Summary and Advanced Topics

10

The finite element method is a huge field of study. This set of notes was designed to give students only a brief introduction to the fundamentals of the method. The implementation, theory, and application of FEM is a subject of immense literature. For general references on the subject, see the well-known books of Ainsworth and Oden [1], Becker et al. [2], Carey and Oden [3], Oden and Carey [4], Hughes [5], Szabo and Babuska [6], Bathe [7], and Zienkiewicz and Taylor [8]. For a review of the state of the art in finite element methods, see the relatively recent book of Wriggers [9]. Much of the modern research activity in computational mechanics reflects the growing industrial demands for rapid simulation of large-scale, nonlinear, time-dependent problems. Accordingly, the next concepts the reader should focus on are:

1. Error estimation and adaptive mesh refinement,
2. Time-dependent problems,
3. Geometrically and materially nonlinear problems and
4. High-performance computing: domain decomposition and parallel processing.

The last item is particularly important. Thus, we close with a few comments on domain decomposition and parallel processing.

In many cases, in particular in three dimensions, for a desired accuracy, the meshes need to be so fine that the number of unknowns outstrips the available computing power on a single serial processing machine. One approach to deal with this problem is domain decomposition. Decomposition of a domain into parts (subdomains) that can be solved independently by estimating the boundary conditions, solving the decoupled subdomains, correcting the boundary conditions by updating them using information from the computed solutions, and repeating the procedure has become popular over the last 20 years as a means of harnessing computational power afforded by parallel processing machines.

T. I. Zohdi, *A Finite Element Primer for Beginners*, The Basics,
https://doi.org/10.1007/978-3-319-70428-9_10

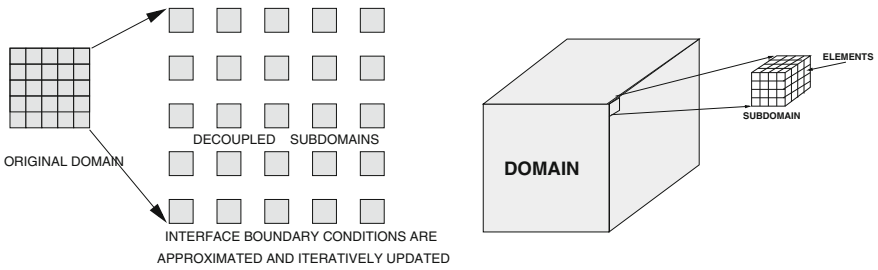

Fig. 10.1 Left: A two-dimensional view of the decomposition of a domain and Right: a three-dimensional view

Consider the three-dimensional block (an elasticity problem) where we use linear brick elements with the following parameters (Fig. 10.1):

- The number of subdomains: $M \times M \times M$.
- The number elements in each subdomain: $N \times N \times N$.

For the original (nonpartitioned domain):

- The number elements: $(N \times M) \times (N \times M) \times (N \times M)$.
- The number nodes: $(N \times M + 1) \times (N \times M + 1) \times (N \times M + 1)$.
- The number degrees of freedom (for elasticity): $3 \times (N \times M + 1) \times (N \times M + 1) \times (N \times M + 1)$.
- The number elements in the entire decoupled domain: $(N \times M) \times (N \times M) \times (N \times M)$.
- The data storage for the entire domain: $(N \times M)^3 \times 300$ (symmetric storage for elasticity).

For the partitioned domain:

- The number nodes in each subdomain: $(N + 1) \times (N + 1) \times (N + 1)$.
- The number degrees of freedom (for elasticity) in each subdomain: $3 \times (N + 1) \times (N + 1) \times (N + 1)$.
- The data storage per subdomain: $N^3 \times 300$ (symmetric storage for elasticity).

Let us now consider:

- The number processors involved: P.
- The number iterations needed to update the interface solution: I.

The operation counts for solving the whole domain is

$$C_d \propto ((3(NM + 1))^3)^\gamma, \tag{10.1}$$

while for each subdomain

$$C_{sd} \propto ((3(N+1))^3)^\gamma, \tag{10.2}$$

where $1 \leq \gamma \leq 3$ is an exponent that reflects the extremes of solving efficiency. The ratio of the amount of work done by solving the total domain to that of solving the subdomain problems (taking into account the number of iterations (I) needed to update the interface boundary conditions) is approximately

$$\frac{C_d}{IC_{sd}} \propto \frac{((3(NM+1))^3)^\gamma}{I((3(N+1))^3)^\gamma} \approx \frac{M^{3\gamma}}{I}, \tag{10.3}$$

where we have ignored the costs of computing the updated interface conditions (considered small). If we assume that the amount of time to solve is also proportional to the operation counts, and assume that each domain is processed in the same amount of time, using P processors yields:

$$\frac{C_d}{IC_{sd}/P} = \frac{PM^{3\gamma}}{I}. \tag{10.4}$$

In order to understand the scaling numerically, consider

- One-thousand processors: $P = 10^3$,
- One-thousand subdomains: $M \times M \times M = 10 \times 10 \times 10$.
- The number of updates: $I = 10^2$.

The resulting ratio of computational costs is:

- For $\gamma = 3$: $\frac{C_d}{IC_{sd}/P} = 10^{10}$,
- For $\gamma = 2$: $\frac{C_d}{IC_{sd}/P} = 10^7$.
- For $\gamma = 1$: $\frac{C_d}{IC_{sd}/P} = 10^4$.

This idealized simple example illustrates the possible benefits in reduction of solution time, independent of the gains in data storage. For a historical overview, as well as a thorough analysis of the wide range of approaches, see Le Tallec [10]. In many cases, interprocessor communication and synchronization can be a bottleneck to obtain a high-performance parallel algorithm. The parallel speedup (relative to a sequential implementation), S, can be approximated by Amdahl's law (Amdahl [11]), $S = \frac{1}{1-f}$, where f is the fraction of the algorithm that is parallelizable. For example, if 40% of the code is inherently sequential, then $f = 0.6$ and $S = 2.5$. This provides an upper bound on the utility of adding more processors. A related expression is "Gustafson's law" Gustafson [12], $S(f) = f - k(f-1)$, where k represents the parts of the algorithm that are not parallelizable. Amdahl's law assumes that the problem is of fixed size and that the sequential part is independent of the number of processors; however, Gustafson's law does not make either of these assumptions. We refer the

reader to the works of Papadrakakis et al. [13,14] for parallel strategies that are directly applicable to the class of problems of interest.

Remarks: Some comments of the convergence of such iterative schemes are provided in Appendix C.

References

1. Ainsworth, M., & Oden, J. T. (2000). *A posterori error estimation in finite element analysis*. New York: Wiley.
2. Becker, E. B., Carey, G. F., & Oden, J. T. (1980). *Finite elements: An introduction*. Englewood Cliffs: Prentice Hall.
3. Carey, G. F., & Oden, J. T. (1983). *Finite elements: A second course*. Englewood Cliffs: Prentice Hall.
4. Oden, J. T., & Carey, G. F. (1984). *Finite elements: Mathematical aspects*. Englewood Cliffs: Prentice Hall.
5. Hughes, T. J. R. (1989). *The finite element method*. Englewood Cliffs: Prentice Hall.
6. Szabo, B., & Babúska, I. (1991). *Finite element analysis*. New York: Wiley Interscience.
7. Bathe, K. J. (1996). *Finite element procedures*. Englewood Cliffs: Prentice Hall.
8. Zienkiewicz, O. C., & Taylor, R. L. (1991). *The finite element method* (Vol. I and II). New York: McGraw-Hill.
9. Wriggers, P. (2008). *Nonlinear finite element analysis*. Berlin: Springer.
10. Le Tallec, P. (1994). Domain decomposition methods in computational mechanics. *Computational Mechanics Advances*, *1*, 121–220.
11. Amdahl, G. (1967). The validity of a single processor approach to achieving large-scale computing capabilities. In *Proceedings of AFIPS Spring Joint Computer Conference* (pp. 483–485). Atlantic City, N. J.: AFIPS Press.
12. Gustafson, J. L. (1988). Reevaluating Amdahl's law. *Communications of the ACM*, *31*(5), 532–533.
13. Papadrakakis, M. (1993). *Solving large-scale problems in mechanics*. New York: Wiley.
14. Papadrakakis, M. (1997). *Parallel solution methods in computational mechanics*. Chichester: Wiley.