Yixue Zhang, Zheran Li, Xiao Ling

EC 500 Term Project Report

Processor Heat Transfer Simulation

Boston University
Mechanical Engineering

1 Introduction

Heat transfer is a discipline of thermal engineering that concerns the generation, use, conversion, and exchange of thermal energy (heat) between physical systems. Heat transfer is classified into various mechanisms, such as thermal conduction, thermal convection, thermal radiation, and transfer of energy by phase changes. Engineers also consider the transfer of mass of differing chemical species, either cold or hot, to achieve heat transfer. While these mechanisms have distinct characteristics, they often occur simultaneously in the same system.

Heat conduction, also called diffusion, is the direct microscopic exchange of kinetic energy of particles through the boundary between two systems. When an object is at a different temperature from another body or its surroundings, heat flows so that the body and the surroundings reach the same temperature, at which point they are in thermal equilibrium. Such spontaneous heat transfer always occurs from a region of high temperature to another region of lower temperature, as described in the second law of thermodynamics.

Heat convection occurs when bulk flow of a fluid (gas or liquid) carries heat along with the flow of matter in the fluid. The flow of fluid may be forced by external processes, or sometimes (in gravitational fields) by buoyancy forces caused when thermal energy expands the fluid (for example in a fire plume), thus influencing its own transfer. The latter process is often called "natural convection". All convective processes also move heat partly by diffusion, as well. Another form of convection is forced convection. In this case the fluid is forced to flow by use of a pump, fan or other mechanical means.

Thermal radiation occurs through a vacuum or any transparent medium (solid or fluid). It is the transfer of energy by means of photons in electromagnetic waves governed by the same laws.

2 Our Project Model

We choose to simulate the heat transfer problem on a Intel newly released Hades Canyon NUC board, which is an Intel CPU and AMD GPU combined board. To emphasize simulation of main functional part and to simplify the system complexity, we consider the center part of this board to be isolate with environment. The processor unit, VRAM unit and power supply unit will be the only heat resource at a same heating efficiency. The size is set to 256×256 pixels. The layout of the target object is shown in figure 1.

2.1 Mathematic Model

The heat source in our model has a constant heat generating ratio $\dot{q}(x, y, t)$ per unit area at location x, y. It has a flux of $k(x, y)\nabla T$ where k is the conductivity for heat to flow from high temperature to low temperature and its divergence $(\nabla \cdot k\nabla T(x, y, t))$ tells how much total heat is conducted away from that point

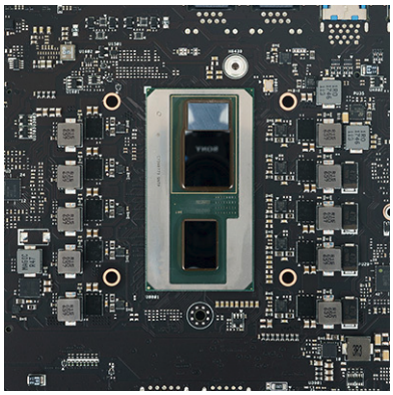

Figure 1: The center part of Intel Hades Canyon NUC

at x, y. Finally what heat remains raises the temperature $\rho c_p \dot{T}$. The equation is shown below:

$$\dot{q} = \rho c_p \frac{\partial T(x,y,t)}{\partial t} - \nabla \cdot (k \nabla T(x,y,t))$$

If we ignore the x, y dependence of k, ρ, c_p this is just the equation we are using to solver the $Ax = b$ problem in discrete form:

$$\frac{T[x,y,t+\delta t] - T[x,y,t]}{\delta t} = k \frac{T[x+h,y,t] + T[x-h,y,t] + T[x,y+h,t] + T[x,y-h,t] - 4T[x,y,t]}{h^2} + \dot{q}[x,y,t]$$

where for simplicity we have redefined $k/\rho c_p$ to k and $\dot{q}/\rho c_p$ to $\dot{q}$.

3 Three Methods We Used in Gradient Descent

Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or of the approximate gradient) of the function at the current point.

3.1 Jacobian Gradient

In numerical linear algebra, the Jacobi method (or Jacobi iterative method[1]) is an algorithm for determining the solutions of a diagonally dominant system of linear equations. Each diagonal element is solved for, and an approximate value is plugged in. The process is then iterated until it converges. This algorithm is a stripped-down version of the Jacobi transformation method of matrix diagonalization. The method is named after Carl Gustav Jacob Jacobi. Let $Ax = b$

be a square system of n linear equations, where:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

Then A can be decomposed into a diagonal component D, and the remainder R: $A = D + R$ where:

$$D = \begin{bmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{nn} \end{bmatrix}, R = \begin{bmatrix} 0 & a_{12} & \dots & a_{1n} \\ a_{21} & 0 & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & 0 \end{bmatrix}$$

The solution is then obtained iteratively via:

$$x^{(k+1)} = D^{-1}(b - Rx^{(k)})$$

where $x^{(k)}$ is the kth approximation or iteration of x and $x^{(k+1)}$ is the next or $k + 1$ iteration of x. The element-based formula is thus:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(x)} \right), i = 1, 2, ..., n$$

The computation of $x_i^{(k+1)}$ requires each element in $x^{(k)}$ except itself. Unlike the GaussSeidel method, we can't overwrite $x_i^{(k)}$ with $x_i^{(k+1)}$ as that value will be needed by the rest of the computation. The minimum amount of storage is two vectors of size n.

3.2 Multigrid Method

Multigrid (MG) methods in numerical analysis are algorithms for solving differential equations using a hierarchy of discretizations. They are an example of a class of techniques called multiresolution methods, very useful in problems exhibiting multiple scales of behavior. For example, many basic relaxation methods exhibit different rates of convergence for short- and long-wavelength components, suggesting these different scales be treated differently, as in a Fourier analysis approach to multigrid. MG methods can be used as solvers as well as preconditioners.

The main idea of multigrid is to accelerate the convergence of a basic iterative method (known as relaxation, which generally reduces short-wavelength error) by a global correction of the fine grid solution approximation from time to time, accomplished by solving a coarse problem. The coarse problem, while cheaper to solve, is similar to the fine grid problem in that it also has short- and long-wavelength errors. It can also be solved by a combination of relaxation and

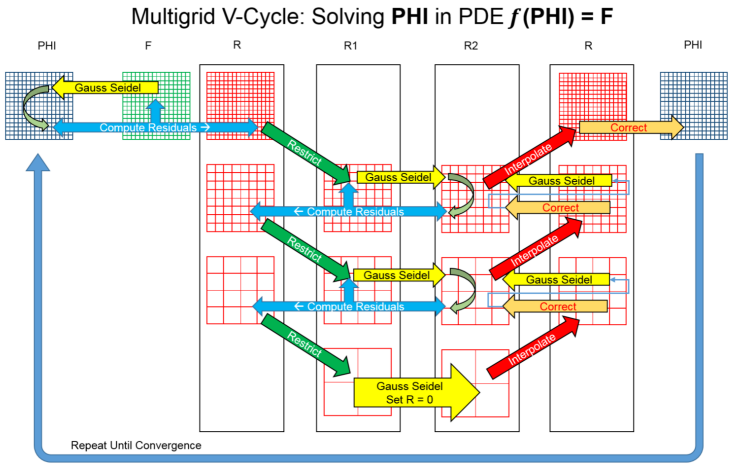

Figure 2: Multigrid V-Cycle visualization

appeal to still coarser grids. This recursive process is repeated until a grid is reached where the cost of direct solution there is negligible compared to the cost of one relaxation sweep on the fine grid. This multigrid cycle typically reduces all error components by a fixed amount bounded well below one, independent of the fine grid mesh size. The typical application for multigrid is in the numerical solution of elliptic partial differential equations in two or more dimensions.

Multigrid methods can be applied in combination with any of the common discretization techniques. For example, the finite element method may be recast as a multigrid method. In these cases, multigrid methods are among the fastest solution techniques known today. In contrast to other methods, multigrid methods are general in that they can treat arbitrary regions and boundary conditions. They do not depend on the separability of the equations or other special properties of the equation. They have also been widely used for more-complicated non-symmetric and nonlinear systems of equations, like the Lam equations of elasticity or the Navier-Stokes equations.

This approach has the advantage over other methods that it often scales linearly with the number of discrete nodes used. In other words, it can solve these problems to a given accuracy in a number of operations that is proportional to the number of unknowns.

Assume that one has a differential equation which can be solved approximately (with a given accuracy) on a grid i with a given grid point density N_i. Assume furthermore that a solution on any grid N_i may be obtained with a given effort $W_i = \rho K N_i$ from a solution on a coarser grid $i+1$. Here $\rho = N_{i+1}/N_i < 1$ is the ratio of grid points on "neighboring" grids and is assumed to be constant throughout the grid hierarchy, and K is some constant modeling the effort of computing the result for one grid point.

The following recurrence relation is then obtained for the effort of obtaining

the solution on grid k:

$$W_k = W_{k+1} + \rho K N_k$$

And in particular, we find for the finest grid N_1 that:

$$W_1 = W_2 + \rho K N_1$$

Combining these two expressions (and using $N_k = \rho^{k-1} N_1$) gives:

$$W_1 = K N_1 \sum_{p=0}^{n} \rho^p$$

Using the geometric series, we then find (for finite n):

$$W_1 < K N_1 \frac{1}{1-\rho}$$

That is, a solution may be obtained in $O(N)$ time.

3.3 Conjugate Gradient

A comparison of the convergence of gradient descent with optimal step size (in green) and conjugate vector (in red) for minimizing a quadratic function associated with a given linear system. Conjugate gradient, assuming exact arithmetic, converges in at most n steps where n is the size of the matrix of the system (here n=2). In mathematics, the conjugate gradient method is an algorithm for the numerical solution of particular systems of linear equations, namely those whose matrix is symmetric and positive-definite. The conjugate gradient method is often implemented as an iterative algorithm, applicable to sparse systems that are too large to be handled by a direct implementation or other direct methods such as the Cholesky decomposition. Large sparse systems often arise when numerically solving partial differential equations or optimization problems.

The conjugate gradient method can also be used to solve unconstrained optimization problems such as energy minimization. It was mainly developed by Magnus Hestenes and Eduard Stiefel.

The biconjugate gradient method provides a generalization to non-symmetric matrices. Various nonlinear conjugate gradient methods seek minima of non-linear equations. Suppose we want to solve linear equation $Ax = b$, the main steps in shown in figure 3.

4 Simulation

In our project, fixed initial boundary condition is considered as ideal. We set boundary condition of Jacobi method and Conjugate method in this way, but it still needs further research in setting boundary condition to multigrid method. In this case, we used periodic boundary condition to multigrid method. The

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
$\mathbf{p}_0 := \mathbf{r}_0$
$k := 0$
repeat
$\alpha_k := \dfrac{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}{\mathbf{p}_k^\mathsf{T}\mathbf{A}\mathbf{p}_k}$
$\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k\mathbf{p}_k$
$\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k\mathbf{A}\mathbf{p}_k$
if r_{k+1} is sufficiently small, then exit loop
$\beta_k := \dfrac{\mathbf{r}_{k+1}^\mathsf{T}\mathbf{r}_{k+1}}{\mathbf{r}_k^\mathsf{T}\mathbf{r}_k}$
$\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k\mathbf{p}_k$
$k := k + 1$
end repeat
The result is $\mathbf{x}_{k+1}$

Figure 3: The basic idea of conjugate gradient method

```
for (i=0;i<N;i++) {
    for(j=0;j<N;j++)
    {
        T[j+i*N] = 0.0;
        Ttmp[j+i*N]= 0.0;
        b[j+i*N] = 0.0;
    }
}
for (i = 66; i < 84; i++) {
    for (j =112;j<139;j++){
        b[j + (i) * N] = 1 ;
    }
}
for (i = 84; i < 130; i++) {
    for (j =111;j<141;j++){
        b[j + (i) * N] = 1 ;
    }
}
```

Figure 4: Initialize temperature field

initial temperature of our testing object except hear source area is zero and heat source area with initial temperature to be one.

In the first step, we set the initial temperature as mentioned above by using the method shown in figure 4. Once the residual is smaller than 0.001, our algorithm break loops and considered the stable situation has been found. By using openCV to generate colors to each pixel based on the temperature obtained by each algorithm, we can draw the heat field graph of each loop. Figure 5 shows

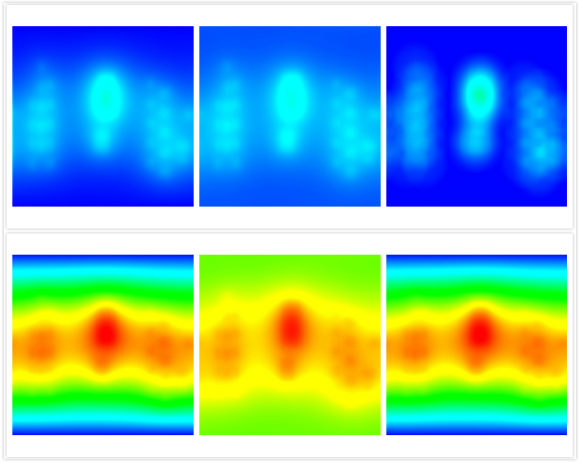

Figure 5: Simulation of temperature field in initial condition and in stable condition, from left to right: Jacobi, Multigrid, Conjugate

some temperature field samples in different time condition. As you can see, Jacobi method and conjugate method have a similar stable temperature field and multigrid method has a slightly different stable result. From our knowledge, this is happening because of the boundary condition. Multigrid method is the only one with periodic boundary condition which means the heat at upper and bottom edges can transfer to each other and higher the two edges temperature as a result.

5 Conclusion and Future Work

From figure 6 and figure 7, you can see the residual to iteration times graph. It is obvious that Jacobi method takes the most iteration times and Multigrid method and Conjugate have similar efficiency in our specific situation. In log scale graph, Jacobi method and Multigrid method have an almost linear curve so we consider their result can better demonstrate the real heat transfer process. The conjugate method takes fewest steps to simulate stable heat field, and its curve is somehow hard to predict. The reason is because Conjugate gradients next step direction is only dependent on its current step, so the graph looks a little bouncing.

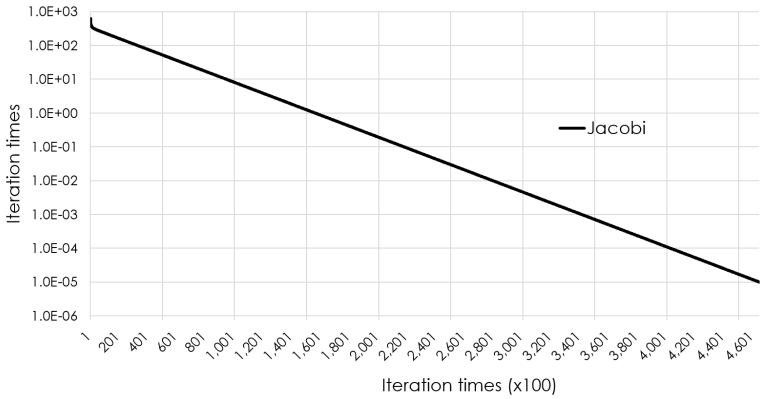

Figure 6: Iteration times to residual of Jacobi method

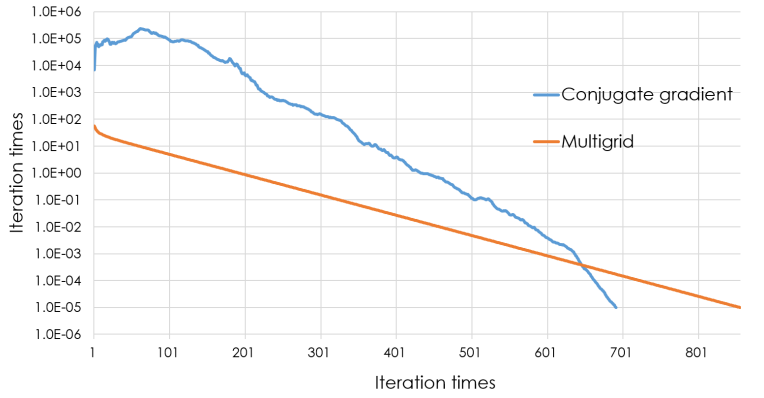

Figure 7: Iteration times to residual of Multigrid method and Conjugate method

For future research, we would like to push the simulation method one step further into 3D object. The application of Jacobi iteration of a Laplace discretisation on a uniform 3D grid is defined by:

$$u_{i,j,k}^{(n+1)} = \frac{1}{6}\left(u_{i-1,j,k}^{(n)} + u_{i+1,j,k}^{(n)} + u_{i,j-1,k}^{(n)} + u_{i,j+1,k}^{(n)} + u_{i,j,k-1}^{(n)} + u_{i,j,k+1}^{(n)}\right)$$

It is necessary to figure out the model to simulate heat transfer in multigrid method and in conjugate method. Besides, more research in changing the multigrid method boundary condition into fixed boundary is also an interesting aspect.

6 References

[1] Shewchuk, J.R., 1994. *An introduction to the conjugate gradient method without the agonizing pain*

[2] Hestenes, Magnus R.; Stiefel, Eduard (December 1952). *"Methods of Conjugate Gradients for Solving Linear Systems". Journal of Research of the National Bureau of Standards. 49 (6). doi:10.6028/jres.049.044*

[3] Atkinson, Kendell A. (1988). *"Section 8.9". An introduction to numerical analysis (2nd ed.). John Wiley and Sons. ISBN 0-471-50023-2*

[4] Avriel, Mordecai (2003). *Nonlinear Programming: Analysis and Methods. Dover Publishing. ISBN 0-486-43227-0*

[5] Golub, Gene H.; Van Loan, Charles F. *"Chapter 10". Matrix computations (3rd ed.). Johns Hopkins University Press. ISBN 0-8018-5414-8*