Parallel Solution of Cascaded ODE Systems Applied to ^{13}C-Labeling Experiments

Katharina Nöh and Wolfgang Wiechert

University of Siegen
Paul-Bonatz-Str. 9-11, D-57068 Siegen, Germany
{noeh,wiechert}@simtec.mb.uni-siegen.de
http://www.simtec.mb.uni-siegen.de

Abstract. In the rapidly growing field of computational methods within the discipline of Metabolic Engineering, the simulation of *instationary* ^{13}C labeling experiments is a new research focus. The underlying mathematical model is a high-dimensional cascaded system of differential equations and must be exploited to obtain efficient simulation algorithms. Additionally the sensitivity matrices of the system have to be computed as efficient as possible. For this purpose several ways for parallel implementation are introduced, compared and discussed.

1 Cascaded Systems

In this contribution high-dimensional cascaded ODE systems are considered which have the general form

$$ {}^{i}\dot{\mathbf{y}} = \mathbf{f}\left({}^{0}\mathbf{y}, {}^{1}\mathbf{y}, \ldots, {}^{i-1}\mathbf{y}, \mathbf{p}\right), \quad i = 1(1)m $$

with initial values ${}^{i}\mathbf{y}\left(t_0\right) = {}^{i}\mathbf{y}_0$ and parameters $\mathbf{p}$. Herein the upper left index of the vectors ${}^{i}\mathbf{y}$ specifies the stage of the cascade. To demonstrate the general methods developed in this paper a class of biotechnological examples is taken that arise in the framework of Metabolic Flux Analysis (MFA) [1-3].
In case of metabolic stationarity (constant metabolic fluxes $\mathbf{v}$ and pool sizes $\mathbf{X}$) the detailed dynamics of an isotopically instationary ^{13}C labeling experiment is described by a so called cumomer vector $\mathbf{x}$ [4] and has the cascaded form

$$ \operatorname{diag}({}^{i}\mathbf{X}) \cdot {}^{i}\dot{\mathbf{x}} = {}^{i}\mathbf{A}(\mathbf{v}) \ \cdot \ {}^{i}\mathbf{x} + {}^{i}\mathbf{b}(\mathbf{v}, \mathbf{x}^{inp}, {}^{0}\mathbf{x}, {}^{1}\mathbf{x}, \ldots, {}^{i-1}\mathbf{x}), \quad i = 1(1)m \qquad (1) $$

with ${}^{0}\mathbf{x} = \mathbf{1}$ and for some given initial values ${}^{i}\mathbf{x}\left(0\right) = {}^{i}\mathbf{x}_0$ and known input labeling $\mathbf{x}^{inp}$ [3]. Here ${}^{i}\dot{\mathbf{x}}$ depends linearly on ${}^{i}\mathbf{x}$ but nonlinearly on ${}^{0}\mathbf{x}, {}^{1}\mathbf{x}, \ldots, {}^{i-1}\mathbf{x}$. The equation ${}^{0}\mathbf{x} = \mathbf{1}$ is a conservation law which is exactly fulfilled. In [3] it is shown that for a reasonable metabolic network under weak conditions $(\mathbf{v}, \mathbf{X} > 0)$ the non-singularity of ${}^{1}\mathbf{A}$ is equivalent to the global stability of the complete cascaded system (1). In the context of MFA the ODE (1) can have a very high dimension and moreover it constitutes an inverse problem for the fluxes $\mathbf{v}$. Accordingly, the cascade (1) has to be solved repeatedly in the course of a parameter fitting procedure for flux determination. An efficient and aligned ODE solver thus is desirable because a standard "black-box" solver will cause tremendous computational costs.

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3037, pp. 594–597, 2004.

Sensitivity Equations: Because the biological system and its measurements are usually rather noisy a detailed statistical analysis for parameter fits is always required. For that purpose the time dependent sensitivity matrices have to be computed by another cascaded system

$$\frac{d}{dt}\frac{\partial^i \mathbf{y}}{\partial \mathbf{p}} = \sum_{k=1}^{i-1} \frac{\partial \mathbf{f}}{\partial^k \mathbf{y}} \cdot \frac{\partial^k \mathbf{y}}{\partial \mathbf{p}} + \frac{\partial \mathbf{f}}{\partial \mathbf{p}}, \quad \frac{\partial^i \mathbf{y}}{\partial \mathbf{p}}(t_0) = 0, \quad i = 1(1)m. \tag{2}$$

These systems take the main part of computational time and the way how the sensitivity matrices are computed has a large impact on the overall performance of the method. Each column of the matrices $\partial^i \mathbf{y}/\partial \mathbf{p}$ corresponds to one parameter. Consequently, (2) has a high dimension of dim $\mathbf{y} \cdot$ dim $\mathbf{p}$. It is well known that the sensitivity ODEs (2) in principle have the same form as the systems ODE (1) and hence have the same numerical stability characteristics.

Example Systems: As a typical example in MFA the amino acid producer *C. glutamicum* may serve. Simulating an experiment with this bacterium with (1) results in 5608 equations with 11 stages. The *C. glutamicum* network has a total of 244 parameters and so (1)+(2) has the overall dimension 1,368,352. In this contribution one important part of the network is taken as an example: the so called PPP network of cumomer dimension 522 with 214 parameters.

2 Implementations of the Cascaded Systems

Several ways for implementation are proposed in this section. In the first instance the cascade (1) is considered.

Implementation ('s'): For a first (serial) simulation the solution of (1) is implemented on a single processor system (AMD Athlon 1800+). The explicit high order Runge-Kutta method DOPRI8 [5] with embedded step size control and an accurate dense output is chosen. If the system is regarded to be stiff the ODE solver is changed to a linear singly diagonally implicit method of order 4. One simulation run of (1) is carried out until the stationary state of the system is reached. DOPRI8 needs 10.43 s without and 16.59 s with an inner stage interpolation. An absolute accuracy of 10^{-4} resp. 10^{-7} is reached which both satisfies the accuracy requirements.
Now the sensitivity system (2) is added. There are two main methods to perform a sensitivity analysis numerically. On the one hand the sensitivity matrices of an ODE can be deployed analytically by integrating the variational equation (2), where the complex derivative matrices are given explicitly. On the other hand the derivatives of the solution in stage i can be computed by a standard differential quotient (DQ). This has some well known obstacles: The discretization error must be very small to meet the desired sensitivity accuracy within a given error tolerance. Furthermore there is no accuracy control of the sensitivities. However, this approach is very simple to implement.

The so arising computational effort is very high and it should be shortened by using multiple processors. All programs are tested on a cluster / SPMD architecture with 10 dual processor PCs (1.5 GB main memory each, 1000 MBps Ethernet interface). This contribution will concentrate on *parallelism across the problem* [6] because this approach seems to be the most promising one.

Functional Parallelism ('p'): The most apparent type of functional parallelism is pipelining: give each processor its own unique task and allow the data to "flow" through the pipeline in only one direction. As one processor finishes its calculation, it passes the solution along to the next processor and receives another solution from the previous processor.
Clearly, this is a "natural" parallelism for both the simulation problem (1) and the sensitivity equations (2). Each processor calculates one stage and the time shifts between the stages are variable. A rather good speedup is expected, because the last stage should theoretically have completed only a short time after the first stage.

Trivial Parallelism ('t'): This method does not involve parallelizing the code at all, rather the sequential code is run on a number of processors in parallel, with each processor having its own variant of the problem. The processors are working independently and there is no communication between the processes – except for the very end.
This method fits perfectly to the requirements of a sensitivity analysis using differential quotients as numerical approximation. Each processor gets a slightly varied parameter set and makes a simulation run. At the end all results are collected and the sensitivity matrices are given by a standard differential equation formula (DQ).

Decomposition of Sensitivity Matrices ('d'): Instead of dividing tasks between processors, each processor works on its own section of a dataset of parameters. This parameter set is initially divided between the processors, the calculations are carried out using the data, and the results are collected at the end of the computational process. Thereby the data can be easily distributed in such a way that the processor load is well balanced.
This seems to be the favorite method for the variational differential equation system (2) provided the solution vector $\mathbf{x}$ from (1) is given. It can be implemented by solving the cascade (1) additionally on *each* processor. Again no communication overhead is needed until the very end.

3 First Results

Each of these methods exploits a special feature of the systems to solve and can be combined with others. The following table shows the computational time for reasonable combinations of the parallel approaches. For comparability the ODE solver here is restricted to a constant step size sequence (1000 steps).

eqns.	(solution, sensitivity) method					
	(s,s)	(s,t)	(s,d)	(p,p)	(p,t)	(p,d)
(1, 2)	1021.51s	–	205.82s	534.88s	–	[1]
(1, DQ)	1591.87s	279.86s	–	–	381.98s	–

[1]: The implementation is currently under work.

The trivial parallelism method 't' gives a good speedup because it is not limited by a permanent communication overhead. This is also the case for the decomposition method 'd' even though the cascade (1) has to be computed multiple times. Generally, the expected time ranges are achieved for both methods.
Unlikely, this does not apply for the functional parallelism 'p'. Under precise consideration this method has three main limits: First of all there are idle processors at the beginning and the end of the computation. A more serious limit is the problem of load balancing. The stages have not the same dimensions and these differences are amplified by the sensitivities. So the processors are in general not load balanced and this will cause a bottleneck in the pipeline. To overcome this problem an additional level of parallel tasks has to be introduced in future versions of the code. The third limitation is given by the network communication since the whole solution must flow through all processors.

4 Conclusions and Outlook

After the special cascaded structure of the mathematical model of a CLE is introduced, possible sequential and parallel approaches are figured out. Already a moderately sized metabolic network shows that a sequential implementation is too slow for a repeated call by a parameter fitting procedure. Generally speaking only these parallel approaches are competitive which run without too much network communication because all benefits gained by multiple processors are discarded through it. It would also be interesting to test the scalability of the methods on clusters with more knots and to confirm the results for more extended networks like *C. glutamicum*.

References

1. Van Winden, W.: ^{13}C-Labelling Technique for Metabolic Network and Flux Analysis. PhD. Thesis, Delft University of Technology, 2002.
2. El Massoudi M., Drysch, A., Spelthahn, J., de Graaf, A.A., Takors, R.: Production Process Monitoring by Serial Mapping of Microbial Carbon Flux Distributions Using a Novel Sensor Reactor Approach. Metab. Eng. **5** (2003), pp. 86–95.
3. Wiechert, W., Wurzel, M.: Metabolic isotopomer labeling systems. Part I: Global dynamic behavior. Math. Biosciences **169** (2001), pp. 173–205.
4. Wiechert, W.: ^{13}C Metabolic Flux Analysis. Metab. Eng. **3** (2001), pp. 195–206.
5. Hairer, E., Nørsett, S.P., Wanner G.: Solving ordinary differential equations I + II. 2^{nd} ed., Springer, 2000 resp. 1996.
6. Burrage, K.: Parallel and Sequential Methods for Ordinary Differential Equations. Oxford Science Publications, Clarendon Press, 1995.