Parallelizing a legacy CFD program

ID5130: Parallel Scientific Computing
Ganesh B, CH22B002

***Abstract*—A legacy CFD code written in C uses the SIMPLER algorithm to solve the Navier-Stokes equations. It is commonly used in modeling various heat transport phenomena. In particular, it is helpful in studying the material properties altered by a localized Gaussian laser heat source. The goal is to profile and identify various hot-spots in the program and parallelize them with familiar methods(OpenMP, OpenMPI and OpenACC) accordingly for each machine(for example, perform GPU optimizations if a GPU is available) for performance speed-up and compare their results.**

I. Introduction

Laser welding is a versatile technique with chemically pure heat source and a good control over power level to join extremely thin as well as very thick plates. Recent advances in the manufacturing processes using laser have led to increased use of advanced and dissimilar materials. There is a lot of literature available on joining of similar metals/alloys. Not much, however, is known about dissimilar metal joining, which promises an important role in the future.

A computational modelling of the transport phenomena that take place during laser welding of dissimilar metals of was developed using control volume formulation. Since the process is inherently three-dimensional in nature, a 3D transient model was developed to calculate conservation of mass, momentum, enthalpy, and composition. The model uses Patankar's SIMPLER algorithm, originally written in FORTRAN90.

The C version of the same was developed by Prof. Gandham Phanikumar for his Doctoral Thesis [1], and it has since been used extensively in various other computational models due to its versatility. Throughout this project, we will identify the major functions in the program that do the heavy lifting and try to parallelize them.

II. Computational Model

A schematic diagram of the computational domain used for numerical simulation is shown in Figure $1(a)$. Two pieces of copper and nickel with equal dimensions are kept in a butt joint arrangement. A laser heat input with a Gaussian distribution is applied on the top at the centreline of the butt joint such that the heat is distributed equally on both pieces. For a continuous welding process, melting and solidification take place simultaneously, as shown in Figure $1(b)$.

The following are the assumptions made in the model:

- The weld pool surface is flat.
- The fluid motion in the melt pool is assumed to be laminar and incompressible and driven by buoyancy and surface tension gradient (Marangoni) forces.

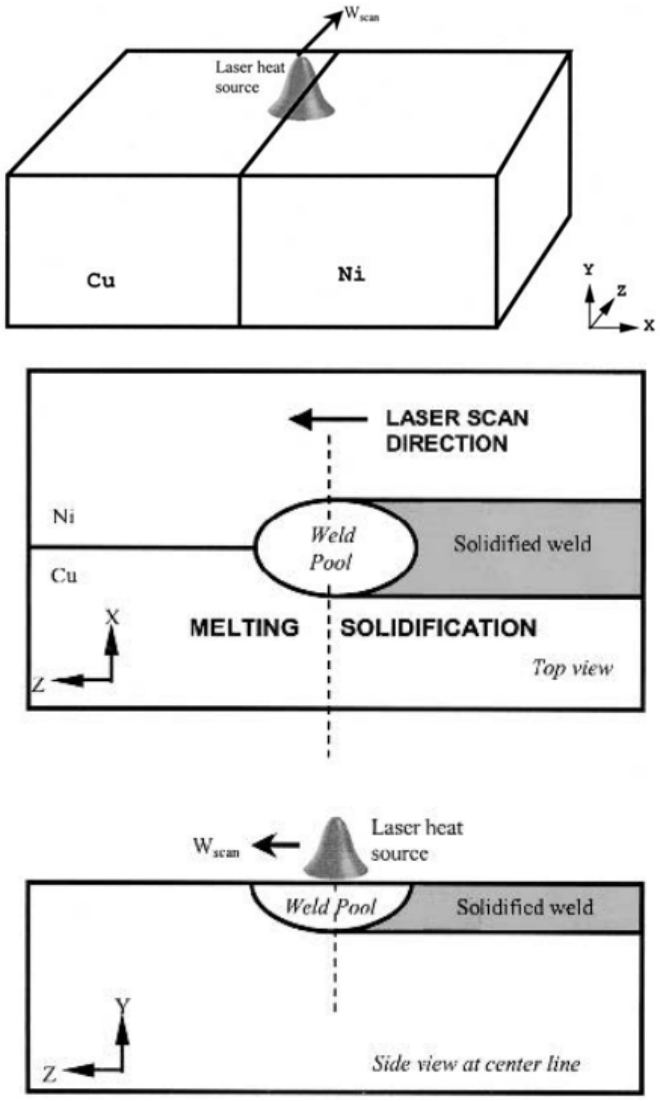

Fig. 1. (a) Schematic of laser welding setup. (b) Schematic illustrating simultaneous melting and solidification during continuous welding.

- Laser power is distributed in a Gaussian manner at the top surface. The coupling coefficient (laser power efficiency) remains uniform and constant.
- Welding is in conduction mode

The governing equations include the Navier–Stokes equation for the solution of flow of liquid, diffusion–convection equation for determination of temperature and composition

variations. The formulation, which is described in detail in the thesis [2], consists of a set of highly nonlinear partial differential equations. The governing equations can be written in the canonical form as given below.

$$\frac{\partial(\rho\phi)}{\partial t} + \nabla.(\rho U\phi) = \nabla.(\Gamma\nabla\phi) + S$$

ϕ is the variable we are solving for, such as tempera- ture, velocity, pressure or composition, ρ is the density, U is the velocity field, Γ is the diffusivity and S is the source term. The solution procedure involves reduction of the governing equations into the following form amenable for solution by tridiagonal matrix algorithm (TDMA).

$$a_i\phi_i = \sum_{j(\text{all neighbors})} a_i\phi_i + b_i$$

Here a_i is the coefficient that determines the flux between the control volume in consideration and its neighbour, and b_i is the source term for the control volume. A typical problem involves a moderate number of grids $(30\times30\times30)$, 6 variables, 1000 time-steps, 100 iterations per time-step and 6 rounds of TDMA leading to about 100 billion operations.

III. The Serial Program

Let us consider a uniform unstructured grid of size $N \times N \times N$ for easy analysis. The time taken for performing one time step ($t = 0.000005$) for various grid sizes is observed in Figure (2).

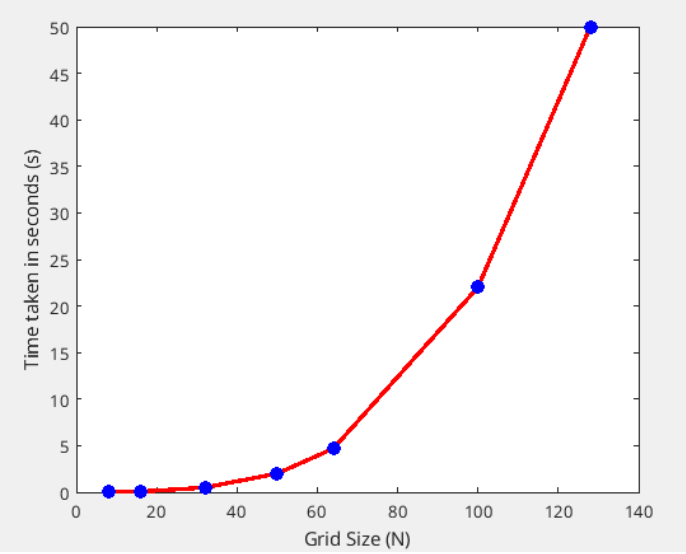

Fig. 2. Time taken for various uniform grid sizes

To gain a deeper insight, let us profile the program with GNU gprof. The results are seen in Table 1.

These are the top 4 function calls. As mentioned earlier, TDMA indeed consumes the largest time (57%!), followed by velocity coefficient functions in each direction. Therefore, we will try to parallelize the TDMA function first, followed by the velocity coefficient updating functions.

TABLE I
Function Profiling

% time	self seconds	self calls	total s/call	s/call	name
57.00	11.36	42	0.27	0.27	tdma
7.38	1.47	14	0.10	0.12	wcof
7.28	1.45	14	0.10	0.11	vcof
7.18	1.43	14	0.10	0.11	ucof

IV. Parallelizing the Program

Since we isolated the most important function calls, we will begin by having a look at its structure. The TDMA function has 6 successive triple loops. The cof functions have a lot more triple and double loops, which screams OpenMP. Although the TDMA function has a lot of loop dependencies, there are none in the cof functions. Let us look at the performance when the program is run with 4 (the optimal number on the machine) threads :

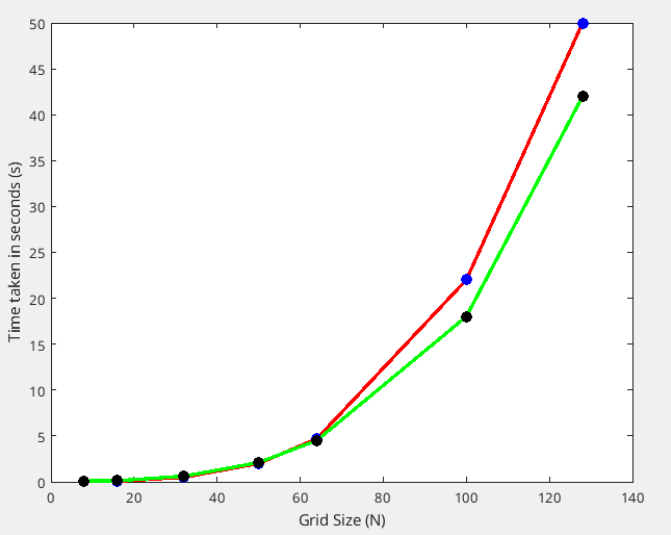

Fig. 3. Comparison of time taken for various uniform grid sizes with OpenMP

The speed-up $\psi(n, p)$ is observed in Figure(4).

We notice a decent improvement in the performance of the program. An OpenACC version of the program will be very similar to the OpenMP version, with extra care given to data handling. MPI is possibly the most useful out of them all when it comes to the TDMA function, since it naturally supports the Cartesian data distribution. I wish to continue working on the MPI version this summer.

V. Conclusion

Laser welding of dissimilar materials consisting of a copper-nickel couple has been studied numerically. In spite of some simplifying assumptions, the model is able to capture some of the key features of the process such as differential heating of the two metals, asymmetric weld pool development, and mixing of molten metals. The computational results show a

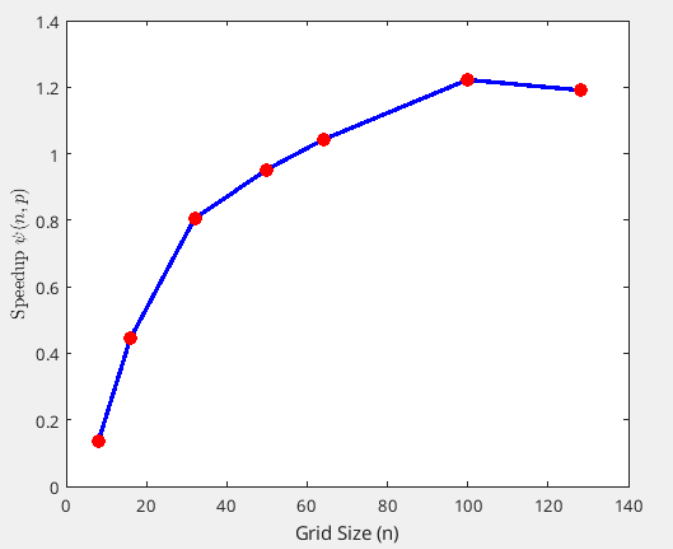

Fig. 4. Speedup $\psi(n,p)$ vs n

good qualitative agreement with the corresponding experimental ones.

The OpenMP version produced noticeable speedup. The solution was indeed verified to be the same as the serial one. Below are few plots of the results at the first timestep $t = 0.000005s$ after post processing with MATLAB.

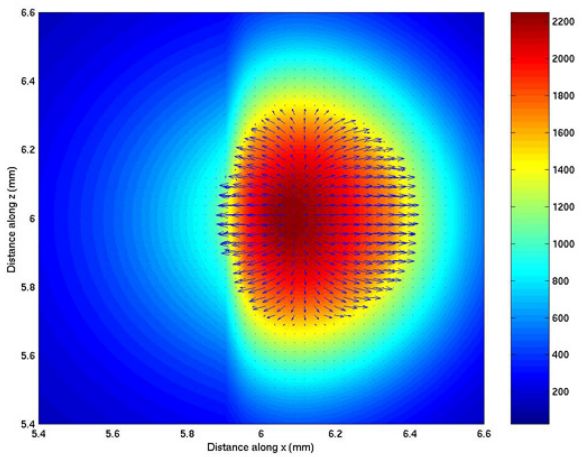

Fig. 5. Temperature variation at the first timestep

VI. Acknowledgement

I would like to thank Prof. Gandham Phanikumar, Dept. Of Metallurgy and Material Sciences, IITM for letting me use his program for this project. Working with such a huge program for the first time taught me a lot and I would like to thank him, Prof. Kameswararao and the TAs for guiding me throughout the course.

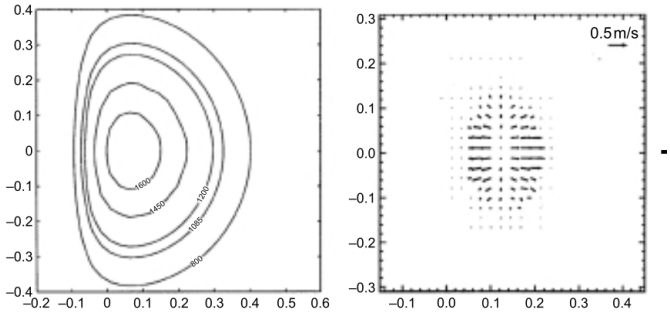

Fig. 6. Temperature contours and velocity profile at the first time step

VII. Nomenclature

- cof: coefficients
- TDMA: Tri-Diagonal MAtrix

List of Figures

List of Tables

References

[1] 10.13140/RG.2.2.33522.50880
[2] 10.1016/S0142-727X(02)00177-7
[3] https://www.researchgate.net/publication/\289654117_Supercomputing_applications_in_materials_engineering
[4] https://link.springer.com/article/10.1007/s11663-004-0034-4