East Asian Journal on Applied Mathematics
doi: 10.4208/eajam.210715.250316a

Vol. **6**, No. 3, pp. 278-289
August 2016

Parallel Solution of Linear Systems

Sidi-Mahmoud Kaber[1,*], Amine Loumi[2] and Philippe Parnaudeau[1]

[1] *Sorbonne Universités, UPMC Univ Paris 06, CNRS, UMR 7598, Laboratoire Jacques-Louis Lions, 4, place Jussieu 75005, Paris, France.*
[2] *Laboratoire de Mathématiques et Applications, Université Hassiba Benbouali, Chlef, Algeria and Laboratoire d'Équations aux Dérivées Partielles non linéaires et d'Histoire des Mathématiques, ENS-Kouba, Algers, Algeria.*

Received 21 July 2015; Accepted (in revised version) 25 March 2016.

Abstract. Computational scientists generally seek more accurate results in shorter times, and to achieve this a knowledge of evolving programming paradigms and hardware is important. In particular, optimising solvers for linear systems is a major challenge in scientific computation, and numerical algorithms must be modified or new ones created to fully use the parallel architecture of new computers. Parallel space discretisation solvers for Partial Differential Equations (PDE) such as Domain Decomposition Methods (DDM) are efficient and well documented. At first glance, parallelisation seems to be inconsistent with inherently sequential time evolution, but parallelisation is not limited to space directions. In this article, we present a new and simple method for time parallelisation, based on partial fraction decomposition of the inverse of some special matrices. We discuss its application to the heat equation and some limitations, in associated numerical experiments.

AMS subject classifications: 65Y05, 65F05, 65F10

Key words: Parallel computation, numerical linear algebra.

1. Introduction

A major challenge in scientific computing is to decrease the time required in numerical simulations. To achieve this, algorithms and scientific software must be adapted to fit new computer architectures, often involving parallel programming paradigms. Parallelism is not a new topic in the High Performance Computing (HPC) community, but it has become important for all programmers from the beginning of the century when there were maybe 2 cores per processor, for there are typically 12 cores already and some $60-80$ cores are expected in the near future.

The Domain Decomposition Method (DDM) is an efficient approach to parallelisation in spatial directions, where the whole domain is subdivided to produce efficient iterative

*Corresponding author. *Email address:* kaber@ljll.math.upmc.fr (S.-M. Kaber)

and independent solutions of smaller problems on the resulting subdomains. Correction steps are necessary to propagate information from one subdomain to others. However, the time direction is also a candidate for parallelisation. In 1964, Nievergelt [1] introduced a parallel algorithm based on time decomposition, and a few years later Miranker *et al.* [2] defined a parallel solver based on a predictor-corrector scheme. Ref. [7] describes the state of the art on time parallel integration.

In Section 2, we present our Partial Differential Equation (PDE) solver involving time parallelisation, where the original linear system is split into uncoupled linear systems to be solved separately. Our method falls in the category of "Direct Parallel Time Integration" — cf. Refs. [3,5]. We eventually discuss its application to the two-dimensional heat equation in Section 3 and some limitations of our method in Section 4, and make brief concluding remarks in Section 5.

2. Parallel Computation for Linear Systems

Let $(A_i)_{i=1}^m$ be a collection of m nonsingular real matrices of size $n \times n$ and set

$$X = A_1 \cdots A_m \, .$$

We are interested in the following problem: how to compute quickly the solution $x \in \mathbb{R}^n$ of the linear system

$$Xx = y \, , \tag{2.1}$$

where $y \in \mathbb{R}^n$ is any given vector?

Sequential approach. One should automatically eliminate the computation of the product of all m matrices before solving the linear system, since this is very expensive (requires a large computational time). Let us recall that solving a generic $n \times n$ linear algebraic system by Gauss elimination (n large) requires $n^3/3 + \mathcal{O}(n^2)$ operations. (For simplicity, we only take into account multiplications and divisions, neglecting additions and subtractions.) The product of two matrices of size $n \times n$ involves n^3 operations, so computation of the product X above requires mn^3 multiplications. The following algorithm is a sensible sequential computation of the solution:

- compute the vector $x_1 \in \mathbb{R}^n$ such that $A_1 x_1 = y$;
- then compute x_2, the solution of $A_2 x_2 = x_1$;
- ...
- and ultimately compute $x_m = x$, the solution of $A_m x_m = x_{m-1}$.

Remark 2.1. The computing cost of this algorithm is m times the computing cost of solving one linear system — i.e. $mn^3/3$ operations.