Parallel Jacobian Heat Equation

Maddela Sai Karthik and Srujay Reddy

Jacobian heat equation 1d

- Stationary heat equation with u as temperature: $\frac{\partial^2 u}{\partial x^2} = f$ in
- Boundaries have zero temperature, no sources: $f = 0$
- Solution is known: $u = 0$

$$u_i^{k+1} = \frac{1}{a_{ii}}(f_i - \sum_{\substack{j=1 \\ j \neq i}}^{N} a_{ij} u_j) \quad \Longrightarrow \quad \boxed{u_i^{k+1} = \frac{1}{2}(u_{i-1}^k + u_{i+1}^k)}$$

Jacobian Heat Equation 2d

A two-dimensional steady-state heat equation is governed by $\nabla^2 T = 0 \;\; i.e$

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \quad \text{Let this be } eq^n - I$$

Discritising $eq^n - I$ using cetral difference scheme:

$$\frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{\Delta x^2} + \frac{T_{i,j-1} - 2T_{i,j} + T_{i,j+1}}{\Delta y^2} = 0 \quad \text{Let this be } eq^n - II$$

As $\Delta x = \Delta y$, $eq^n - II$ now looks like

$$\frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j} + T_{i,j-1} - 2T_{i,j} + T_{i,j+1}}{\Delta x^2} = 0 \quad \text{Let this be } eq^n - III$$

Jacobian Heat Equation contd.

Transpose $eq^n - III$:

$$T_{i,j} = \frac{T_{i-1,j} + T_{i+1,j} + T_{i,j-1} + T_{i,j+1}}{4}$$

In all of the above equations, $i \rightarrow$ indexing along x grid and $j \rightarrow$ indexing along y grid.

Iterative Algorithm

By calculating the T(i,j) for each iteration using the values of previous iterations we will be able to calculate the matrix for new iteration. Now we will find the sum of squares of differences in each matrix element and will do the iterations till we reach a minimum threshold (least squares method).

Jacobi Method

The iterative formula is given by $T_{i,j} = \frac{Told_{i-1,j} + Told_{i+1,j} + Told_{i,j-1} + Told_{i,j+1}}{4}$ *and the snippet:*

```
T(i,j) = 0.25*(Told(i-1,j) + Told(i+1,j) + Told(i,j-1) + Told(i,j+1));
```

Parallelism using MPI

Using mpi we will parallelise every computation possible in each iteration.

For example we can use reduce functionality of openmp to calculate sum of all the squares of differences in a parallel manner.

#pragma omp parallel for reduction(+:sum) private(diff)

```
for (int ii=0; ii<nbx; ii++) {
        for (int jj=0; jj<nby; jj++)  {
                for (int i=1+ii*bx; i<=min((ii+1)*bx, sizex-2); i++) {
                        for (int j=1+jj*by; j<=min((jj+1)*by, sizey-2); j++) {
                                utmp[i*sizey+j]= 0.25 * (u[ i*sizey         + (j-1) ]+  // left
                                                         u[ i*sizey     + (j+1) ]+  // right
                                                         u[ (i-1)*sizey + j     ]+  // top
                                                         u[ (i+1)*sizey + j     ]); // bottom
                                diff = utmp[i*sizey+j] - u[i*sizey + j];
                                sum += diff * diff;
                        }
                }
        }
}

return sum;
```

The results obtained that are used for plotting.

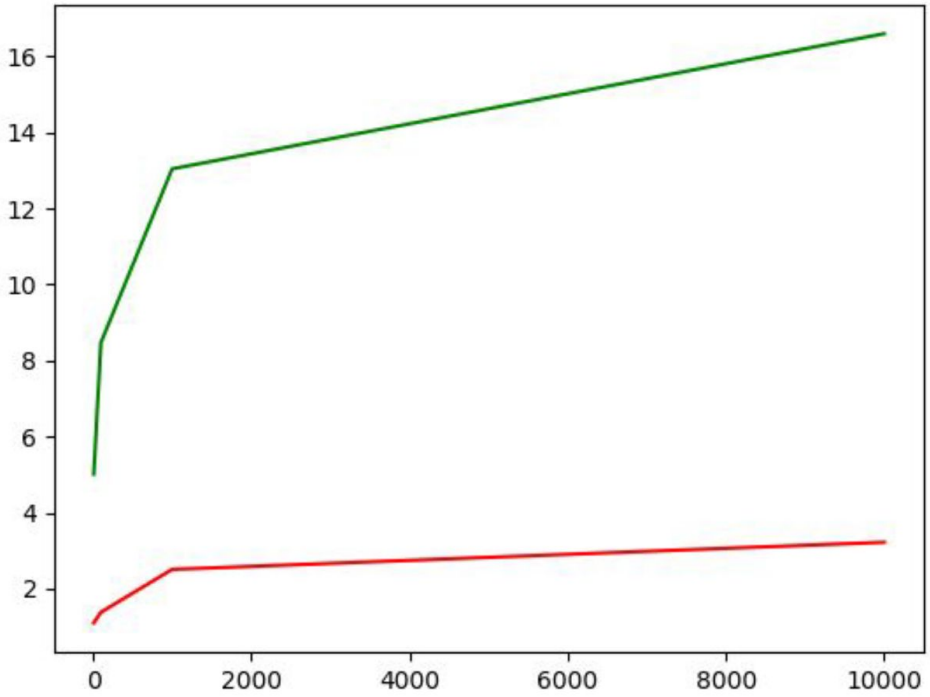
X:the number of heat resources; y: Time taken with parallelisation; z= time without parallelism ; convergence ratio = z/y.

x	y	z	convergence ratio
10	1.098	5.012	4.564663024
100	1.376	8.471	6.15625
1000	2.504	13.043	5.208865815
10000	3.218	16.6	5.15848353

No of heat resources vs time

Green : serial

Red : parallelised using mpi



No of threads vs time taken

The graph is drawn between no of threads and (x - axis) and time taken (y-axis).

The number of heat resources used are 1000.