Assignment no. 3

Parallelizing Topology Optimization with MPI

DCAMM PhD Course
Scientific Computing

DTU

January 2008

1 Problem setting

This assigment comes in two different formats. A safe-bet version which is designed for newcomers to Fortran and MPI. To satisfy the needs by experienced users of Fortran and/or MPI a jack-pot version has also been created.

The main idea in this project is to develop and study parallelization of a simple code which uses a method from mechanical engineering. In this project the finite element method is used, but it could equally well has been the finite volume method. The underlying iterative solver is a conjugate gradient method (see e.g. [3, 9]) which is used in a wide range of applications where an equilibrium equation is to be solved. Your expertise is needed, since in practice it is very easy to generate large problems which call for an efficient and parallelized solution technique.

The code is provided as a serial MATLAB version which is an iterative version of the 99-line topology optimization code [4, 2]. A large part of it has already been translated to Fortran and your task is to translate a minor part. Your main task is to parallelize the code as described in the following sections.

If you find errors, have comments or suggestions for service packs please let me know.
Good luck
Allan Roulund Gersborg

Department of Mechanical Engineering, Solid Mechanics
Technical University of Denmark
Building 404, room 128
e-mail/homepage: arg@mek.dtu.dk / www.topopt.dtu.dk/arg

2 Code

You find the code at the project home page: www.topopt.dtu.dk/MPI08

There should be no problems in running the MATLAB code. The difficulty is to get an overview of the computational flow, see [4] for more information.

The Fortran code is provided with makefiles which allows you to run it on either the SUN machines or on a Linux Cluster. If you have access to a linux cluster this allows you to compare the parallel efficiency and wall clock time of the two architectures. Moreover this gives an option to look into the issue of code portability. This is of course very relevant since it enables direct use the codes that you have developed in this course at your home institution.

With respect to plotting in Fortran, you can either export files to MATLAB or Paraview, where the latter is recommended. Use Google to find the download site if you want to install Paraview on your laptop. To transfer files one can use e.g. filezilla and connect to the ftp server at DTU ftp.student.dtu.dk. The program ThinLinc may be used to connect to from a windows laptop to the SUN computer (use thinlinc.gbar.dtu.dk) in order to work from home.

3 Safe-bet version

1. Translate the MATLAB code to Fortran. Briefly verify that your translation was successful. When does it make sense to use MATLAB ?
2. Identify the largest bottleneck in the FORTRAN code. Implement a parallelization strategy. Efficiencies larger than 90% have been reported in the literature for MPI implementations (see e.g. [1, 6]). The efficiency is, however, affected by many problem and implementation parameters, thus it strongly problem dependent and 90% can be used as a value to strive for.
3. Discuss the parallel efficiency in terms of efficiency (Speedup/processor) and wall clock time. Is there a difference in efficiency between the 2D and 3D problem?
4. Follow your own ideas.

3.1 More ideas

- Study the efficiency of the iterative solver. How does the condition number of the system affect the rate of convergence?
- To improve the scalability of your code you may implement that data is dumped in parallel. Use Paraview to visualize the data. If you want to visualize a field or similar look on the internet how this should be done for a format that Paraview can read.
- Implement a multi grid pre-conditioner to reduce the number of conjugate gradient iterations.
- If you have a background in (topology) optimization you may want to study the impact of changing the optimization algorithm from the present Optimality Criterion [2] to the MMA algorithm[7], the GCMMA algorithm[8], SNopt[5] or similar.

4 Jack-pot version

Rather than doing Fortran and MPI programming from scratch, it may be advantageous to use the existing PETSc library, see http://www-unix.mcs.anl.gov/petsc/petsc-as/. PETSc is *reported* to be a set of scalable routines which are written by experts to run on massively parallel machines. Essentially the end-user only needs to specify the (local) element matrix and the (global) boundary conditions.

We have access to one Linux cluster where PETSc is installed, talk with Allan RG if this version has your interest. Warning: This exercise has not been tested on the Linux machine, and the teacher is not an expert on PETSc but very interested in testing its performance. Thus, this exercise is associated with a high potential and a high risk.

1. Translate the MATLAB code to Fortran. Briefly verify that your translation was successful.
2. Make a PETSc implementation of the Fortran program based on the PETSc documentation.
3. As for the safe-bet version, investigate the parallel efficiency. How well does the iterative solver in PETSc perform ?
4. Follow your own ideas.

5 Writing a report on the results

The report will be graded based on its technical depth and accuracy, its organization and the effectiveness of your presentation. Therefore, make rational and structured reasoning and analysis, express your self clearly and briefly, and explain your findings properly. Discuss with fellows, but do not copy. Cite relevant references that you have read.

The report has to have the following issues covered:

1. Problem description
2. A brief theoretical part
3. Numerical experiments
4. Conclusions and a few relevant ideas for extensions of the work

The report must be written in English. A listing of the program code has to be attached to the report. Standard requirements are put on the design of the code, namely, structure and comments. You are encouraged to work in pairs. However, all topics in the assignment should be covered by each student, and you have to hand in individual reports!

6 Deadlines and credit points

The full report (paper copy!) must be submitted no later than January 30, 2008. Assignments submitted after this date will not be approved.

7 Acknowledgements

The author thanks Jonas Dahl and other members of the TopOpt group for helpful discussions related to this exercise.

References

[1] Bane, M. and Keller R. (2000) *A Comparison of MPI and OpenMP Implementations of a Finite Element Analysis Code*, url = "citeseer.ist.psu.edu/bane00comparison.html"

[2] Bendsøe, M. P. and Sigmund, O. (2003) *Topology Optimization - Theory, Methods, and Applications*, Springer Verlag: Berlin Heidelberg

[3] Johnson, C. (1987) *Numerical solutions of partial differential equations by the finite element method*, ISBN: 91-44-25241-1, Lund: Studentlitteratur

[4] Sigmund, O. (2001) *A 99 line topology optimization code written in MATLAB*, Structural and Multidisciplinary Optimization 21(2), pp. 120-127. Click on "MATLAB program" at www.topopt.dtu.dk

[5] Gill, P.E. and Murray, W. and Saunders, M. A. (2006) *Users Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming*, Web link: http://www.cam.ucsd.edu/~peg/sqdoc7.pdf

[6] Steijl, R. and Barakos, G. and Badcock, K. (2006) *Parallelisation of CFD methods for multi-physics problems*, In P. Wesseling and E. Oñate and J. Périaux: European Conference on Computational Fluid Dynamics, TU Delft, The Netherlands, ISBN 90-9020970-0.

[7] Svanberg, K. (1987) *The Method of Moving Asymptotes – A New Method for Structural Optimization*, International Journal for Numerical Methods in Engineering, vol. 24, pp. 359-373, DOI 10.1002/nme.1620240207

[8] K. Svanberg (2002) *A Class of Globally Convergent Optimization Methods Based on Conservative Convex Separable Approximations*, SIAM Journal on Optimization, Vol. 12(2), pp. 555-573

[9] Wesseling, P. (1991) *Principles of computational fluid dynamics*, ISBN: 3-540-67853-0, Berlin Heidelberg: Springer Verlag