PERFORMANCE EVALUATION OF NONLINEAR SYSTEM ARISING FROM LENGYEL-EPSTEIN MODEL

Maria A. Kartawidjaja
Fakultas Teknik Unika Atma Jaya
maria.kw@atmajaya.ac.id

ABSTRAK
Suatu cara yang umum untuk menyelesaikan sistem nonlinear berukuran besar adalah dengan menggunakan metode Newton. Metode Newton mentransformasikan suatu sistem nonlinear ke suatu sistem linear, yang kemudian dapat diselesaikan dengan suatu metode langsung atau metode iteratif. Penyelesaian suatu sistem nonlinear yang berukuran besar membutuhkan waktu komputasi yang sangat panjang, dan karenanya komputasi paralel merupakan suatu alternatif yang menarik. Pada penelitian ini suatu sistem persamaan nonlinear yang terbentuk dari solusi persamaan diferensial biasa (ODE) dari model Lengyel-Epstein diselesaikan dengan menerapkan paralelisisasi across the method, dan sistem persamaan linear yang terbentuk kemudian diselesaikan dengan paralelisisasi across the system. Penguji dilakukan pada sekelompok komputer pribadi (PC) dengan lingkup paralel yang diemulasikan dengan menggunakan perangkat lunak Parallel Virtual Machine (PVM). Dari hasil pengujian diperoleh kesimpulan bahwa penerapan paralelisisasi dua tahap ini menghasilkan peningkatan kinerja seiring dengan peningkatan jumlah prosesor, kecuali untuk data yang kecil.

Kata Kunci: Kinerja, Nonlinear, Paralelisisasi, PVM

1. Introduction
In many areas of applications, such as mathematics, chemistry and physics, one will encounter a classical problem arising from solving nonlinear equations of the following form:

\[ f(x) = 0 \]  

where \( f: \mathbb{R}^n \rightarrow \mathbb{R}^n \). A common iterative method to solve the arising nonlinear system assuming \( f \) is a continuously differentiable function is provided by Newton method. This method transforms a nonlinear system into a linear system that is solved subsequently using a direct or iterative linear solver. Newton method might require a large computation time depending on stiffness of the system, and therefore parallelization becomes a reasonable approach.

To solve the nonlinear system, Newton approach takes an initial guess, \( x_0 \) for instance, and refines this value assuming \( f(x_k) \) is locally linear. The iteration is terminated when the relative nonlinear residual or the value of \( \| f(x_k) \| \) is small.

This article discusses performance of a two level parallelization, across the method to solve the nonlinear system by using Newton method, and across the system to solve the arising linear system by using an iterative scheme, Generalized Minimal Residual (GMRES) method. The experiment was performed on a cluster of PCs using a test problem derived from Lengyel-Epstein model.

2. Parallel Performance
The aim of using parallel computation is to obtain a better performance than that of a sequential machine can offer. As in sequential system, most performance issues in parallel system can be addressed by programming or architectural techniques or by both. The focus of this research is on performance issues addressed by programming techniques. There are various metrics to evaluate performance of parallel systems, such as run time, speedup, efficiency, and cost. Among those metrics, speedup is the most frequently used performance metric\(^{1}\).

Speedup is a dimensionless quantity that describes the performance gain of parallel processing vs. serial processing. A measure of relative performance between a parallel code running on a single processor to that running on \( p \) processors is termed as relative speedup and is written as:

\[ Sp = \frac{T_1}{T_p} \]  

where \( T_1 \) and \( T_p \) are the execution time of the parallel code on a single and multiprocessors respectively. The absolute speedup \( T_1 \) denotes the execution time of the best sequential code.

The maximum speedup of a parallel system using \( p \) processors is \( p \). This condition is called linear speedup. However, in some cases a superlinear speedup, a condition where \( Sp > p \), can occur. Superlinear speedup does not reflect an actual performance gain; it is usually caused by cache effect or some unique feature of the architecture that favors the parallel code, such as extra memory in multiprocessor system\(^{2, 3}\).

There are two distinct programming models in parallel computing platform that have very different communication architectures; those models are shared memory model and message-passing model. A shared memory model employs a
single address space, meaning that there is a global memory that can be accessed by each processor in the system. In contrary, each processor in a message-passing model has its own local memory, and therefore communication between processors is an explicit I/O operation and obviously would induce a significant overhead to the system.

A common approach in programming on distributed memory system is to utilize existing sequential high level language, such as C or FORTRAN, and provide library routines of external procedures for message-passing. Two popular message-passing libraries, Parallel Virtual Machine (PVM) and Message Passing Interface (MPI) have been widely used.

3. Parallel Virtual Machine (PVM)
PVM is a software that makes a collection of computers appear as one large virtual machine\(^4\). PVM can be used on homogeneous and heterogeneous platforms. It handles transparently all message routing, data conversion and task scheduling across a network of incompatible computers. Due to its ubiquitous nature, especially the virtual machine concept, its simple but complete programming interface, and partly due to its availability at no charge, PVM has been widely accepted in high performance computing community.

PVM system is composed of two parts. The first part is a daemon process call pvmd that resides in all computers forming the virtual machine. The second part is the library of PVM interface routines that contains a complete collection of primitives that are required for cooperating processes.

PVM programs are generally organized in a master-slave arrangement, where a single process, master process, is first executed, and all other processes, slave processes, are created by calling a spawning routine from master process. Communication between processes in PVM is performed by message passing using PVM library routines. The library routines are written in C or FORTRAN programming languages. PVM allows the creation of any processes, independent on the number of processors. Each process is identified by a task ID. The process will be mapped on to processors automatically unless overridden by the user.

4. Problem Definition
Lengyel-Epstein model is a mathematical model derived from chemical reaction Chlorite Iodide Malonic Acid (CIMA) written in the form:\(^5\):

\[
\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} + a - u - \frac{4uv}{1 + u^2} = 0
\]

\[
\frac{\partial v}{\partial t} = \sigma \left[ c \frac{\partial^2 v}{\partial x^2} + b \left( u - \frac{uv}{1 + u^2} \right) \right] + \gamma
\]

(3)

The reactor domain \(\Omega\) is assumed bounded in \(\mathbb{R}^n\), with a smooth boundary \(\partial \Omega\). The parameters \(u\) and \(v\) denote the chemical concentration of the activator iodide and the inhibitor chlorite correspondingly, at time \(t > 0\) and at point \(x \in \Omega\). These parameters can either be constants or sinusoidal functions. The terms \(a\) and \(b\) are parameters related to the feed concentrations, \(c\) is the ratio of diffusion coefficients, and \(\sigma\) is a rescaling parameter that depends on the concentration of starch\(^5\). Those aforementioned parameters must have positive values\(^6\). Stiffness of the system is mainly determined by the rescaling parameter \(\sigma\).

5. Overview of the Relevant Numerical Concepts
5.1 Newton Method
In autonomous form the system of nonlinear equations is written as\(^7\):

\[
Y = e \otimes y_n + h(A \otimes I_n)F(Y)
\]

(4), where

\[
Y = \begin{pmatrix} Y_0^T, \ldots, Y_n^T \end{pmatrix} \in \mathbb{R}^{m\times n}, \quad Y_0 \in \mathbb{R}^m, \quad F(Y) = \begin{pmatrix} f(Y_0^T), \ldots, f(Y_n^T) \end{pmatrix}^T, \quad e = [1, \ldots, 1]^T, \quad A \in \mathbb{R}^{m\times n} \quad \text{and} \quad y \in \mathbb{R}^m. \quad I \quad \text{and} \quad A \quad \text{represent an identity matrix and a Runge-Kutta matrix respectively. The differential equation is expressed as:}
\]

\[
y_{n+1} = y_n + h(b^T \otimes I_n)F(Y)
\]

(5).

Using Newton iteration scheme results in the following linear system:

\[
(I_n \otimes I_m - hA \otimes J)x = G
\]

(6), where \(J\) is a Jacobian matrix and \(G\) is a vector derived from Equation (3) such as

\[
G = -Y + e \otimes y_n + h(A \otimes I_n)F(Y)
\]

(7).
It is apparent from Equation (6) and Equation (7) that the solution of nonlinear system requires the computation of Jacobian matrix at each iteration. The Jacobian matrix is usually computed by using finite-difference scheme rather than using analytical approach. In order to reduce the computational cost, the Jacobian matrix in Equation (6) is evaluated at a single point, at $y_0$ for instance, and the method is known as modified or inexact Newton method\cite{7,8}. Modified Newton method is commonly used for solving stiff ODE where the Jacobian matrix may not be updated for several integration steps\cite{9,10,11}. Convergence of Newton’s method is guaranteed if the starting point of the integration is sufficiently close to the solution and the Jacobian and the solution is nonsingular.

5.2 Generalized Minimal Residual (GMRES) Method
GMRES, proposed by Saad and Schultz in 1986\cite{12} is a widely used method for solving large non-symmetric linear systems. It is one of Krylov subspace methods that searches for a suitable approximation of the solution vector $x$ from a linear system by using the minimum residual approach at every iteration step. In this method, the solution vector $x_k$ is taken as the vector that minimizes $||b - Ax_k||_2$ over all $x_k$ with $(x_i - x_0)$ is $K_i(A;r_0)$, where $r_0 = b - Ax_0$ is the initial residual and $K_i(A;r_0)$ is the $k$-th Krylov subspace generated by the matrix $A$ and vector $r_0$. The correction $x_k$ with respect to $x_0$ minimizes the residual over this Krylov subspace, and hence the residual does not increase with the iteration.

GMRES has a nice property that it will never break down unless the approximate solution has been achieved\cite{13}. However because we have to store all the successive residual vectors, storage requirement becomes prohibitive and the construction of the projected system becomes increasingly complex as well. One way to alleviate this disadvantage is to use a restart strategy, which considerably reduces the amount of storage required. The restarted GMRES method restricts the Krylov subspace dimension to a predefined value $k$ and restarts the method after $k$ iterations using the last iterate result of $x_k$ as a new initial guess. This restarted version is called GMRES ($k$). GMRES uses the Arnoldi process to construct an orthonormal bases $V_k = [v_1, ..., v_k]$ for the Krylov subspace $K_i(A;r_0)$. The common implementation relies on using modified Gram-Schmidt orthogonalization due to its numerical stability.

6. Implementation Issues
Firstly, the Lengyel-Epstein model in Equation (3) is discretized into following equations:

\[
\begin{align*}
\frac{d}{dt}u_i &= \frac{1}{(\Delta t)}[u_{i-1} - 2u_i + u_{i+1}] + a - a \frac{4u_i v_i}{1 + u_i^2} \\
\frac{d}{dt}v_i &= \frac{\sigma c}{(\Delta t)}[v_{i-1} - 2v_i + v_{i+1}] + \sigma b u_i - \frac{\sigma b u_i v_i}{1 + u_i^2} \\
\Delta t &= 1/(N + 1) \\
x_i &= i \Delta x, \quad (1 \leq i \leq N)
\end{align*}
\]

The initial condition is taken as the following sinusoidal functions\cite{5}:

\[
\begin{align*}
u_i(0) &= 3 + \sin(x_i) \\
v_i(0) &= 10 + \cos(x_i)
\end{align*}
\]

In the experiment we used a four-stage Runge-Kutta matrix proposed by Iserles and Nørsett\cite{14} as depicted in Figure 1. This Runge-Kutta matrix has a block diagonal structure as revealed in Figure 1. For simplicity the block matrices are named as $A_1$ and $A_2$ correspondingly as given in Figure 2.

\[
\begin{array}{c|ccc}
\frac{3-\sqrt{3}}{6} & \frac{5}{12} & \frac{1-2\sqrt{3}}{12} & 0 \\
\frac{3+\sqrt{3}}{6} & \frac{1+2\sqrt{3}}{12} & \frac{5}{12} & 0 \\
\frac{3-\sqrt{3}}{6} & 0 & 0 & \frac{1}{2} \\
\frac{3+\sqrt{3}}{6} & 0 & 0 & \frac{1}{2} \\
\end{array}
\]

\[
\begin{array}{c|ccc}
\frac{3}{2} & \frac{3}{2} & -1 & -1
\end{array}
\]

Figure 1. Runge-Kutta Matrix
Within each Newton iteration, the Jacobian is computed using finite-difference approach. The resulting linear system of equations is then solved by GMRES. After each Newton iteration, the convergence is checked by a control parameter, measuring the relative nonlinear residual. The termination criterion is based on a mixed relative and absolute error tolerances given as:

$$\|g(y)\| \leq r_r \|G(Y_0)\| + r_a$$

(10), where $r_r$ and $r_a$ denotes relative and absolute error tolerances respectively. The combination of these error tolerances are frequently used in numerical methods for solving ODEs and DAEs[10, 11, 16, 17] as mentioned in[15]. If the Newton iteration does not converge, the time step is reduced and the process is repeated. If convergence is not achieved after a predefined maximum iteration, a very small step size is taken, a time step of $10^{-7}$ in this research.

7. Parallelization Strategy

As previously mentioned, the matrix used in the experiment consists of two completely decoupled blocks, $A_1$ and $A_2$ as given in Figure 2. Therefore, vector $Y$ in Equation (3) can also be viewed as two vectors of equal size of $\frac{1}{2}sm$, $Y_1$ and $Y_2$ for instance. Likewise vector $G$, vector $x$ and Jacobian matrix $J$ in Equation (5) as $G_1$ and $G_2$, $x_1$ and $x_2$, $J_1$ and $J_2$, respectively. When using two processors, each processor has one block of matrix $A$, one part of vector $Y$, one part of vector $G$, and the computation on each block can be carried out concurrently. If there are more processors, the resulting linear system is also solved in parallel.

In order to balance the workload among processors, the number of processors used in the parallel environment must be even. Since only eight processors are available in our working platform, we use two, four, six and eight processors in parallel, and each processor is mapped onto each processor. For clarity of discussion, we provide a working relationship of two, four, six processors in parallel as given in Figure 3, where $p_1$ denotes the master processor, and $p_2$, $p_3$, $p_4$, $p_5$, and $p_6$ denote slave processors. The eight processors interrelationship can be constructed analogously.

8. Numerical Experiments and Results

The experiment was performed on a cluster of six PCs, connected through a 100 MBit Ethernet. The processors are INTEL Pentium with similar characteristics, running on 4.3 GHz. The system performance is evaluated based on relative speedup. The test problem is a Lengyel-Epstein model with dimension ranging from 300 to 800. The stiffness of the system is obtained by taking the rescaling parameter $\sigma = 8 \cdot 10^8$. Other parameters are taken as follows: $a = 20$, $b = 1.2$, $c = 2$, $u_0(x) = 3 + \sin(x)$, and $v_0(x) = 10 + \cos(x)$. The execution time is given in Table 1.

![Figure 2. Block Diagonal Matrix](image)

![Figure 3. Processor Working Relationship](image)
Table 1. Execution Time of Nonlinear System For Lengyel Epstein Model

<table>
<thead>
<tr>
<th>Dimension</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_4$</th>
<th>$T_6$</th>
<th>$T_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>243</td>
<td>197</td>
<td>360</td>
<td>313</td>
<td>385</td>
</tr>
<tr>
<td>400</td>
<td>651</td>
<td>512</td>
<td>465</td>
<td>414</td>
<td>391</td>
</tr>
<tr>
<td>500</td>
<td>1684</td>
<td>1276</td>
<td>915</td>
<td>668</td>
<td>538</td>
</tr>
<tr>
<td>600</td>
<td>3301</td>
<td>2309</td>
<td>1423</td>
<td>1022</td>
<td>830</td>
</tr>
<tr>
<td>700</td>
<td>5941</td>
<td>3600</td>
<td>2014</td>
<td>1470</td>
<td>1235</td>
</tr>
<tr>
<td>800</td>
<td>7546</td>
<td>4146</td>
<td>2239</td>
<td>1666</td>
<td>1403</td>
</tr>
</tbody>
</table>

From Table 1 we notice the increasing number of processors decreases the execution time except for small matrix dimension, i.e. 300. The corresponding speedup is then calculated using Equation (2), and the result is given in Table 2.

Table 2. Speedup of Nonlinear System For Lengyel Epstein Model

<table>
<thead>
<tr>
<th>Dimension</th>
<th>$Sp_2$</th>
<th>$Sp_4$</th>
<th>$Sp_6$</th>
<th>$Sp_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>1.23</td>
<td>0.67</td>
<td>0.65</td>
<td>0.63</td>
</tr>
<tr>
<td>400</td>
<td>1.27</td>
<td>1.40</td>
<td>1.57</td>
<td>1.66</td>
</tr>
<tr>
<td>500</td>
<td>1.32</td>
<td>1.84</td>
<td>2.52</td>
<td>3.13</td>
</tr>
<tr>
<td>600</td>
<td>1.43</td>
<td>2.32</td>
<td>3.23</td>
<td>3.98</td>
</tr>
<tr>
<td>700</td>
<td>1.65</td>
<td>2.95</td>
<td>4.04</td>
<td>4.81</td>
</tr>
<tr>
<td>800</td>
<td>1.82</td>
<td>3.37</td>
<td>4.53</td>
<td>5.38</td>
</tr>
</tbody>
</table>

Table 2 reveals that the parallel nonlinear system is able to reached a maximum speedup of 1.82 by using two processors, 3.37 by using four processors, 4.53 by using six processors, and 5.38 by using eight processors. Our observation on a single processor reveals that for this solver approximately 50 to 70% of the overall execution time is spent for solving the arising linear system. This extensive time required to solve the linear system will cause in a better performance if more processors are available except for matrix with small dimension, i.e.300, where the performance will decrease because of the low computational work compared to the communication overhead.

9. Conclusion
From the experiment we conclude that using more processors will contribute to a better performance except for small matrix dimension. This is reasoned by the fact that a large percentage of time, approximately 50-70%, is required to solve the arising linear system, and hence parallelization inside the linear system (across the system) will contribute to a better performance than that already provided by the parallelization in solving the nonlinear system (across the method). In short, by applying two levels parallelization in solving the nonlinear system arising from Lengyel-Epstein model, across the system and across the method, a notable performance in terms of speedup can be attained excluding small data size.

10. Limitation of Research
This research was performed on a cluster of PCs connected through an Ethernet bus. It is apparent that in bus based environment, communication is a major bottleneck, and thus the performance of the parallel system is bounded by the communication overhead. It is interesting to investigate to what extent one can push the performance of this parallel system if the program is executed on parallel system with a better communication architecture, an environment with multistage interconnection for instance. In addition, one can also investigate the optimal number of processors that will give a maximum performance gain of this parallel system.

Bibliography


