# PARALLEL ALGORITHMS FOR THE ANALYSIS OF NONLINEAR BETATRONIC MOTION*

M. Giovannozzi†, INFN, Sezione di Bologna, Italy
E. McIntosh, CERN, 1211 Geneva 23, Switzerland

*Abstract*

The dynamic aperture represents the volume in phase-space in which stable motion occurs. This is a key parameter to define the performance of a circular machine. The computation of this volume requires CPU-intense numerical simulations.

In this paper, some original parallel algorithms to speed-up the evaluation of the dynamic aperture are presented. A detailed analysis of different algorithms is carried out. Furthermore, the dependence of the CPU-time on the phase-space parameters as well as the load balancing of the proposed techniques are studied.

## 1 INTRODUCTION

The particles circulating in superconducting accelerators experience nonlinear forces which produce strong instabilities and losses. These effects could prevent safe operation of the machine with a consequent reduction of the overall performance.

The main parameter to quantify these harmful effects is the so-called dynamic aperture (hereafter DA); this is the volume in phase-space in which stable motion occurs. A large dynamic aperture implies a wide stable region where one can operate without experiencing particle losses.

In a previous paper [1], the problem of computing the DA in the presence of strong nonlinear perturbations was analysed. Some algorithms were described and numerical simulations were carried out to test the precision of the proposed techniques. All these tools were implemented in PLATO [2], a program library developed to analyse nonlinear phenomena in accelerator physics.

To further improve the efficiency in the computation of the dynamic aperture, in the sense of improving the accuracy and/or the CPU-time needed by the calculations, a promising approach is to convert the sequential tools developed so far into parallel algorithms [3].

Nowadays, computer systems with a relatively large number of processors (10 to 100) are widely available, either as workstation clusters, shared memory multiprocessor machines, or scalable parallel-processing systems with physically distributed memory. At CERN, for instance, there are several such systems of each type and, in particular, there is a MEIKO CS-2, a scalable parallel computer with 128 processors, developed with the support of the European Union, in the framework of the ESPRIT III Programme, and installed at CERN in 1994.

* Work partially supported by EC Human Capital and Mobility contract Nr. ERBCHRXCT940480.

† Present address: CERN PS Division

## 2 DYNAMIC APERTURE: DEFINITION AND NUMERICAL COMPUTATION

To define the dynamic aperture, the first step is to consider the phase-space volume of the initial conditions which are bounded after $N$ turns:

$$\int\int\int\int \chi(x, p_x, y, p_y)\, dx\, dp_x\, dy\, dp_y, \qquad (1)$$

where $\chi(x, p_x, y, py)$ is the characteristic function of the set, i.e. it is unity if $(x, p_x, y, p_y)$ is stable and zero otherwise. Since in 4D the invariant curves (i.e. 2D KAM tori) do not separate different domains of phase-space, the concept of a last invariant curve surrounding stable initial conditions is no longer valid [4, 5]. Although, from a purely theoretical point of view, the dynamic aperture could be a rather peculiar set, numerical simulations of lattices modelling circular machines showed that these situations are not typical [6, 7, 8, 9]. This means that, in general, there exists a connected region of initial conditions which are stable for a given number of iterations.

Two methods can be used to compute the integral in Eq. (1) (see Ref. [1] for more details).

**Method 1: direct integration** To exclude the disconnected part of the stability domain in the integral (1), we have to use polar variables. A scan of the four phase-space variables is performed and the DA evaluated by averaging the value of the maximum stable amplitude over the angular variables.

**Method 2: integration over the dynamics** A scan along two phase-space variables is performed. The information on the two neglected directions can be recovered via an average procedure.

## 3 PARALLEL ALGORITHMS TO EVALUATE THE DA

In both algorithms to evaluate the DA, the computation is split into two stages: firstly the last stable radius, as a function of the phase-space parameters, is determined, and secondly the DA is evaluated. It is clear that the algorithm has a natural parallel structure. Two procedures can be designed [3]:

**Parallel algorithm 1: direct approach** The direct method implies a scan over the four variables. The most efficient way to parallelise such a structure is to assign to each processor the task of performing the radial scan along a given direction in the 4D phase-space to determine the maximum stable amplitude. A procedure determines which processor, among the group of $N_{proc}$ units, will perform the scan

0-7803-4376-X/98/$10.00 © 1998 IEEE

along which direction. Once the first unstable initial condition is reached, the processor stops and it becomes available to compute along a different direction. The results obtained by the different processors are then gathered together (each processor is unaware of the results obtained by the others) and only one CPU is used to perform the evaluation of the DA.

**Parallel algorithm 2: integration over the dynamics** The integration over the dynamics allows us to compute the DA by scanning over two phase-space variables. The only difference with respect to the direct method is the evaluation of an average to recover the information from the neglected variables. As for the direct method, the most efficient way to parallelise this algorithm is to assign to each processor the task of determining the last stable radius for a given direction.

## 4 LOAD BALANCE

An important issue in the definition of a parallel algorithm is the load balance between the different processors; clearly, the optimal solution is to have an equal amount of work for each of the CPUs. Two principal methods of assigning work, static and dynamic, each with two variants, have been implemented and evaluated [3].

**Static Cyclic allocation**: the cases over which the parallelization is performed are divided into blocks of length $N_{proc}$. the $j$th processor will only work on the cases with $j + kN_{proc}$ where $0 \leq k \leq [N_{cases}/N_{proc}]$. This approach is rather appealing due to its simplicity.

**Dynamic allocation with a Master**: this technique allows very high computational efficiency even in those situations where the difference in CPU-time needed for different cases is relevant. As soon as a processor is available, it will start the calculations on the current case. The procedure designed to balance the work-load is based on a master-slave structure [3]. The drawbacks are the fixed overhead of $1/N_{proc}$, the overhead due to the communications between the master and the slaves, and a possible bottleneck if many slaves request a new iteration at the same time. In the cases considered, with $N_{proc}$ typically between 20 and 80, these effects were negligible with respect to the improvement introduced by the more optimal load-balance.

**Atomic Dynamic allocation**: this technique provides the same advantage as dynamic load balancing as described above, but without the fixed overhead of a master processor. Every processor uses a global shared variable to label the different cases. This method is also extremely simple, but can be implemented only on systems with a global shared memory capability such as the MEIKO CS-2.

The last method has been implemented using the MEIKO Atomic library, while the others are based on the MPI library [10].

## 5 MEIKO DESCRIPTION

The MEIKO CS-2 computer is a distributed-memory, scalable, parallel system using SPARC micro-processors and a MEIKO-developed interconnection which enables programs to read and write memory in remote nodes without context switching. The CERN CS-2 has 64 nodes, each with two 100 MHz HyperSPARC processors (rated at over 100 Specint92 per processor) and 128 MB of memory. Each node has a local disk for temporary data storage and paging or swapping as well as SCSI connections for additional peripheral equipment.

The CS-2 service is at present used for the support of data recording and event reconstruction for high energy physics experiments, and for event-parallel simulation using a specially developed version of the GEANT program [11]. It also provides a Parallel Interactive Analysis Facility (PIAF [12]). Currently under development is a system (GRACE [13, 14]) for the automatic generation of Feynman diagrams in parallel.

## 6 RESULTS

The different algorithms presented in Section 3 have been tested using a realistic lattice of the CERN Super Proton Synchrotron (SPS). The model simulates the special machine conditions used during the experimental sessions dedicated to dynamic aperture studies [15].

As a first step, the performance of the first parallel algorithm applied to the model of the SPS was analysed. For such computations, ten steps in the angular variables were used, while the number of iterations has been fixed to 1000. As far as the number of radial initial conditions is concerned, the step between two successive points was specified: the program then increases the value of the radial variable until an unstable condition is met. The simulations have been used to test how the CPU-time needed to compute the DA scales as a function of $N_{proc}$, independently of the different type of implementation. The results are shown in Fig. 1.

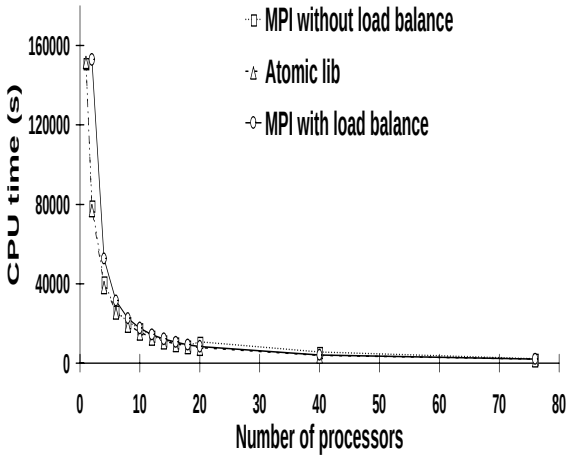


Figure 1: Performance of the first parallel algorithm. The total CPU-time is depicted as a function of $N_{proc}$.

Independent of the implementation, the CPU-time decreases with the number of processors. In this respect the performance seems to be optimal. As expected, using the

MPI library has some drawbacks: one processor is used as a master, therefore the total number of active CPUs is reduced by one with respect to the other methods. This effect is clearly visible on the plot. As soon as the $N_{proc}$ exceeds 10 units, this negative effect disappears and the beneficial effect of a good load balance makes this particular implementation more efficient than using MPI alone without load balance and practically at the same level as the algorithm based on Atomic library.

To quantify the impact of the load balance approach on the efficiency of the algorithm, we plot the normalised mean CPU-time, namely:

$$\mu(N_{proc}) = \frac{1}{N_{proc}} \sum_{i=1}^{N_{proc}} \frac{\tau_i}{\tau_{max}}, \qquad (2)$$

where $\tau_{max} = \max_{j=1,\cdots N_{proc}} \tau_i$, and $\tau_i$ represents the total CPU-time used by the $i$th processor. In the ideal case of perfect load balance $\mu(N_{proc}) = 1$. The results are reported in Fig. 2.

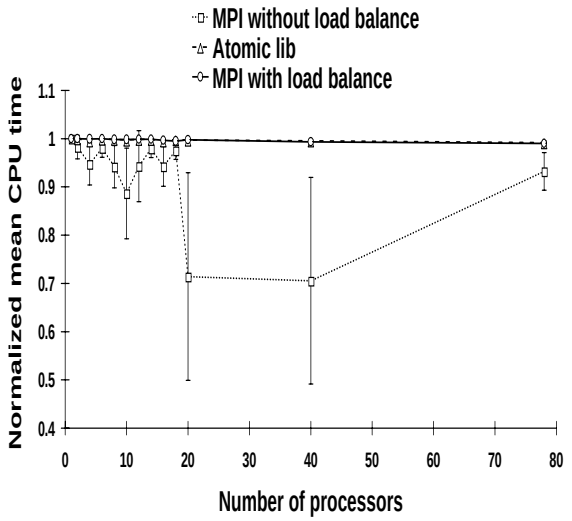


Figure 2: Load balance for the first parallel algorithm. The average CPU-time $\mu$ is shown as a function of $N_{proc}$. The error bars are computed using the standard deviation of the CPU-time for the different processors.

The difference in performance between the three implementations is clearly seen. For the algorithm without any built-in load balance control, the results are rather poor: the normalised mean time fluctuates wildly, especially when the number of processors is greater than 20. Correspondingly the CPU-time performance (see Fig. 1) becomes worse than the other algorithms.

On the other hand, the other two approaches behave very well: in both cases $\mu$ is almost constant as a function of $N_{proc}$ and very close to one. In this respect the implementation based on MPI and the one using the Atomic library are almost equivalent, because the time spent in communications is negligible with respect to the CPU-time required for computations.

Similar tests have been carried out on the second parallel algorithm giving similar results as far as the speed-up and the load-balance are concerned.

## 7 CONCLUDING REMARKS

In this paper parallel algorithms to evaluate the dynamic aperture together with their performances using a realistic model of the CERN SPS have been discussed. The results show that the algorithms are optimal and that the problem of load balance can be efficiently solved. The solutions based on the MPI and Atomic library have very similar performance: although the latter gives a CPU-time shorter by some percent than the version implemented with MPI, the overall results are comparable.

## 8 REFERENCES


[1] E. Todesco and M. Giovannozzi, Phys. Rev. **53**, 4067 (1996).

[2] M. Giovannozzi, E. Todesco, A. Bazzani and R. Bartolini, CERN PS (PA) **96–12** (1997).

[3] M. Giovannozzi and E. McIntosh, Int. J. Mod. Phys. C , in press (1997).

[4] A. Bazzani, E. Todesco, G. Turchetti and G. Servizi, CERN **94–02** (1994).

[5] J. D. Meiss, Rev. Mod. Phys. **64**, 795 (1992).

[6] W. Scandale, in *Third European Particle Accelerator Conference* , edited by H. Henke (Edition Frontiéres, Gif sur Yvette, 1993) pp. 264.

[7] F. Zimmermann, in *Fourth European Particle Accelerator Conference* , edited by V. Suller and C. Petit-Jean-Genaz (World Scientific, Singapore, 1995) pp. 327.

[8] F. Galluccio and F. Schmidt, in *Third European Particle Accelerator Conference*, edited by H. Henke (Edition Frontiéres, Gif sur Yvette, 1993) pp. 640.

[9] M. Giovannozzi, R. Grassi, W. Scandale and E. Todesco, Phys. Rev. E **52**, 3093 (1995).

[10] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra, MPI: The Complete Reference (The MIT Press, Cambridge, 1995).

[11] Application Software Group, CERN Program Library Long Writeups **Q123**.

[12] T. Hakulinen and F. Rademakers, to be published in the Proceedings of *HPCN 95, High Performance Computing and Networking Conference*, Milan - Italy, 2-5 May, 1995.

[13] T. Kaneko, Comput. Phys. Comm. **92**, 127 (1995).

[14] F. Yuasa, D. Perret-Gallix, S. Kawabata and T. Ishikawa, to be published in the Proceedings of *Fifth International Workshop on Software Engineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics*, Lausanne - Switzerland, 2-5 Sep., 1996.

[15] W. Fischer, M. Giovannozzi and F. Schmidt, Phys. Rev. E **55**, 3057 (1997).