Imperial College London

Department of Aeronautics

AE3-422 High-Performance Computing

Assessment Report

Author:
Federico Semeraro

Student CID Number:
00862704

March 26, 2017

Abstract

This assignment includes five questions on the implementation of parallel numerical code for simulating a beam fixed at both ends and loaded with distributed and concentrated loads using finite elements. Firstly, the static equilibrium case was solved using simple matrix inversion. By considering the inertial terms, the dynamic equilibrium equations were obtained, which were solved using the Explicit Integration Scheme (Central Difference Method) and the Implicit Integration Scheme (Newmark Method). Finally, these methods were speeded up by the use of parallelisation. The programming language used was C++, git version control was used to track and record the progress, while a makefile accepting command line arguments was created for both compiling and running the code including all the targets.

Task 1

In the first task, the static equilibrium case was solved by inverting the equation:

$$[\mathbf{K}]\{\mathbf{u}\} = \{\mathbf{F}\} \tag{1}$$

using Lapack *dpbsv* function. The following case (Figure 3 in the Appendix) was used to check the program against the analytical solution and it was found to be correct ($0.010334m$).

Task 2

In this task the dynamic beam case was solve by using the following equation [2] and the Explicit Integration Scheme (Central Difference Method, equations [3] and [4] in the Appendix):

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{F} \tag{2}$$

The matrices were turned into arrays, following the rule: $\mathbf{K}(i,j)$ accessed as $\mathbf{K}[(i-1) + K_{dim} * (j-1)]$, where K_{dim} was the number of diagonals of the banded symmetric matrices (same for the diagonal $\{\mathbf{M}\}$ matrix, for which $M_{dim} = 1$). The solution was implemented via the creation of a function (called *ExpDynEqSolver*), later modified to allow the parallelisation. This makes use of the CBLAS library and the Lapack *dpbsv* function to invert the system.

Figure 1 shows the deflection at the midpoint of the beam versus time. As we can observe, the dynamic solution oscillates around the static solution once the full load is applied.

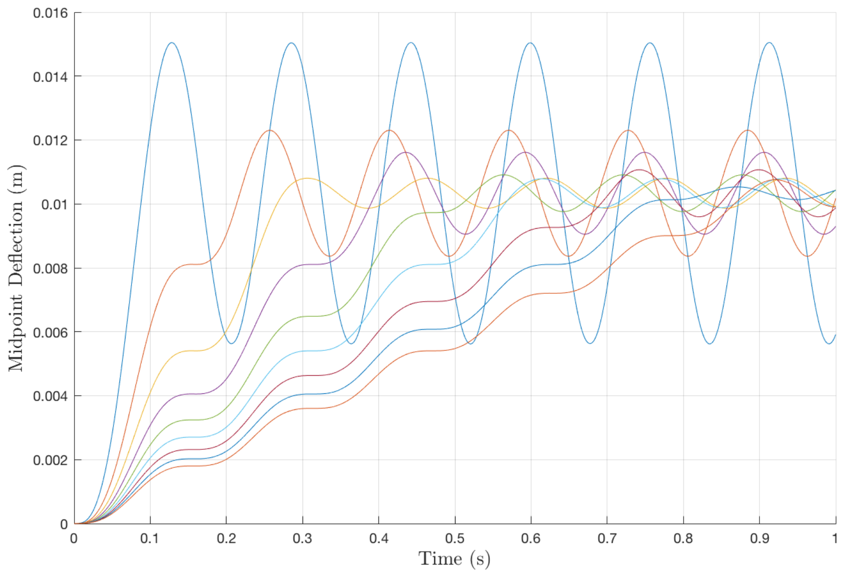

Figure 1: Deflection at the point L/2 against Time

The amplitude of the oscillations depends on how quickly the beam is loaded. The graph in Figure 2 shows the oscillations amplitude as a function of the loading time T_l in a log-log scale, keeping the time step Δt constant. This graph was obtained by running C++ and MATLAB simultaneously, constantly changing T_l and feeding the generated .txt file into a plotting script.

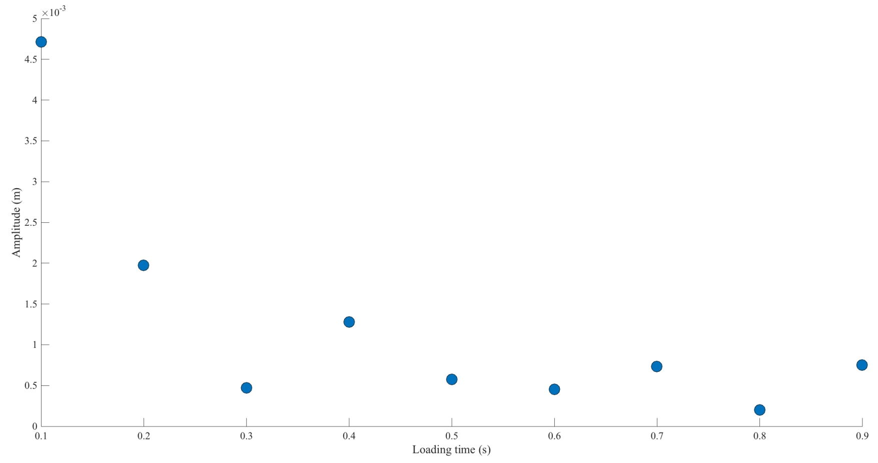

Figure 2: Oscillation amplitudes against Loading Time T_l, increasing T_l from left to right

Task 3

In this task, the dynamic equation [2] was solved using the Implicit Time Integration Scheme (Newmark Method, equations [5], [6], [7] and [8] in the Appendix). This task was compared to the previous and the same deflections were generated (Figure 1). Moreover, the same method was applied to calculate the oscillations amplitude.

Task 4

In this task, the parallelisation of task 2 was implemented using the MPI library. The domain was splitted equally into two processes, therefore triggering the forced choice of an even number of elements $N_x = 2p$. The solution was compared to the serial code and it was verified that the same deflection values were obtained.

The diagonal $\{\mathbf{M}\}$ matrix, the vector $\{\mathbf{F}\}$ and the banded $\{\mathbf{K}\}$ matrix were split into two using the same filling functions as in Tasks 1, 2 and 3 (to be rigorous they were split into $N_x/2 + 1$, so that the two processes had three nodes in common, i.e. 9 degrees of freedom). However, the Explicit Integration Scheme Solver had to be extended to allow the passage of information between the two processes in order not to lose the information about the third to last and last nodes of the first process, and the first and third nodes of the second process. This procedure exploited the symmetrical nature of the problem and ensured that the midpoint deflection was correct, while roughly halving the number of operations accomplished by each process. The *time* command was used to verify this using one process to solve Task 2 and two processes to solve Task 4.

Task 5

In this task, the parallelisation of task 3 was attempted using the same logic as in Task 4.

References

[1] Dr Chris Cantwell & Dr Omar Bacarreza, *HPC Coursework Assignment*, Department of Aeronautics, Imperial College London, 2017

Appendix

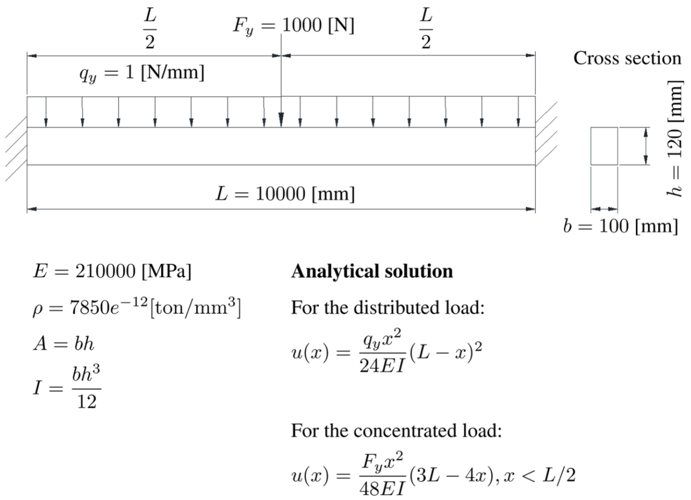

Figure 3: Beam fixed at both ends [1]

Explicit Integration Scheme (Central Differences Method):

$$\frac{1}{\Delta t^2}\left[\mathbf{M}\right]\{\mathbf{u}\}_{n+1} = \{\mathbf{F}\}_n - \left(\left[\mathbf{K}\right] - \frac{2}{\Delta t^2}\left[\mathbf{M}\right]\right)\{\mathbf{u}\}_n - \frac{1}{\Delta t^2}\left[\mathbf{M}\right]\{\mathbf{u}\}_{n-1} \tag{3}$$

$$\ddot{\mathbf{u}}_n = \frac{1}{\Delta t^2}\left(\{\mathbf{u}\}_{n+1} - 2\{\mathbf{u}\}_n + \{\mathbf{u}\}_{n-1}\right) \quad where \quad \mathbf{u}_0 = 0, \dot{\mathbf{u}}_0 = 0, \ddot{\mathbf{u}}_0 = 0 \quad at \quad t = 0 \tag{4}$$

Implicit Integration Scheme (Newmark Method):

$$\left[\mathbf{K}^{eff}\right]\{\mathbf{u}\}_{n+1} = \{\mathbf{F}\}_{n+1} + \left[\mathbf{M}\right]\left\{\frac{1}{\beta\Delta t^2}\{\mathbf{u}\}_n + \frac{1}{\beta\Delta t}\{\dot{\mathbf{u}}\}_n + \left(\frac{1}{2\beta} - 1\right)\{\ddot{\mathbf{u}}\}_n\right\} \tag{5}$$

$$\left[\mathbf{K}^{eff}\right] = \frac{1}{\beta\Delta t^2}\left[\mathbf{M}\right] + \left[\mathbf{K}\right] \tag{6}$$

$$\{\ddot{\mathbf{u}}\}_{n+1} = \frac{1}{\beta\Delta t^2}\left(\{\mathbf{u}\}_{n+1} - \{\mathbf{u}\}_n\right) - \frac{1}{\beta\Delta t}\{\dot{\mathbf{u}}\}_n - \left(\frac{1}{2\beta} - 1\right)\{\ddot{\mathbf{u}}\}_n \tag{7}$$

$$\{\dot{\mathbf{u}}\}_{n+1} = \{\dot{\mathbf{u}}\}_n + \Delta t(1-\gamma)\{\ddot{\mathbf{u}}\}_n + \Delta t\gamma\{\ddot{\mathbf{u}}\}_{n+1} \quad where \quad \mathbf{u}_0 = 0, \dot{\mathbf{u}}_0 = 0, \ddot{\mathbf{u}}_0 = 0 \quad at \quad t = 0 \tag{8}$$

where $\beta = 1/4$ and $\gamma = 1/2$ for Newmark's constant average acceleration scheme.