Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

A parallel function evaluation approach for solution to large-scale equation-oriented models

Yannan Ma[a], Zhijiang Shao[a], Xi Chen[a,*], Lorenz T. Biegler[b]

[a] State Key Laboratory of Industrial Control Technology, College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China
[b] Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA

A R T I C L E I N F O

Article history:
Received 15 January 2016
Received in revised form 6 June 2016
Accepted 12 July 2016
Available online 14 July 2016

Keywords:
Equation-oriented model
Function evaluation
Parallel computing
GPU
Multi-core processor

A B S T R A C T

The equation-oriented (EO) approach is widely used for process simulation and optimization. Nevertheless, large-scale EO models consist of a huge number of nonlinear equations and make the solution procedure a challenging and time-consuming task. For most gradient-based numerical algorithms, function evaluations are the dominant step during the solution procedure. Here, a parallel computation method is developed for function evaluations within EO optimization strategies. After dividing the equations into several groups, function evaluations are calculated by using multiple threads on a parallel hardware platform simultaneously. Theoretical analysis for the speedup ratio is conducted. The implementation of the proposed method on a multi-core processor platform as well as a graphics processing unit (GPU) platform is then presented with several case studies. Numerical results are compared and discussed to show that the multi-core processor implementation has good computational performance, whereas the GPU implementation only achieves computational acceleration under relatively specific conditions.

1. Introduction

Process simulation and optimization is an essential task to analyze the process productivity, product quality, and plant performance in a reliable, environmental, and economical way. Due to the complexity of the physicochemical phenomena in reactions and the nonlinearities arising from their mathematical models, the simulation and optimization task in chemical processes remains complicated and challenging.

Both in industry and academia, two well-established approaches are widely applied to design and optimize systems, especially for large-scale systems. The first approach is an object-oriented approach, where the solution procedure is tied to the model. For steady-state process simulation and optimization, this usually appears as the sequential modular (SM) approach, where process modules are connected by streams according to process topology and solved iteratively by “tearing” the recycle streams, in which an optimizer is coupled with a simulator. In this approach, modules are relatively easy to construct and generalize because of the tailored procedures for the blocks. Nevertheless, the module models need to be solved repeatedly and parallelization is strongly influenced by flowsheet structure. Also, caution is required to prevent potential intermediate module failures during simulation and optimization. Another drawback is that exact derivatives for the implicit unit models are not directly available, leading to inaccuracy and round-off errors with the replacement by finite differences.

The second approach is the equation-oriented (EO) approach. All equations and variables that describe the unit operations are simultaneously solved. Compared to the SM mode, this approach requires more efforts on modeling and initialization by the user. However in the EO mode, exact derivatives are directly available and round-off errors in gradients are avoided. Thus, EO formulations directly exploit advances in large-scale nonlinear programming (NLP) algorithms (Dowling and Biegler, 2015). Generally, it is preferred for complex flowsheets with many nested recycle loops and implicit design specifications.

Numerous research efforts have been made to improve EO methods to deal with process simulation and optimization. To conduct kinetic parameter estimation and productivity optimization on an industrial high-density polyethylene (HDPE) slurry process with specified molecular weight distribution (MWD), Zhang et al. (2013) established and solved a complete EO model, including rigorous kinetic mechanism and thermodynamics. Pattison and Baldea (2015) developed a robust EO modeling and optimization framework to compute stream temperatures for multi-stream heat exchangers. Huang et al. (2009) applied the advanced step nonlinear model predictive control (asNMPC) on a rigorous dynamic EO

* Corresponding author.
E-mail address: xichen@iipc.zju.edu.cn (X. Chen).

http://dx.doi.org/10.1016/j.compchemeng.2016.07.015
0098-1354/© 2016 Elsevier Ltd. All rights reserved.

model of air separation units to reduce the online computational delay. Fu et al. (2015) conducted process analysis and optimization of a cryogenic air separation unit using a complete EO model to achieve satisfactory convergence performance. Kamath et al. (2010) proposed a novel general EO approach for selecting the appropriate root of the cubic equation of state by incorporating derivative constraints specific to the desired phase. Although the EO approach has distinct advantages in simultaneous optimization and convergence efficiency, its solution procedure for large-scale models could still require considerable computational effort. The time consumption is always considered as a key performance indicator for real-time optimization in operations. For example, a rapid and efficient search is required to generate a new operation point in the circumstance of frequent fluctuations in product demand. Compared with conventional serial computing, parallel computing takes more effort and time for its algorithm design, but achieves much more significant acceleration effect.

Parallel computing requires multiple tasks to perform simultaneously (Almasi and Gottlieb, 1989). As the improvement of single-processor computing has slowed, an increasing amount of work is being conducted to take full advantage of parallel computing in industrial engineering problems. Weng et al. (2015) implemented a multi-thread parallel strategy to compute the MWD of the multisite free-radical polymerization. Velez and Maravelias (2013) developed a parallel branch-and-bound algorithm for a chemical production scheduling problem using a discrete-time model. Fesko (2012) considered a parallel algorithm and achieved a significant gain in speed for computing an initial approximation to the optimal control problem. For the graphics processing unit (GPU) computing platform, the large-scale simulation and optimization of chemical process is a recent application area. Zhang et al. (2011) developed a parallel implementation of the algorithm that utilizes modern GPU technology for Monte-Carlo-based risk assessment in CO_2 sequestration. Lu et al. (2014) proposed a coarse-grained discrete particle method to simulate hydrodynamics of gas–solid flows, which takes full advantage of CPU-GPU hybrid supercomputing. Klimeš and Štětina (2015) presented a rapid GPU-based heat transfer and solidification model for continuous steel casting, the performance of which makes it suitable for real time applications in casting control and optimization. Most researchers develop parallel computation methods based on a specific chemical process itself. However, few papers exploit the structure of general-purpose numerical algorithms and parallelize the solution procedure of diverse large-scale EO models. On the other hand, there has been little specific effort in GPU-based general-purpose parallel solution strategies, due to the fact that NLP algorithms do not have enough synchronized and independent execution thread paths, which are favored by the parallel computation on GPUs. Nevertheless, GPU is especially effective in algorithms that are characterized by a large number of floating point operations per memory data transfer.

In this study, we propose a parallel computation method to solve large-scale EO models. Under the framework of Successive-Quadratic-Programming (SQP)-based algorithm, this method allows its function evaluation task to be executed using parallel threads that communicate with the main solution procedure. Its implementation is developed with both multi-core processor as well as GPU technology. A number of case studies are presented to illustrate the performance of the proposed method on both platforms. Finally, guidelines are given for selecting applicable platforms based on the model characteristics.

2. Parallel computation method to function evaluation

According to the EO formulation, detailed process steady-state and dynamic behaviors can be described by a rich set of nonlinear models. Associated with a scalar quantitative performance measures to be minimized or maximized, a typical NLP problem is developed. The general form of an NLP problem can be given as:

$$\begin{aligned} &\min_x f(x) \\ s.t.\quad & g(x) \le 0 \\ & h(x) = 0 \end{aligned} \tag{1}$$

where x is an n-dimensional vector of continuous variables; $f(x)$ is a scalar objective function; $g(x)$ is the set of inequality constraint functions; $h(x)$ is the set of equality constraint functions. We assume that $f(x)$, $g(x)$ and $h(x)$ all have continuous first and second derivatives. Great progress in finding NLP solutions makes the ability to solve large-scale process optimization models cheaply. In particular, gradient-based NLP algorithms have made the formulation and solution of large-scale EO models accessible to a much wider user base (Biegler, 2010).

2.1. Function evaluation in gradient-based NLP methods

For constrained optimization, the basic idea of gradient-based NLP methods is to consider a modified equation set of the Karush–Kuhn–Tucker (KKT) conditions from the original models (Biegler et al., 1997). Then, these equations can be solved with a Newton-based method, with a set of linear equations solved at each iteration to define the Newton step. The Newton step from the iterate (x^k, μ^k, λ^k) can be given by:

$$\begin{bmatrix} B_k & \nabla g_A & \nabla h \\ \nabla g_A^T & 0 & 0 \\ \nabla h^T & 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ \Delta\mu \\ \Delta\lambda \end{bmatrix} = - \begin{bmatrix} \nabla_x L(x^k, \mu^k, \lambda^k) \\ g_A(x^k) \\ h(x^k) \end{bmatrix} \tag{2}$$

where L(x,μ,λ) is the Lagrangian function of (1) with the Lagrange multipliers μ and λ; B_k is the Hessian of the Lagrange function $\nabla_{xx}L$, or its approximation; $g_A(x)$ is made up of the active inequality constraints if the optimal active set can be identified correctly in advance; d is the predicted search direction; $\Delta\mu = \mu^{k+1} - \mu^k$, $\Delta\lambda = \lambda^{k+1} - \lambda^k$. The next point $x^{k+1} = x^k + \alpha d$, where the step size α is determined so as to reduce a penalty function that balances the decrease of the objective function and the violation of the constraints.

Typically, we need to perform a function evaluation, i.e., to evaluate the values of objective and constraint functions, and their first derivatives and second derivatives (or their approximations) for the coefficients in Eq. (2) of Newton step at the current iterate (x^k,μ^k,λ^k). After the function evaluation step, the updating procedure of a specific NLP algorithm is conducted to generate a new iterative point.

To further illustrate the function evaluation, we take the Sequential Quadratic Programming (SQP)-type method as an example. The SQP method has emerged as an efficient algorithm for process optimization. It inherits fast convergence properties from Newton's method and can be tailored to a wide variety of problem structures (Biegler et al., 1997). The basic idea of the SQP method is to model (1) at the current iteration point x^k by a quadratic programming subproblem:

$$\begin{aligned} &\min_d \nabla f(x^k)^T d + \frac{1}{2} d^T B_k d \\ s.t.\quad & h(x^k) + \nabla h(x^k)^T d = 0 \\ & g(x^k) + \nabla g(x^k)^T d \le 0 \end{aligned} \tag{3}$$

The solution of the quadratic program (3) is equivalent to the Newton step in Eq. (2). By solving this subproblem, a new point