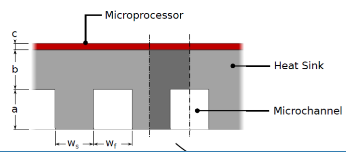
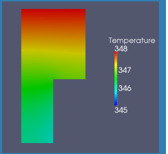
Unstructured FEM:
2D Unsteady Heat Diffusion





Abhishek Y. Deshmukh, Raghavan Lakshmanan,
Mohsin Ali Chaudry, A. Emre Ongut

January 31, 2014

German Research School
for Simulation Sciences
JÜLICH FORSCHUNGSZENTRUM
RWTH AACHEN UNIVERSITY

Contents

Objective

- To develop a parallel finite element solver that uses unstructured grid to solve a 2D transient heat diffusion problem.
- Application to a realistic engineering heat transfer problem, Heat diffusion in Microchannels in the heat sink of microprocessors.

German Research School for Simulation Sciences

Contents

General Steps

1. Preprocessing
   - Define a domain of interest
   - Discretization of domain into finite elements
2. Solution
   - Define shape functions to represent the behaviour of elements
   - Construct local stiffness matrix and then ultimately global stiffness matrix
   - Apply boundary condition, initial conditions and forcing function
   - Solving linear system of algebraic equations
3. Postprocessing
   - Analyse the solution

Contents

2D Unsteady Heat Diffusion Equation

$$\frac{\partial T}{\partial t} - \alpha \nabla^2 T = f, \text{in } \Omega \in \Re^2$$

where α is diffusion coefficient.

$$\alpha = \frac{k}{\rho c_p}$$

k is heat conductivity, ρ is the density and c_p is the specific heat capacity of the material.

$$f = \frac{\dot{q}}{\rho c_p}$$

where $\dot{q}$ is heat generation per unit volume.

Initial and Boundary Conditions

- Intial condition: $T(x, y, 0) = T_0(x, y)$
- Boundary conditions
  - Dirichlet:
    $T(x, 0, t) = T_1,\ T(x, L, t) = T_2,\ T(0, y, t) = T_3,$
    $T(L, y, t) = T_4$
  - Neumann:
    $k\left(n_x \frac{\partial T}{\partial x} + n_y \frac{\partial T}{\partial y}\right) = q_0$
  - Robin (Mixed):
    $h(T - T_{\text{inf}}) = -k\left(n_x \frac{\partial T}{\partial x} + n_y \frac{\partial T}{\partial y}\right)$

Weighted Residual Method

$$\int_\Omega \left(w\frac{\partial T}{\partial t} + \alpha w \nabla^2 T\right) d\Omega = \int_\Omega wf d\Omega$$

where w is a weigthing function or test function.
Integrating by parts,

$$\int_\Omega \left(w\frac{\partial T}{\partial t} + \alpha \nabla w . \nabla T\right) d\Omega = \int_\Omega wf d\Omega + \frac{1}{\rho c_p} \int_\Gamma w \underbrace{(k\hat{n}.\nabla T)}_{k(n_x \frac{\partial T}{\partial x} + n_y \frac{\partial T}{\partial y})} d\Gamma$$

Approximation

- Approximate the temperature as a piecewise linear combination of shape functions $S_j(x, y)$.
- These shape functions remain constant over time. The coefficients $T_j(t)$ are time dependent.

$$T(x, y, t) = \sum_{j=1}^{nn} T_j(t) S_j(x, y)$$

- Galerkin Finite Element Method:

$$w = S_i(x)$$

Approximation [contd.]

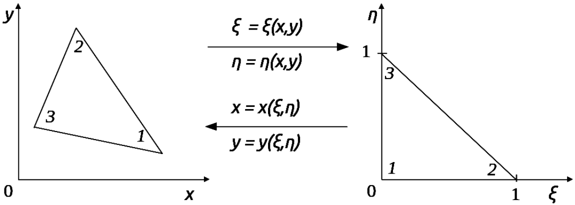
$$\underbrace{\sum_{j=1}^{nn}\int_{\Omega}(S_iS_j)d\Omega\frac{\partial T_j}{\partial t}}_{[M][\dot{T}]}-\underbrace{\sum_{j=1}^{nn}[\int_{\Omega}\alpha(\frac{\partial S_i}{\partial x}\frac{\partial S_j}{\partial x}+\frac{\partial S_i}{\partial y}\frac{\partial S_j}{\partial y})d\Omega]T_j}_{[K][T]}$$

$$=\underbrace{\int_{\Omega}S_ifd\Omega}_{[F]}+\underbrace{\frac{1}{\rho c_p}\int_{\Gamma}S_i(k\hat{n}.\nabla T)d\Gamma}_{[B]}$$

$i,j=1,2,...,nn$

$$[K]_{ij}=\int_{\Omega}\alpha(\frac{\partial S_i}{\partial x}\frac{\partial S_j}{\partial x}+\frac{\partial S_i}{\partial y}\frac{\partial S_j}{\partial y})d\Omega$$

Triangular Elements



Co-ordinate transformation

Shape functions:

$S_1(\xi,\eta) = 1 - \xi - \eta$, $S_2(\xi,\eta) = \xi$, $S_3(\xi,\eta) = \eta$

$$\frac{\partial S_1}{\partial \xi} = -1, \frac{\partial S_2}{\partial \xi} = 1, \frac{\partial S_3}{\partial \xi} = 0, \frac{\partial S_1}{\partial \eta} = -1, \frac{\partial S_2}{\partial \eta} = 0, \frac{\partial S_3}{\partial \eta} = 1$$

$$\frac{\partial S_i}{\partial \xi} = \frac{\partial S_i}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial S_i}{\partial y}\frac{\partial y}{\partial \xi}, \frac{\partial S_i}{\partial \eta} = \frac{\partial S_i}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial S_i}{\partial y}\frac{\partial y}{\partial \eta}$$

$$\begin{bmatrix} \frac{\partial S_i}{\partial \xi} \\ \\ \frac{\partial S_i}{\partial \eta} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix}}_{Jacobian} \begin{bmatrix} \frac{\partial S_i}{\partial x} \\ \\ \frac{\partial S_i}{\partial y} \end{bmatrix}$$

Isoparametric formulation:

$x(\xi,\eta) = \sum_{j=1}^{nen} x_j^e S_j(\xi,\eta)$, $y(\xi,\eta) = \sum_{j=1}^{nen} y_j^e S_j(\xi,\eta)$

Jacobian:

$$[J] = \begin{bmatrix} \frac{\partial S_1}{\partial \xi} & \frac{\partial S_2}{\partial \xi} & \frac{\partial S_3}{\partial \xi} \\ \frac{\partial S_1}{\partial \eta} & \frac{\partial S_2}{\partial \eta} & \frac{\partial S_3}{\partial \eta} \end{bmatrix} \begin{bmatrix} x_1^e & y_1^e \\ x_2^e & y_2^e \\ x_3^e & y_3^e \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix}$$

But we need $[J]^{-1}$ which is the inverse mapping.

$$\begin{bmatrix} \frac{\partial S_i}{\partial x} \\ \frac{\partial S_i}{\partial y} \end{bmatrix} = \underbrace{\begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} \end{bmatrix}}_{[J]^{-1}} \underbrace{\begin{bmatrix} \frac{\partial S_i}{\partial \xi} \\ \frac{\partial S_i}{\partial \eta} \end{bmatrix}}_{known}$$

Calculation of element level stiffness matix:

$$K_{ij}^{e} = \int_{\Omega_e} \alpha\left(\frac{\partial S_i}{\partial x}\frac{\partial S_j}{\partial x} + \frac{\partial S_i}{\partial y}\frac{\partial S_j}{\partial y}\right) dxdy \text{ i,j = 1,2,...,nen}$$

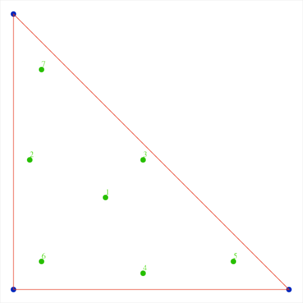
Integral transformation to master element:

$$K_{ij}^{e} = \int_0^1 \int_0^{1-\xi} \alpha[f_k(\xi,\eta)]|J|d\xi d\eta$$

$$f_k(\xi,\eta) = \left(\frac{\partial S_i}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial S_i}{\partial \eta}\frac{\partial \eta}{\partial x}\right)\left(\frac{\partial S_j}{\partial \xi}\frac{\partial \xi}{\partial x} + \frac{\partial S_j}{\partial \eta}\frac{\partial \eta}{\partial x}\right) + \left(\frac{\partial S_i}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial S_i}{\partial \eta}\frac{\partial \eta}{\partial y}\right)\left(\frac{\partial S_j}{\partial \xi}\frac{\partial \xi}{\partial y} + \frac{\partial S_j}{\partial \eta}\frac{\partial \eta}{\partial y}\right)$$

$i, j = 1, 2, ..., nen$

Numerical Integration using 7 point Gauss Quadrature rule:



$$K_{ij}^{e} \simeq \sum_{l=1}^{nGQP} f_k(\xi_l, \eta_l) w_l$$

Solution

In matrix form : $[M][\dot{T}] + [K][T] = [F] + [B]$
Forward Euler explicit differencing in time:

$$\frac{dT}{dt}\Big|_s = \frac{T^{s+1} - T^s}{\Delta t} + O(\Delta t)$$

$$[M]\frac{[T]^{s+1} - [T]^s}{\Delta t} + [K][T]^s = [F] + [B]$$

$$[M][T]^{s+1} = [M][T]^s + \Delta t([F] + [B] - [K][T]^s)$$

$$[M][T]^{s+1} = [RHS]^s$$

Lumping of Mass Matrix

The inversion of Mass matrix can be avoided if we do a physical approximation called lumping of mass matrix.

Consistent element mass matrix: $[M_{cij}]$

Lumped element mass matrix: $[M_{Lij}]$

$$M_{Lii} = M_{cii} \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} M_{cij}}{\sum_{j=1}^{n} M_{cij}}$$

This is correct only for linear elements.

$[M_{Lij}]$ is a diagonal matrix.

Boundary Integral

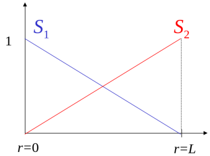
- Dirichlet: $[B]_j = T_w$, $[K]_{jj} = 1$, $[K]_{ij} = 0 \; \forall i \neq j$
  For example, if for some element, node temperatures T1 and T3 are known then [B] and [K] for that element can be written as:

$$[B] = \begin{bmatrix} T_w \\ B_2 \\ T_w \end{bmatrix} \text{ and } [K] = \begin{bmatrix} 1 & 0 & 0 \\ * & * & * \\ 0 & 0 & 1 \end{bmatrix}$$

$*$ indicate non-zero values.

Boundary Integral [contd.]

- Neumann (Flux Variable(FV) = q_{in}):



$$B_1^e = \frac{1}{\rho c_p}\int_\Gamma S_1(FV)d\Gamma = \frac{1}{\rho c_p}\int_0^L (1-\frac{r}{L})q_{in}dr = \frac{q_{in}L}{2\rho c_p}$$

$$B_2^e = \frac{1}{\rho c_p}\int_\Gamma S_2(FV)d\Gamma = \frac{1}{\rho c_p}\int_0^L \frac{r}{L}q_{in}dr = \frac{q_{in}L}{2\rho c_p}$$

$$[B] = \begin{bmatrix} \frac{q_{in}L}{2\rho c_p} \\ \frac{q_{in}L}{2\rho c_p} \\ 0 \end{bmatrix}$$

Boundary Integral [contd.]

- Robin(Mixed type): Flux variable (FV) can be expressed as

$$-k\hat{n}.\nabla T = h(T - T_\infty)$$

$FV = aPV + b$ where a = -h and b = hT_∞

$$B_1^e = \frac{1}{\rho c_p}\int_\Gamma S_1(FV)d\Gamma = \frac{1}{\rho c_p}\int_0^L (1 - \frac{r}{L})(aPV + b)dr$$

$$B_2^e = \frac{1}{\rho c_p}\int_\Gamma S_2(FV)d\Gamma = \frac{1}{\rho c_p}\int_0^L \frac{r}{L}(aPV + b)dr$$

Primary variable (PV) temperature:

$$PV = S_1T_1 + S_2T_2$$

Boundary Integral [contd.]

$$B_1^e = \frac{1}{\rho c_p} \int_0^L (1 - \frac{r}{L})[a((1 - \frac{r}{L})T_1 + \frac{r}{L}T_2) + b]dr$$

$$= \frac{L}{6\rho c_p}(3b + 2aT_1 + aT_2)$$

$$B_2^e = \frac{1}{\rho c_p} \int_0^L \frac{r}{L}[a((1 - \frac{r}{L})T_1 + \frac{r}{L}T_2) + b]dr$$

$$= \frac{L}{6\rho c_p}(3b + aT_1 + 2aT_2)$$

Boundary Integral [contd.]

Element boundary vector can be written as:

$$[B] = \begin{bmatrix} \frac{L}{6\rho c_p}(3b + 2aT_1 + aT_2) \\ \frac{L}{6\rho c_p}(3b + aT_1 + 2aT_2) \\ 0 \end{bmatrix}$$

Coefficients of T_1 and T_2 transferrd to K. Modified [B] and corresponding [K] can be written as:

$$[B] = \begin{bmatrix} \frac{Lb}{2\rho c_p} \\ \frac{Lb}{2\rho c_p} \\ 0 \end{bmatrix} \text{ and } [K] = \begin{bmatrix} (* - \frac{aL}{3\rho c_p}) & (* - \frac{aL}{6\rho c_p}) & * \\ (* - \frac{aL}{6\rho c_p}) & (* - \frac{aL}{3\rho c_p}) & * \\ * & * & * \end{bmatrix}$$

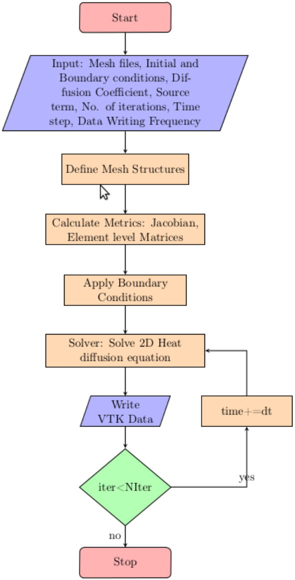
Contents

Program Flow Chart

1 Preprocessing Stage:
- Reading settings file
- Reading mesh data

2 Solution Stage:
- Calculating Jacobian matrix for all elements
- Calculating element stiffness matrix for all elements
- Applying boundary conditions
- Explicit Solver

3 Post Processing Stage



Contents

Case 1: 1D

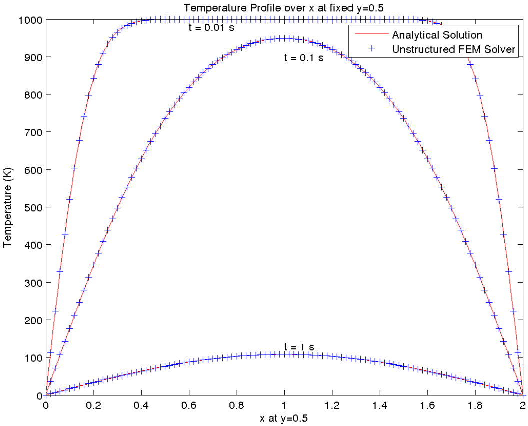
A simple 1D case is taken as an example. Domain: x $\in$ [0,L]
L = 2.
Initial condition: $T_i = T(x,0) = 1000K$
Boundary Conditions: $T_L = T(0,t)) = 0$, $T_R = T(L,t)) = 0$,
T_{top} and T_{bottom} are insulated
Analytical Solution:

$$T(x,t) = \sum_{n=1}^{\infty} B_n sin(\frac{n\pi x}{L}) e^{-\frac{n^2\pi^2\alpha t}{L^2}}$$

where α is diffusion coefficient and

$$B_n = -T_i \frac{2(-1+(-1)^n)}{n\pi}$$

Comparison with Analytical solution



Case 2: 2D Quenching of a Billet[1]

Domain: A long billet of rectangular cross section $2a \times 2b$ (a=2, b=1). Because of symmetry, analyze one quarter of the cross section
Shifted temperature: $\theta(x, y, t) = T(x, y, t) - T_\infty$
Material properties: $\alpha = 1$, $\rho = 1$, $c_p = 1$, $h = 1$
Initial condition: $\theta(x, y, 0) = T_i - T_\infty = \theta_i$, $T_i = 1000$, $T_\infty = 300$
Boundary Conditions:

$x = 0$: $\frac{\partial \theta}{\partial x} = 0$

$x = a$: $\frac{\partial \theta}{\partial x} = -\frac{h}{k}\theta(a, y, t)$

$y = 0$: $\frac{\partial \theta}{\partial y} = 0$

$y = b$: $\frac{\partial \theta}{\partial y} = -\frac{h}{k}\theta(x, b, t)$

Analytical solution:

$$\frac{\theta}{\theta_i} = 4\sum_{n=1}^{\infty}\sum_{m=1}^{\infty} e^{(-\alpha(\lambda_n^2+\beta_m^2)t)} \times$$

$$\frac{sin(\lambda_n a)cos(\lambda_n x)sin(\beta_m b)cos(\beta_m y)}{[\lambda_n a + sin(\lambda_n a)cos(\lambda_n a)][\beta_m b + sin(\beta_m b)cos(\beta_m b)]}$$

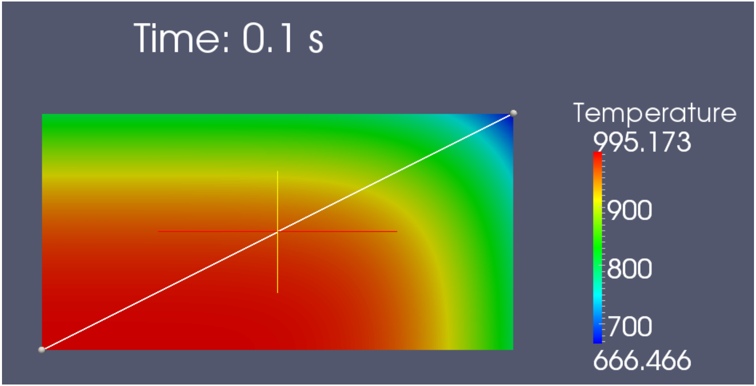
where α is diffusion coefficient and the eigenvalues λ_n and β_m are the roots of trancendental equations.

$$\lambda_n tan(\lambda_n a) = \frac{h}{k} \qquad \beta_m tan(\beta_m b) = \frac{h}{k}$$

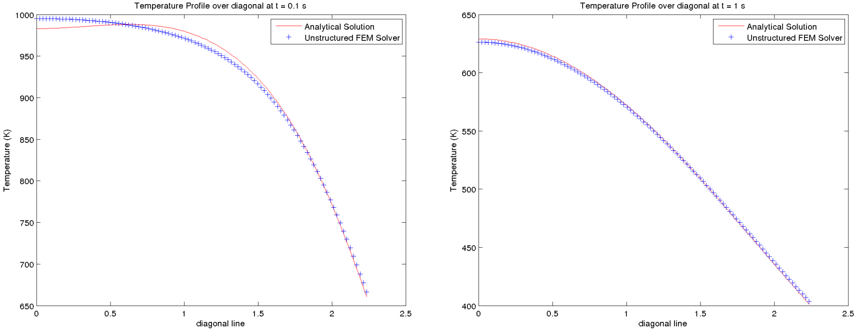
where k is conductivity of the billet matrial.

German Research School
for Simulation Sciences

Results



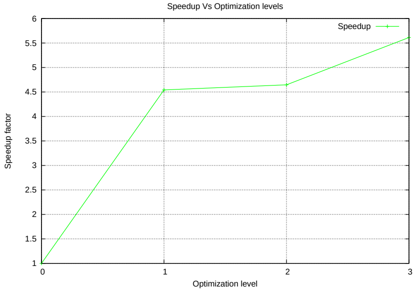
Results [contd.]



Contents

Optimization

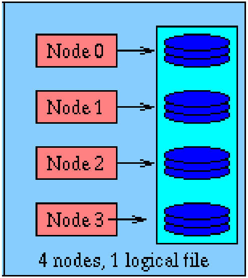
- Manual Optimization
  - Local arrays and variables
  - Loop unrolling
- Automatic Optimization
  - O1
  - O2
  - O3



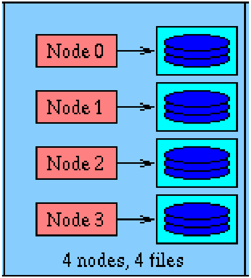
German Research School for Simulation Sciences

Contents

Parallelizing



Parallel I/O 1 [3]



Parallel I/O 2 [3]

Structure of Mesh files (Illustration)

mxyz

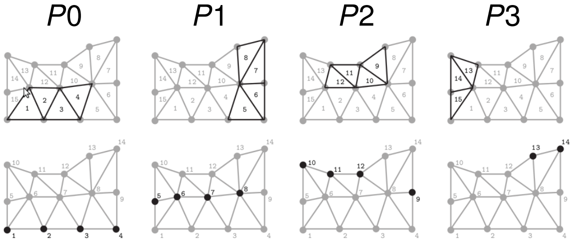
0	0
0.5	0
1	0
2	0
2	0.25
2	0.5
1.5	0.75
1	0.75
0.5	0.75
0	0.5
0	0.25
0.5	0.25

mien

8	11	10
9	2	8
1	4	7
3	5	8
5	11	10
4	7	9
6	8	3
10	3	9
7	1	2
5	4	8
11	9	7

First approach

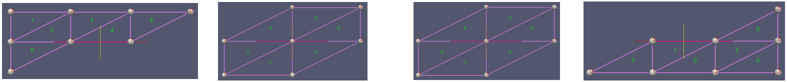
- All the node information not present on indvidual processors. Mesh data needs to be communicated before the actual FEM code starts. This adds to communication overhead.



Division of elements and nodes [4]

Second approach

- All mesh data present on the same processor.

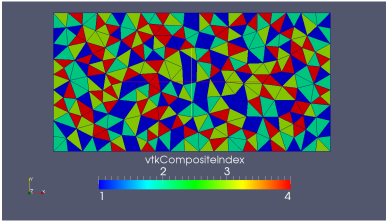


P0 **P1** **P2** **P3**

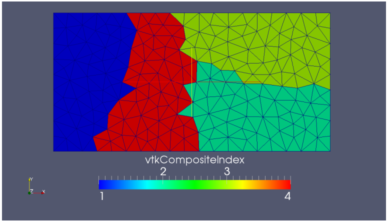
- A preprocessor is needed to generate the separate domains for each processor. Once the mesh is divided it can be reused in further simulations.

Mesh Partitioning

- Mesh partitioning tool needs to be used.



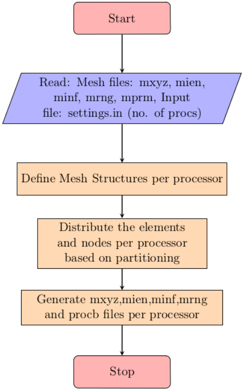
Before Mesh partitioning



After Mesh partitioning

Several public tools available, e.g. SCOTCH, CHACO, HARP, JOSTLE, METIS, etc. We used METIS.

Parallelizing with second approach



Preprocessor Flow Chart

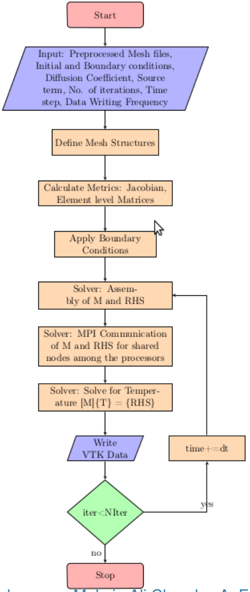
Contents of a sample procb file

0	0	-1	-1	-1	-1
1	0	-1	-1	-1	-1
2	0	-1	-1	-1	-1
3	1	1	-1	-1	-1
13	1	1	-1	-1	-1
14	1	1	-1	-1	-1
15	2	1	2	-1	-1
16	2	1	2	-1	-1

German Research School for Simulation Sciences

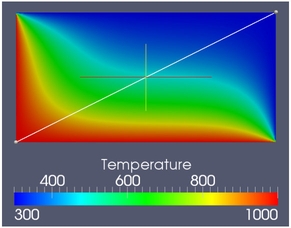
Program Flow Chart

1. Preprocessing Stage:
   - Reading settings and mesh data
2. Solution Stage:
   - Calculating mesh metrics
   - Applying boundary conditions
   - Assembly of [M] and {RHS}
   - Communicate [M] and {RHS} for boundary nodes
   - Solve [M]{T} = {RHS}
3. Post Processing Stage



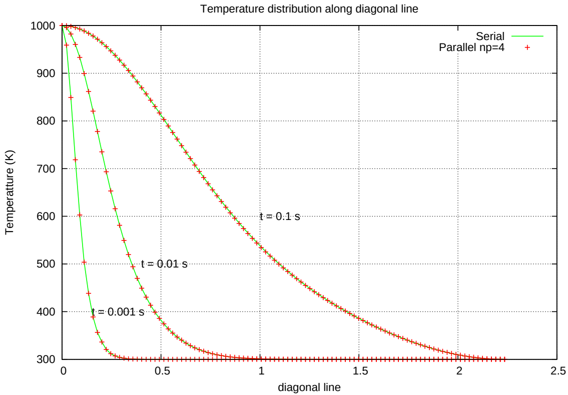
Verification of Parallel code

A rectangular domain is considered for heat diffusion problem. Left and bottom walls are kept at 1000 K while top and right walls are maintained at 300 K and the temperature profile is allowed to develop over time. The snapshots are taken at t=0.001s, t=0.01s and t=0.1s.

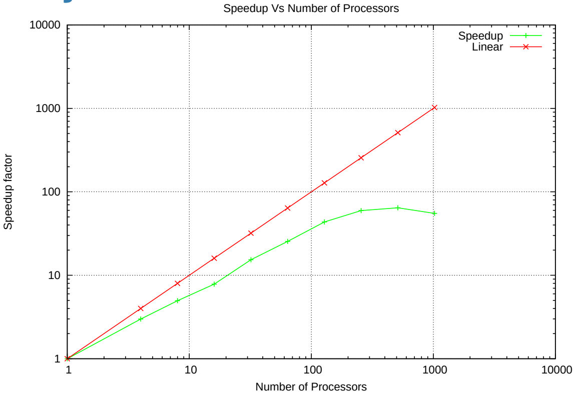


t = 0.1 s

Verification of Parallel code

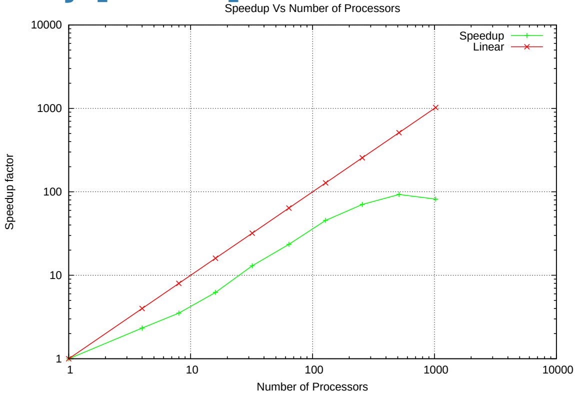


Scalability



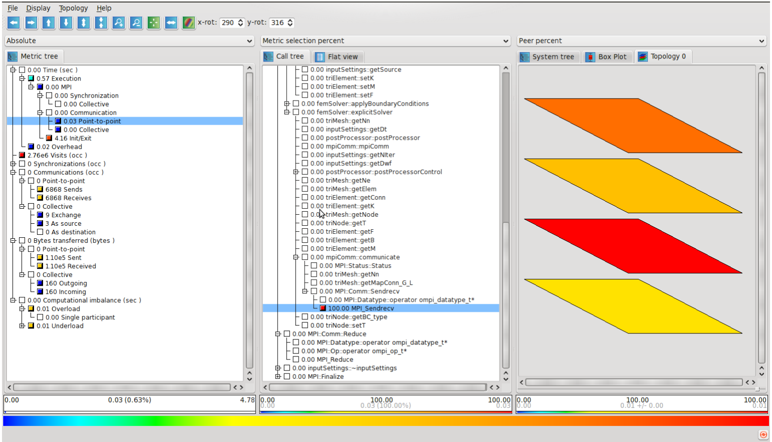
Finestmesh: ne = 516160, nn = 259082

Scalability [contd.]



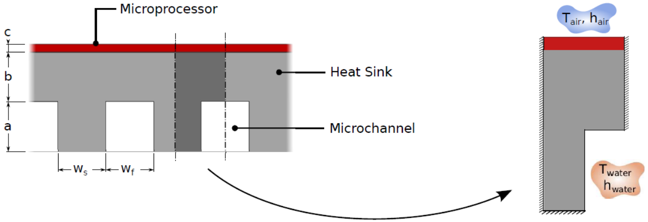
Superfine: ne = 945746, nn = 474074

Analysis using Scalasca



Contents

Application

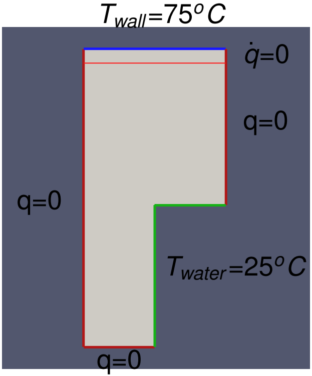


A heat sink for cooling computer chip, which is fabricated from copper with machined microchannels. Within these microchannels, water flows and carries away the heat dissipated by the chips.

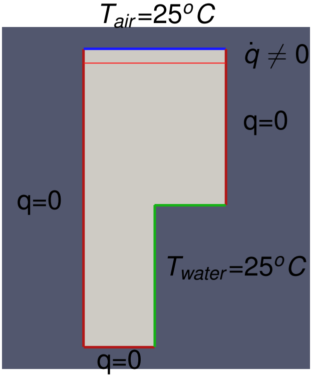
Physical properties and dimensions:

- Dimensions of the heat sink: a = b = w_s = w_f = 0.2 mm, c = 0.02 mm
- Thermal diffusivity of copper: α_{copper} = 1.11 $\times$ 10^{-4} m^2/s
- Density of copper: ρ = 8940 kg/m^3
- Specific heat capacity: C_p = 385 J/kg/K
- Properties of cooling fluid: T_{water} = 25^o C, h_{water} = 30000 $W/m^2/K$
- Properties of ambient air: T_{air} = 25^o C, h_{air} = 2 $W/m^2/K$

Case 1: Steady state temperature distribution

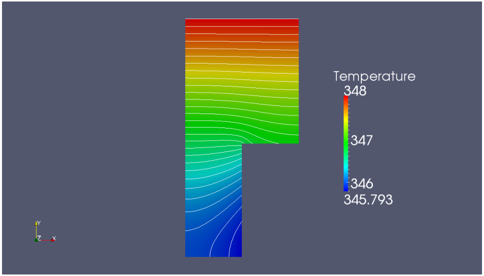


Case 2: Find $\dot{q}_{max}$ for $T_{max} < 75^0C$



Case 1

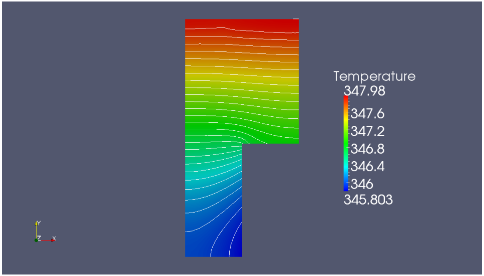
Steady state criteria: Maximum rate of change of temperature in the domain $\left(\frac{dT}{dt}\right)_{max} \leq 0.001$



Time required to reach steady state = **8.4237117 ms**

Case 2

Use the result of case 1 as the initial condition. Maximum allowable temperature in the domain 75^oC or 348 K



Maximum heat generation rate $\approx$ **9.6** $\times\ 10^{10}\ W/m^3$. If we consider a chip of dimensions 10 mm$\times$10 mm $\times$ 0.02 mm, it is allowed to generate maximum **192 W.**

Contents

Summary

- A two dimensional transient heat diffusion equation is solved numerically using finite element method on unstructured mesh.
- A serial object oriented C++ code is developed based on the FE formulation.
- The code is verified against analytical solutions found in [1].
- The code is parallelized to be able to solve larger problems faster.
- The parallel code shows sufficient amount of scalability.
- Finally, the capability of the software to be applied to a realistic problem has been demonstrated.

References

- Jordan Wall.
  Class materials for Conduction Heat Transfer.
  pages 5-8, 21.
- Class materials for Parallel Programming I.
- https://computing.llnl.gov/LCdocs/ioguide/.
- Class materials for Parallel Computing for Continuum Mechanics.

Questions?