The Computation of State Variable Model in Software Test Process Using Alternating Group Explicit Iterative Algorithm

Praveen R Srivastava 1,+, Navnit Jha 2 and G Raghurama 3
1 Computer science and Information Systems Group, Birla Institute of Technology & Science Pilani-333 031, India
2 Mathematics Group, Birla Institute of Technology & Science Pilani-333 031, India
3 Electrical and Electronics Group, Birla Institute of Technology & Science Pilani-333 031, India

(Received November 23, 2008, accepted February 4, 2009)

Abstract. The aim of this paper is to construct a parallel algorithm for the model developed in software testing process. The model treats software testing as a control problem. The software under test serves as a controlled object that is modeled as second order equation. The ordinary simulation to such type of model deteriorates in the vicinity of quality, efforts and complexity. The new approach based on parallel alternating group explicit algorithm shows superiority over corresponding sequential algorithm. Computational results are provided to illustrate the viability of the proposed technique.

Keywords: Software testing, State variable, Feedback control, Software process, AGE method, RMSE.

1. Introduction

In this piece of work, we consider the modeling and control of software testing process. Software testing is classified into different category such as structural, functional or random. The internal description of software to generate test cases categories as structural software testing. Many software testing methods are available including software reliability engineering testing, transaction flow testing, equivalence class partition based testing, control flow testing etc. A testing strategy determines what test case should be selected and when software testing should be stopped. The software testing process can be characterized as a feedback system through control theory in order to regulate the testing process. We present a test model based on error reduction as the time increases. Earlier Cangussu et. al. [1, 2] has discussed similar model. However, the model fails to explain the behavior of system when number of software defects are inversely related with time as is frequent case. The model numerically leads to a tridiagonal system. The problem of actually determining the solution of such tridiagonal system may be time consuming, particularly when, as is frequently the case, the number of equations is large. Since we need to solve large system of equations, the iterative algorithms are frequently used. The proper choice of the iterative algorithm to be used has a great influence on the amount of computational effort required to solve a given problem. With slowly converging iterative algorithm, the amount of time required may be so large, even with a very fast computer, as to make the solution of problem impractical. Hence the resort to parallel algorithmic approach is imperative.

2. Development of Test Process Model

The software testing process involves development of second order model and is appropriate to capture the essential nature of testing phase. The model is used to estimate the fault in software. During the testing process errors are found and removed. Following parameters and variables are defined as

\[ r(t) \]: Number of software defect remains at time \( t \)

\[ s_c \]: Software complexity

\[ \omega_f \]: Work force

* Corresponding author. E-mail address: praveenrsrivastava@gmail.com

Published by World Academic Press, World Academic Union
\( \gamma \): Constant characterizing overall quality of testing methods

\( e_{et} \): Effective test effort

\( e_r \): Error reduction resistance

\( e_n \): The net effort applied

Using these variables and parameters, we have certain assumptions:

The magnitudes of the rate at which the remaining errors are decreasing is proportional to the net applied effort and are inversely proportional to the time.

\[
 r'' = \frac{e_n}{t} \quad (2.1)
\]

The magnitude of the rate at which the remaining errors are decreasing is inversely proportional software complexity (see Cangussu [4]).

\[
 r'' = \frac{1}{s_c} \quad (2.2)
\]

The magnitude of the effective test effort is proportional to the product of applied work force and the number of remaining errors \( \zeta \).

\[
 e_{et} = r\zeta \psi_f \quad (2.3)
\]

The error reduction resistance is proportional to the error reduction velocity and inversely proportional to the quality of software testing \( \xi \).

\[
 e_r = \frac{\zeta}{\gamma} r' \quad (2.4)
\]

Using equations (2.1) - (2.4) and carrying necessary algebra leads to the following model

\[
 r'' = \frac{\zeta}{r t s_c} r' + \frac{\zeta \psi_f}{t s_c} r + \frac{F_d}{s_c}, \quad t \geq 0
\]

If \( F_d \) denotes the disturbance during software testing. Then

\[
 r'' = \frac{\zeta}{r t s_c} r' + \frac{\zeta \psi_f}{t s_c} r + \frac{F_d}{s_c}, \quad t \geq 0
\]

The equation (2.6) is written as

\[
 r'' = d(t)r' + e(t)r + g(t) = \psi\left(r', r, t\right) \quad (2.7)
\]

where \( d(t) = \frac{\zeta}{r t s_c}, e(t) = \frac{\zeta \psi_f}{t s_c}, g(t) = \frac{F_d}{s_c} \).

The solution to equation (2.7), usually deteriorates in case of low software quality (\( \gamma \to 0 \)) and/or software complexity (\( s_c \)). Difficulties were experienced in the past for the solution of (2.7) in the vicinity of singularity. We overcome these situations by modifying our method based on Taylor’s expansion (see Ref. [6]).

Let us define

\[
 \bar{r}_k = \frac{r_{k+1} - r_{k-1}}{2h}, \quad \bar{r}_{k \pm 1} = \left( \pm 3r_{k \pm 1} \mp 4r_k \pm r_{k \mp 1} \right)/(2h)
\]

\[
 \bar{G}_{k \pm 1} = \psi\left(t_{k \pm 1}, r_{k \pm 1}, r_{k \pm 1}\right), \quad \bar{r}_k = r_k - \frac{h}{20} \left( \bar{G}_{k+1} - \bar{G}_{k-1} \right), \quad \bar{G}_k = \psi\left(t_k, r_k, \bar{r}_k\right)
\]

Then the fourth order finite difference replacement of (2.7) is

\[
r_{k+1} - 2r_k + r_{k-1} = \frac{h^2}{12} \left[ \bar{G}_{k+1} + \bar{G}_{k-1} + 10 \bar{G}_k \right], \quad k = 1(1)N \quad (2.8)
\]
where \( r_k \) denote the function satisfying the difference equation at the grids \( t_k = kh, \ k = 0(1)N + 1 \). The difference scheme (2.8) is of fourth order accurate for the numerical treatment of (2.7). We need the following approximations in order to establish the singular free recurrence relation.

\[
d_{k±1}=d_k ± 2h \d_k + \frac{h^2}{2}d_k ± O(h^3)
\]

(2.9a)

\[
e_{k±1}=e_k ± h\e_k + \frac{h^2}{2}e_k ± O(h^3)
\]

(2.9b)

\[
g_{k±1}=g_k ± 2h \g_k + \frac{h^2}{2}g_k ± O(h^3)
\]

(2.9c)

Substituting the approximations (2.9) into equation (2.8) and neglecting the higher order terms, we get a linear difference equation of the form

\[
a_k r_{k-1} + 2b_k r_k + c_k r_{k+1} = RH_k , \ k = 1(1)N
\]

(2.10)

3. The Alternating Group Explicit Method

The main concern to this section is to present an efficient algorithm based on parallel approach to compute the 3-term recurrence (2.10) developed for Test process model. The linear difference equations (2.10) can be in general expressed as

\[
a_k r_{k-1} + 2b_k r_k + c_k r_{k+1} = RH_k , \ k = 1(1)N
\]

(3.1)

The difference equation (3.1) in matrix notation is

\[
Ar = R
\]

(3.2)

where

\[
A = \begin{bmatrix}
2b_1 & c_1 \\
& \ddots & \ddots \\
& & \ddots & c_{N-1} \\
a_N & 2b_{N-1} & \cdots & c_{N-1} \\
& & a_N & 2b_N
\end{bmatrix}, \quad
r = \begin{bmatrix}
r_1, r_2, \cdots, r_N
\end{bmatrix}^T,
\]

\[
R = \begin{bmatrix}
RH_1 - a_1 r_0, \cdots, RH_N - c_N r_{N+1}
\end{bmatrix}^T
\]

We split the matrix \( A \) as follows

\[
A = G_1 + G_2
\]

(3.3)

Where
Assuming $G_1 + \omega I$ and $G_2 + \omega I$ are non singular matrices, then parallel method based on alternating group explicit algorithm is given by

$$
(G_1 + \omega I)r_{(k+1/2)} = R - (G_2 - \omega I)r_{(k)} \tag{3.4a}
$$

$$
(G_2 + \omega I)r_{(k+1)} = R - (G_1 - \omega I)r_{(k+1/2)} \tag{3.4b}
$$

where $\omega > 0$ is an acceleration parameters and $r_{(k+1/2)}$ is an intermediate vector(Evans [6]). Since $G_1 + \omega I$ and $G_2 + \omega I$ are non singular, the equations (3.4a) and (3.4b) in the explicit form

$$
r_{(k+1/2)} = (G_1 + \omega I)^{-1}
\left[
R - (G_2 - \omega I)r_{(k)}
\right]
\tag{3.5a}
$$

$$
r_{(k+1)} = (G_2 + \omega I)^{-1}
\left[
R - (G_1 - \omega I)r_{(k+1/2)}
\right]
\tag{3.5b}
$$

Simplifying equations (3.5a) and (3.5b), we obtain following algorithm

**First sweep:** For $j = 1(2)N - 1$

Let

$$
\Pi = (b_j + \omega)(b_{j+1} - \omega) - c_j a_{j+1} \neq 0
$$

$$
A = R_j - \beta a_j r_{j-1}(k) - (b_j - \omega)r_{j}(k)
$$

$$
B = R_{j+1} - (b_{j+1} - \omega)r_{j+1}(k) - \delta c_{j+1} r_{j+2}(k)
$$

where

$$
\beta = \begin{cases} 
0 & \text{if } j = 1 \\
1 & \text{otherwise}
\end{cases}
$$

$$
\delta = \begin{cases} 
0 & \text{if } j = N - 1 \\
1 & \text{otherwise}
\end{cases}
$$

Then

$$
r_{j}(k+1/2) = \frac{A(b_{j+1} + \omega) - B c_j}{\Pi}
$$

$$
r_{j+1}(k+1/2) = \frac{B(b_j + \omega) - A a_{j+1}}{\Pi}
$$

**Second sweep:** For $j = 1$

JIC email for contribution: editor@jic.org.uk
For $j = 2(2)N - 2$

Let

\[ \Pi = (b_j + \omega)(b_{j+1} - \omega) - c_j a_{j+1} \neq 0 \]

\[ A = R_j - a_j r_{j-1}^{(k+1)/2} - (b_j - \omega)r_j^{(k+1)/2} \]

\[ B = R_{j+1} - (b_{j+1} - \omega)r_{j+1}^{(k+1)/2} - c_{j+1} r_{j+2}^{(k+1)/2} \]

Then

\[ r_j^{(k+1)} = (A(b_j + \omega) - B c_j) / \Pi \]

\[ r_{j+1}^{(k+1)} = (B(b_j + \omega) - A a_{j+1}) / \Pi \]

For $j = N$

\[ r_N^{(k+1)} = \frac{R_N - (b_N - \omega)r_N^{(k+1)/2} - a_N r_{N-1}^{(k+1)/2}}{(b_N + \omega)} \]

### 4. Computational illustrations

To prove the efficiency of the implementation of the parallel alternating group explicit algorithm, all parameters with different valid values are taken. All results of numerical experiments, which were gained from implementation of the algorithms has been recorder in the tables. In all cases, initial guess is taken as zero vector and the iterations were stopped when difference of two consecutive iterative values is less than $10^{-10}$ was achieved. The acceleration parameter $\omega = 0.9$ is chosen for parallel AGE algorithm. The computational results shows superiority over corresponding sequential algorithm (Kai-Yuan Cai, et. al. [5]).

The exact solution $r(t) = \exp(t^4)$ is chosen. The right hand side function and boundary conditions may be obtained using the exact solution as a test procedure. The root mean square errors (RMSE), minimum number of time required to compute in terms of iterations $I_p$ for parallel case and $I_S$ for sequential case are presented in Table 1, 2 and 3.

### 5. Conclusion

From Table 1, it is evident that as software quality increases the average number of iterations in parallel case $I_p$ decreases more as compared to corresponding sequential iterations $I_S$. Table 2 shows that as work force increases the number of iterations in parallel and sequential case decreases and consequently execution time diminish with increase in work force. Table 3 confirms the superiority of parallel iterations on sequential one. Consequently high software complexity leads to very low execution time in parallel case as compared to its counterpart.
Table 1

<table>
<thead>
<tr>
<th>N</th>
<th>I_S</th>
<th>I_P</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>(\gamma = 0.5)</td>
</tr>
<tr>
<td>20</td>
<td>49</td>
<td>210</td>
<td>3.772e-05</td>
</tr>
<tr>
<td>30</td>
<td>100</td>
<td>214</td>
<td>7.796e-06</td>
</tr>
<tr>
<td>40</td>
<td>171</td>
<td>221</td>
<td>2.586e-06</td>
</tr>
<tr>
<td>50</td>
<td>261</td>
<td>232</td>
<td>1.103e-06</td>
</tr>
<tr>
<td>60</td>
<td>369</td>
<td>248</td>
<td>5.508e-07</td>
</tr>
</tbody>
</table>

|     |     |     | \(\gamma = 0.6\) |
| 20  | 60  | 158 | 2.772e-05 |
| 30  | 125 | 163 | 5.891e-06 |
| 40  | 214 | 173 | 1.976e-06 |
| 50  | 326 | 192 | 8.502e-07 |
| 60  | 461 | 236 | 4.277e-07 |

\(s_c = 2, w_f = 4, \xi = 20, \zeta = 4\)

Table 2

<table>
<thead>
<tr>
<th>N</th>
<th>I_S</th>
<th>I_P</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>(w_f = 2)</td>
</tr>
<tr>
<td>20</td>
<td>62</td>
<td>158</td>
<td>2.475e-05</td>
</tr>
<tr>
<td>30</td>
<td>129</td>
<td>164</td>
<td>5.517e-06</td>
</tr>
<tr>
<td>40</td>
<td>220</td>
<td>174</td>
<td>1.915e-06</td>
</tr>
<tr>
<td>50</td>
<td>335</td>
<td>195</td>
<td>8.444e-07</td>
</tr>
<tr>
<td>60</td>
<td>474</td>
<td>242</td>
<td>4.325e-07</td>
</tr>
</tbody>
</table>

\(s_c = 2, \gamma = 0.6, \xi = 20, \zeta = 4\)

Table 3

<table>
<thead>
<tr>
<th>N</th>
<th>I_S</th>
<th>I_P</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>(s_c = 2)</td>
</tr>
<tr>
<td>20</td>
<td>60</td>
<td>158</td>
<td>2.772e-05</td>
</tr>
<tr>
<td>30</td>
<td>125</td>
<td>163</td>
<td>5.891e-06</td>
</tr>
<tr>
<td>40</td>
<td>214</td>
<td>173</td>
<td>1.976e-06</td>
</tr>
<tr>
<td>50</td>
<td>326</td>
<td>192</td>
<td>8.502e-07</td>
</tr>
<tr>
<td>60</td>
<td>461</td>
<td>236</td>
<td>4.277e-07</td>
</tr>
</tbody>
</table>

\(s_c = 4\)

Fig. 1
1. References:


