Synchronous and asynchronous parallel bifurcation analysis of periodically forced reactors

Gaetano Continillo[1], Artur Grabski[1], Erasmo Mancusi[1] and Lucia Russo[2]

[1]Università del Sannio, Department of Engineering, Piazza Roma 21, 82100 Benevento, Italy

[2]Istituto di Ricerche sulla Combustione CNR, Naples, Italy

1 Introduction

There is an increasing interest in chemical reaction engineering towards application of bifurcation theory and parameter continuation in order to obtain a complete dynamical characterization of the mathematical model of the system (for example a reactor plant). In order to properly design and control chemical reactors, it is necessary to accurately describe all the regime conditions when relevant design and operation parameters are changed. More generally, mathematical models of chemically reactive systems could exhibit complex regime transitions marked by catastrophic bifurcations (sudden changes in temperature and/or concentrations). Bifurcation analysis is the tool of choice in investigating dynamic features of non-linear systems and, particularly, in identifying multi-stability regions. However, standard and popular codes for automatic continuation, such as AUTO (Doedel et al., 1997), are generally unsuitable for large scale systems. Moreover, these software packages work efficiently only if the mathematical model of the system has an analytic expression for the Jacobian matrix. It is worth to note that the main information about the dynamical feature of a system is represented by the eigenvalues of the Jacobian matrix. Therefore, the numerical computation of these eigenvalues is one of the main tasks in the continuation algorithms. Generally, this task is the slowest one in the execution of the continuation algorithm and, when the system under study has no analytic expression of the Jacobian matrix, it must be computed numerically and the whole process becomes even more time consuming.

Parallel computation is the most promising approach to reduce the computation time in complex numerical problems. Presence of independent computation tasks that can be conducted in parallel is not the only condition to obtain successful implementation of parallel algorithms. To obtain convenient speedup, the parallel fraction of the whole algorithm must be very high (Amdahl, 1967). For example, when bifurcation analysis requires extensive numerical work to integrate functions for independently changed integration bounds, and this task is dominant in the computing time, we are in a favourable conditions for best performance of parallel algorithms. This condition is met for a wide class of systems, which include – but are not limited to – systems for which the Jacobian matrix must be computed numerically via repeated independent numerical integrations. Several examples are found in the literature. Continillo et al. (2006) analyzed discontinuous periodically–forced reactors such as Reversed-Flow–Reactors (RFR) by conducting bifurcation analysis on a properly constructed discrete map, based on the system's Poincaré map. This map is not available in analytical form and thus it must be computed numerically. Most of the computation time is spent during repeated time integrations of the map. In this work we implement and conduct parallel bifurcation analysis of systems with require extensive independent numerical integration work. The method is applied by means of parallel versions of AUTO (Doedel et al, 1997) modified by the authors. Both synchronous and asynchronous computations are possible, depending on the available computing architecture (shared vs. distributed memory) We report implementation details and results of bifurcation analysis of the discontinuous periodically–forced distributed model of a reactor via Poincaré map.

Correspondence to : continillo@unisannio.it

2 Numerical bifurcation analysis of discontinuous periodically forced systems

Discontinuous periodically forced systems, like those arising from modelling of periodically forced reactors (see Russo et al., 2002), can be formulated in abstract form as:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{p}, g(T)) \tag{1}$$

where represents the discontinuous forcing with period T, x is the state vector and p the parameters vector. The continuous time system can be studied via its Poincaré map. The details of this approach are described in Russo et al, 2002. The authors adopt AUTO 97 (Doedel et al., 1997) to conduct the bifurcation analysis. AUTO can trace the fixed point locus of a discrete map, compute branches of stable and unstable fixed points for a discrete system, and compute the Floquet multipliers that determine the stability along these branches. Standard use of AUTO requires that the user supplies an analytic expression of the discrete-time system. In their case, an analytic expression for the map is unavailable, thus they resort to numerical evaluation of the map. The technique consists of an interaction between AUTO (or any equivalent continuation software) and an ODE solver which efficiently evaluates the map. More explicitly, for the continuous-time forced system reported in Eq. 1, let the Poincaré map **P** be:

$$\mathbf{x}_{k+1} = \mathbf{P}\left(\mathbf{x}_k, \mathbf{p}\right) \tag{2}$$

The map must be evaluated numerically. For periodically forced systems the Poincarè map is easily constructed by sampling the time trajectory at each period *T*. Then, starting from $\mathbf{x}_k$, the equations of the continuous–time system are integrated over a time period *T* and the result can be assumed as initial condition of a new time integration of the equations of the continuous time system, again on a time interval *T*. Numerically, the continuation of the fixed points of the map is conducted with calls from the AUTO main routine to an external integrator. The vector state of the system is sent to the integrator, which sends it back after a time equal to *T*, for the one-iterate. The bifurcation analysis is conducted by finding the zero of the algebraic equation:

$$\mathbf{x} - \mathbf{P}\left(\mathbf{x}, \mathbf{p}\right) = \mathbf{F}\left(\mathbf{x}, \varphi_T\left(\mathbf{x}\right), \mathbf{p}\right) = 0 \tag{3}$$

where $\varphi_T\left(\mathbf{x}\right)$ represents the integration operator along the time variable. Of course, since the map has no analytical expression, this map and the Jacobian are to be computed numerically. This involves expensive numerical integrations in a number that grows with the square of the order of the reduced dynamical system. In our examples, function evaluations include numerical integration of given expressions. Parallelism applies to the numerical computation of derivatives. As an example, the following steps are necessary in order to compute the Jacobian matrix by means of second order difference operators:

1. Prepare *2n* perturbed vectors of starting conditions:

$$\mathbf{x}_{1F} = \begin{bmatrix} x_1 + \varepsilon \\ x_2 \\ \dots \\ x_n \end{bmatrix}, \quad \mathbf{x}_{1B} = \begin{bmatrix} x_1 - \varepsilon \\ x_2 \\ \dots \\ x_n \end{bmatrix} \quad \dots \quad \mathbf{x}_{nF} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n + \varepsilon \end{bmatrix}, \quad \mathbf{x}_{nB} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n - \varepsilon \end{bmatrix} \tag{4}$$

2. Compute values of right hand sides with suitable perturbed conditions. Each of these operations takes significant computation power.

$$\mathbf{F}_{iB} = \mathbf{F}(\mathbf{x}_{iB}), \quad \mathbf{F}_{iF} = \mathbf{F}(\mathbf{x}_{iF}) \qquad i = 1, 2 \dots n \tag{5}$$

3. Compute the numerical derivatives by second order finite difference operators. These operations do not take so much computation power.

$$\frac{\partial \mathbf{F}(\mathbf{x})}{\partial x_i} = \frac{\mathbf{F}_{iF} - \mathbf{F}_{iB}}{2\varepsilon} \qquad i = 1, 2 \dots n \tag{6}$$

where *n* is the dimension of the system (number of state variables) and ε is an arbitrary small perturbation. Subscript *F* means forward, *B* means backward perturbation. It is worth to stress here that all the operations expressed by Eqs (4, 5, 6) are intrinsically independent and might be run in parallel. Particularly, tasks (5) take

almost the entire computation time and are therefore distributed among available processors. It appears that the maximum number of processors that can be usefully involved is limited by twice the dimension of the system.

3 Synchronous and asynchronous approaches

We used MPI and PTHREADS parallel libraries for distributing tasks around the cluster. There are two possible computation regimes, namely synchronous and asynchronous.

The Synchronous regime is designed for homogeneous clusters and for shared memory Symmetric Multi Processing (SMP) machines. This regime gives speedup starting from a 2–processor system. Performance of the algorithm is limited by the slowest processor. Generally in such systems all processors have the same performance and therefore if the job is equally divided among processors. Delivery of results is expected to occur synchronously. For large systems it is expected that the delay of the network connection does not play a significant role. To investigate this aspect, a comparison between a shared memory Symmetric Multi Processor (SMP) machine and a distributed cluster is conducted. SMP machines are composed by few high–performance nodes and latency is very small, as it is governed by the memory bus speed, whereas typically distributed–memory machines (clusters) are made by many medium–performance nodes connected by a network. If the parallel speedup is comparable between the two kinds of architectures, it will be concluded that communication speed is not too important in our problems.

The Asynchronous regime is best suited for heterogeneous clusters or GRID computing, because it distributes the job dynamically for each node. So for example when slower and faster computers are used or the external load on the cluster is unknown/unpredictable, the algorithm behaves similarly to peer–to–peer networks. The speed in general is not limited by the slowest processor. This concept requires at least three processors to yield speedup (one master and two slaves). Since the work is partitioned into $2n$ independent simulations, this concept works well when the dimension of the system is larger than half of the number of available nodes, otherwise, all jobs would be allocated at the first distribution, dynamic allocation would not be possible and speedup would again be limited by the slowest processor.

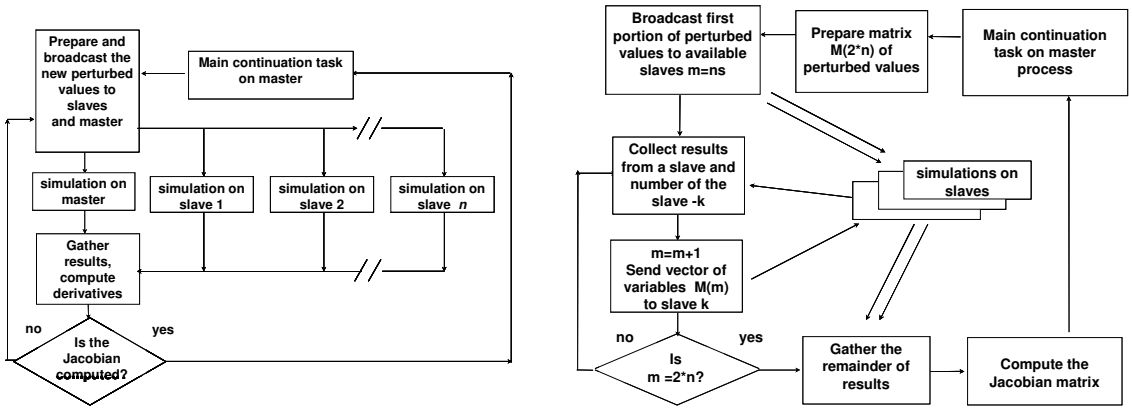


Figure 1 - Sketch of synchronous(left) and asynchronous (right) Jacobian subroutine

The software was prepared and run on a Linux cluster equipped with ROCKS 3.2.0 operating system and LAM-MPI 7.0.6. We used a system with a 8-processor shared memory machine Intel ® Xeon ™ MP CPU 2.80GHz 8Gb RAM plus 6 individual nodes 1.8 GHz 500Mb RAM. Network was 100 GigaEthernet.

The software was tested by computing the solution diagram of a Reverse Flow Reactor (Russo et al., 2002). Details of the construction of the reduced dynamical system are described in Russo *et al* 2002, The system is finally discretised to 36 ordinary differential equations. A typical solution diagram is represented in Fig 2. Complex dynamical regimes are encountered, which require extensive computations. Periodic symmetric and asymmetric regimes, quasi periodic asymmetric regimes and chaotic regimes are detected. Stability computation of each point of the line needs several estimation of Jacobian matrix of the discrete system. Comparison between a homogeneous cluster and a shared memory 4-processor machine, synchronous approach, shows that speedup in

the two cases is similar and very large as seen in Figure 3, left. The asynchronous approach proves effective when used on heterogeneous clusters (Figure 3, right), in that available computing resources are fully exploited for the minimum elapsed time.

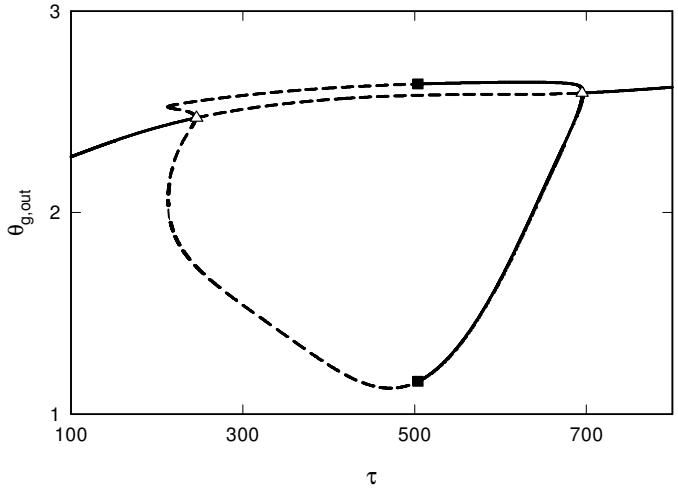
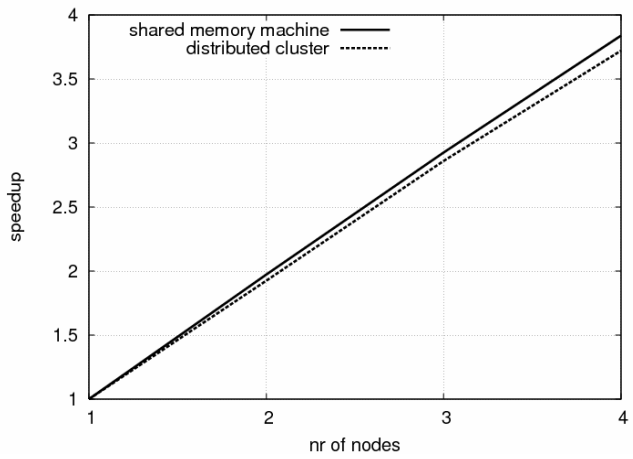


Figure 2 – Solution diagram, RFR reactor (Russo *et al*, 2002), switch time τ as bifurcation parameter.



Processor	State	No. of jobs	Time, s
1	Free	528	105.4
2	Free	529	
3	Free	527	
1	Free	556	131.0
2	Free	529	
3	Busy	499	
1	Free	590	151.1
2	Busy	495	
3	Busy	499	

Figure 3 - Left: Speedup comparison, RFR model, distributed cluster and shared memory machine. Right: demonstration of functionality of the asynchronous approach

References

Amdahl, G. M. (1967), 'Validity of the single-processor approach to achieving large scale computing capabilities', *Proceedings of AFIPS Conference*, pp. 483–485.

Continillo G., Grabski A., Mancusi E., Russo L. (2006). Bifurcation analysis of a periodically forced pair of tubular catalytic combustors, Combustion Theory and Modelling Vol. 10, No. 6, December 2006, 1023–1035

Doedel EJ, Champneys AR, Fairgrieve TF, Kuznetsov YA, Sanstede B, Wang X. (1997) 'AUTO97: Continuation and bifurcation software for ordinary differential equations', Technical Report, CML, Concordia Univ.

Russo, L., Continillo, G., Maffettone, P.L., Mancusi, E., and Crescitelli, S. (2002). Intermittencies and symmetry in a reverse-flow chemical reactor, Dynamics Days Europe 2002, Heidelberg, Germany, July 15–19. First Prize poster award: http://www.iwr.uni-heidelberg.de/conferences/dd02/award/index.html