2 | 5e-05 | 16 | 256 | 289 | 0.0012908 | 1.9750 |
| 3 | 5e-05 | 32 | 1024 | 1089 | 0.00032722 | 1.9750 |
| 4 | 5e-05 | 64 | 4096 | 4225 | 8.2118e-05 | 1.9750 |

* 1. Temporal convergence

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Cycle | Time step | Mesh size | # cells | # dofs | L2 error | convergence |
| 1 | 0.1 | 64 | 4096 | 4225 | 0.00089513 | 1.09 |
| 2 | 0.05 | 64 | 4096 | 4225 | 0.00041947 | 1.09 |
| 3 | 0.025 | 64 | 4096 | 4225 | 0.00019213 | 1.09 |
| 4 | 0.0125 | 64 | 4096 | 4225 | 9.1477e-05 | 1.09 |

1. Forward Euler (alpha = 0)
   1. Spatial convergence

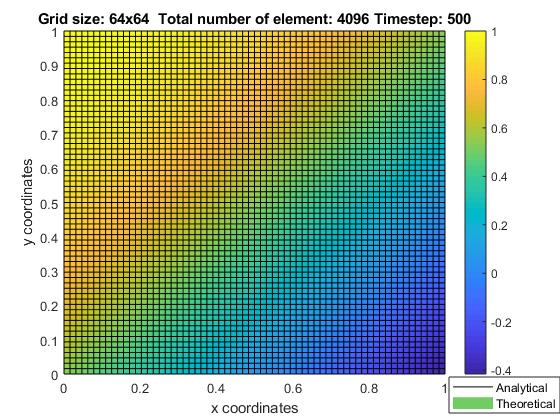
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Cycle | Time step | Mesh size | # cells | # dofs | L2 error | convergence |
| 1 | 5e-05 | 8 | 64 | 81 | 0.0049876 | NaN |
| 2 | 5e-05 | 16 | 256 | 289 | 0.0012946 | NaN |
| 3 | 5e-05 | 32 | 1024 | 1089 | 0.00033109 | NaN |
| 4 | 5e-05 | 64 | 4096 | 4225 | NaN | NaN |

* 1. Temporal convergence

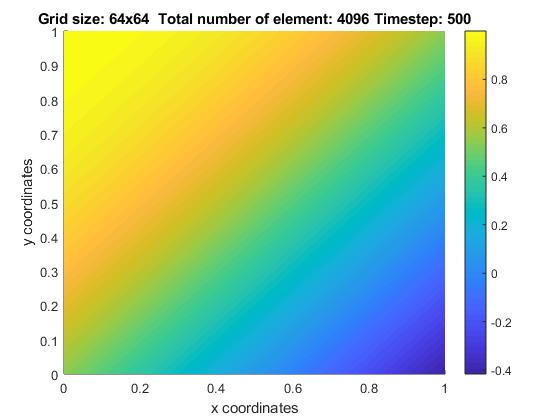
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Cycle | Time step | Mesh size | # cells | # dofs | L2 error | convergence |
| 1 | 0.1 | 64 | 4096 | 4225 | 1.8683e30 | -Inf |
| 2 | 0.05 | 64 | 4096 | 4225 | 2.4648e63 | -Inf |
| 3 | 0.025 | 64 | 4096 | 4225 | 2.3655e124 | -Inf |
| 4 | 0.0125 | 64 | 4096 | 4225 | Inf | -Inf |

1. **Results – Solution plots for benchmark**
2. 2D Plot for 64 by 64 by 500

With mesh

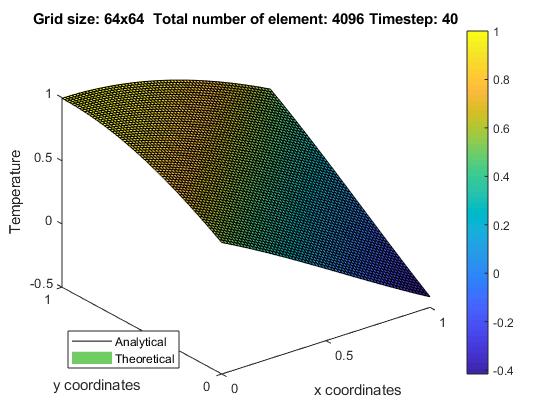
****

Without mesh

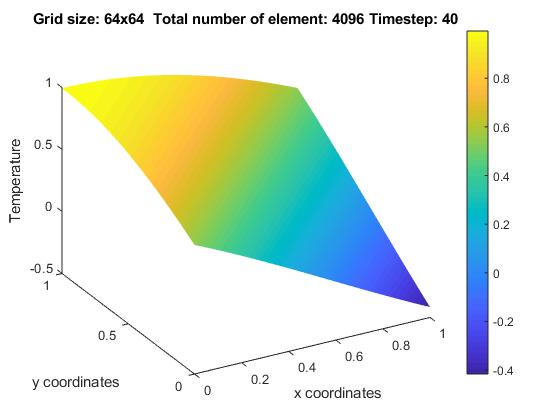
****

1. 3D Plot for 64 by 64 by 40

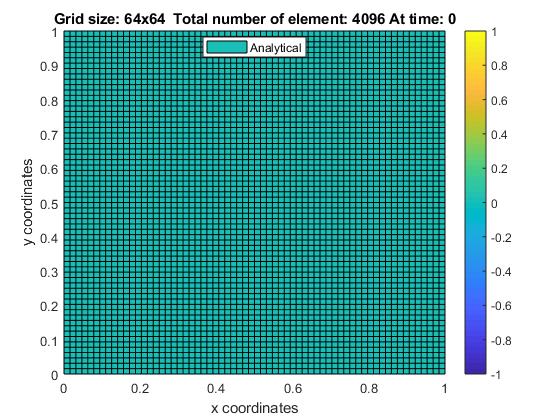
With mesh



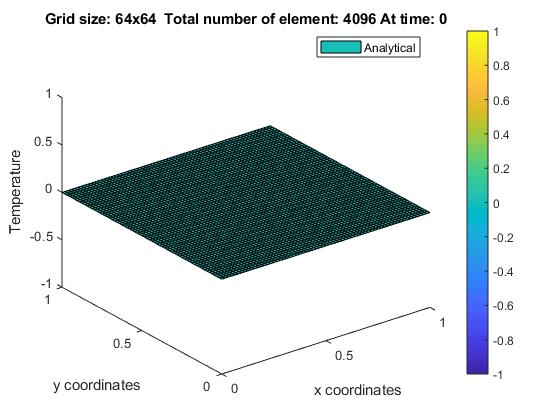
Without mesh

****

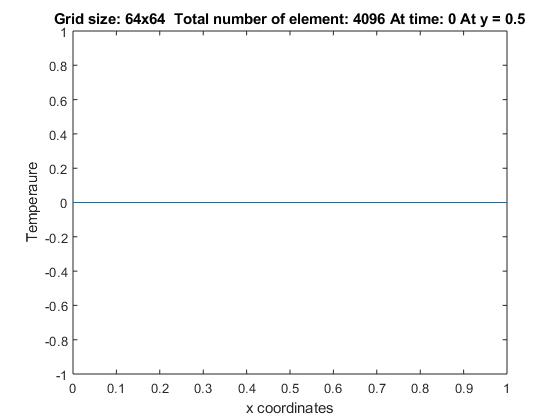
1. **Results – Solution plots for application**
2. Initial (t = 0)
   1. 2D plots



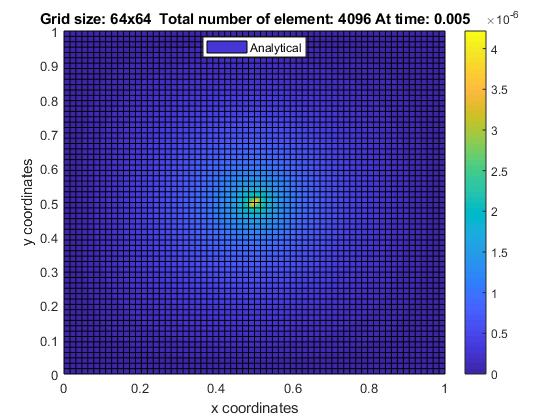
* 1. 3D plots



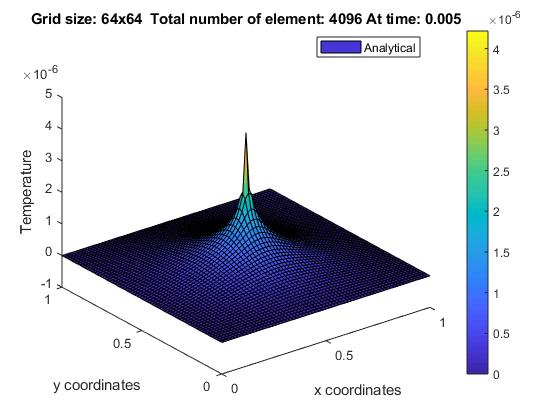
* 1. Profile graph along y = 0.5



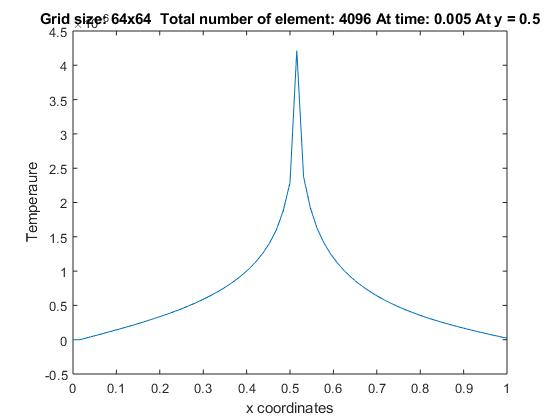
1. Early time (t = 0.005)
   1. 2D plots



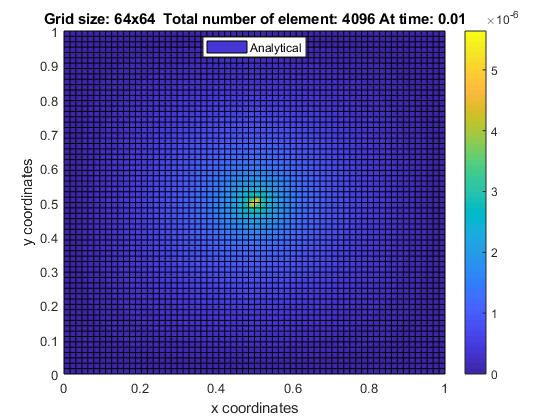
* 1. 3D plots



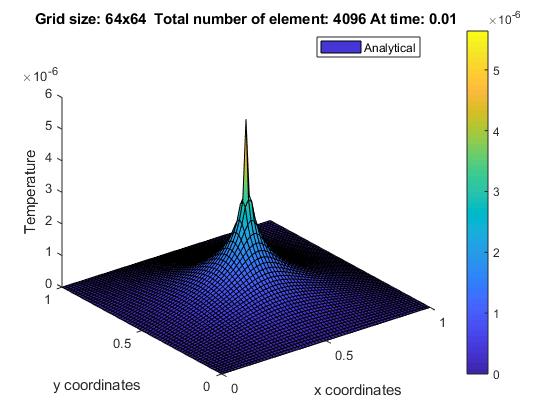
* 1. Profile graph along y = 0.5



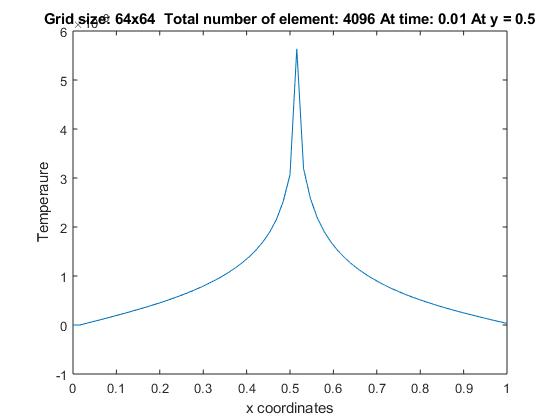
1. Intermediate time (t= 0.01)
   1. 2D plots



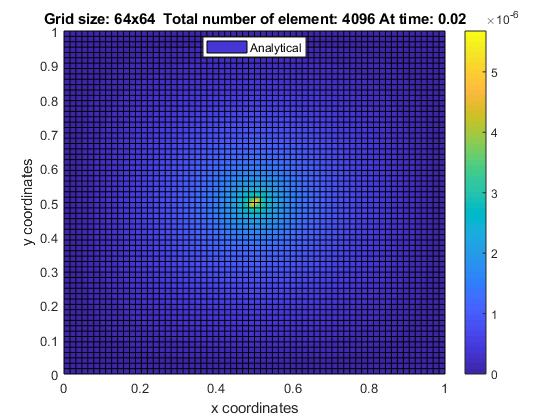
* 1. 3D plots



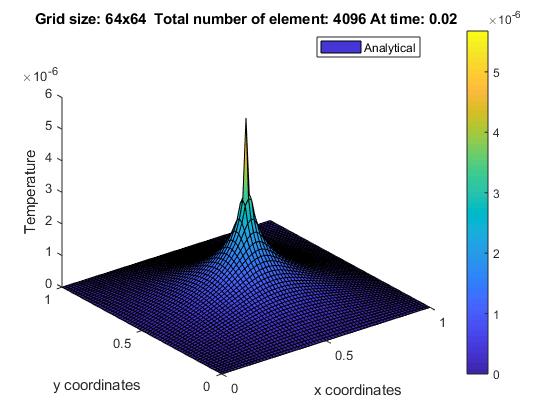
* 1. Profile graph along y = 0.5



1. Late time (t = 0.02)
   1. 2D plots



* 1. 3D plots



* 1. Profile graph along y = 0.5

