Engineering Journal of Qatar University, Vol. 5, 1992, p. 135 - 143

# ACCELERATING THE CONVERGENCE OF CUBIC INTERPOLATION METHOD USING CONCURRENT ALGORITHM BASED ON THE TRANSPUTER TECHNOLOGY

Ahmed Hamad L. Al Mohanadi*, Mostafa Makkey**, Ahmed Saleh***
*Asst. Professor, Electrical Engineering Department, Qatar University, Doha, Qatar
**Asst. Professor, Assiut University, Assiut, Egypt.
***Computer Engineer, Electrical Engineering Department, Qatar University, Doha, Qatar.
*(First Received April 1992; accepted in revised form November 1992)*

## ABSTRACT

This paper describes a concurrent method to speed up the convergence of the QIOM (Cubic Interpolation Optimization Method) to obtain the global minima for any nonlinear continuous function. The QIOM is a naturally slow, sequential procedure, but usually guarantees a global solution. The acceleration of convergence is investigated from two points of view. One concurrency can be seen in the region bounded by the lower and upper limits on the function variables. The other concurrency is seen in investigating the QIOM algorithm itself. Due to the fact that many accelerators involve parallel processors, most of the algorithms today used for optimization are written specifically for sequential processors. This algorithm is implemented using Occam and Transputer Technology. The algorithm is tested on one well known nonlinear function.

## INTRODUCTION

Optimization is an important procedure used in many applications such as electronics, manufacturing, performance, reliability, etc. Many optimization algorithms are developed to run on sequential processing machines which use sequential programming languages like Fortran, Cobol, C, etc. Lately many machines use multiprocessing procedures, like the Cray, MMP, CLIP, and the Espirit 1085 project (4). The paper investigates the use of Transputer Technology and the Occam language in solving an optimization problem. The Transputer is designed to implement the process model of concurrency, expressed through the Occam Programming language which was developed in parallel with the hardware (1).

## THE TRANSPUTER AND OCCAM

The Transputer is a small, but complete Von Neumann computer. Concurrency is associated with the Transputer such that it can readily be built into networks and arrays, each working on its own job using its own local memory. It can execute parallel processes only as it shares processing time between currently active processes, changing from one process to another when the current process has to wait for communication.

A set of interconnected Transputer models is a set of parallel processors. The Transputer itself can model a set of concurrent processes (1). As a result, it is possible to map an Occam program, which is only a collection of processes, into a Transputer system in a variety of ways. If the Transputer has sufficient capacity, it can run the complete Occam program. Occam is the lowest level at which Transputers are programmed based on the concepts of concurrency and communication. The simplicity begins in Occam with the fundamental definitions that underlie its structure. Occam programs are constructed from three primitive processes assignment, input, and output.

The assignment K: = b sets the variable K to the value of the expression b. The output C ! b outputs the value of b to the channel C, and the input C ? k sets the variable k to the value input from channel C.

In order to control the order of execution of such processes, Occam identifies three fundamental constructs:

* The sequential constructor, SEQ; causes its components to be executed one after another, terminating when the last component terminates.

* The parallel constructor, PAR; causes its components to be executed simultaneously, or nearly so, terminating only after all of the components have terminated.

* Alternative constructor, ALT; chooses one component process for execution terminating when the chosen component terminates. In addition, Occam provides the conventional WHILE and IF constructors.

## THE QIOM ALGORITHM

The QIOM optimization algorithm for finding the minima of any function of n variables, f ($x_i$), depends on representing the function between bounded domain by a cubic equation (2).

$$h(\lambda) = a + b\lambda + c\lambda^2 + d\lambda^3 \qquad (1)$$

$$where\ \frac{\partial f}{\partial \lambda}|_A * \frac{\partial f}{\partial \lambda}|_B \leq 0$$

$$for\,all\ A \leq \lambda \leq B$$

The procedure used to represent a function of n variables f ($x_i$) by cubic equation h ($\lambda$) is:

1. Find

$$S_i = -\frac{\frac{\partial f}{\partial X_i}}{\sqrt{\sum \left(\frac{\partial f}{\partial X_i}\right)^2}} \qquad \textit{for all} \quad i=1,2,3,\ldots,n$$

2. Starting from initial point $X_i$, initial value for A, B, and incrementing $X_i$ by h * $S_i$ each time, where h is an incremental small value, find two numbers A and B corresponding to two points such that B = 2 * A, and

$$\sum \frac{\partial f}{\partial X_i} * S_i \; < \; 0 \qquad \textit{at first point}$$

and

$$\sum \frac{\partial f}{\partial X_i} * S_i \; > \; 0 \qquad \textit{at second point}$$

3. The function f ($x_i$) can be approximated by the one dimensional cubic equation h ($\lambda$) as in (1) between the two points found in step 2 with at least a minimum value between A, B.

4. Constants a, b, c and d in equation 1 can be evaluated as follows:

$$f_A = h(\lambda = A) = a + bA + cA^2 + dA^3 \qquad (2)$$

$$f_B = h(\lambda = B) = a + bB + cB^2 + dB^3 \qquad (3)$$

$$\bar{f}_A = \frac{dh}{d\lambda}(\lambda = A) = b + 2cA + 3dA^2 \qquad (4)$$

$$\bar{f}_B = \frac{dh}{d\lambda}(\lambda = B) = b + 2cB + 3dB^2 \qquad (5)$$

equations 2, 3, 4 & 5 can be solved to find the constants as:

$$Z=\frac{3\,(f_A-f_B)}{B-A}+\bar{f}_A+\bar{f}B \qquad (6)$$

$$b=\frac{B^2\bar{f}_A+A^2\bar{f}_B+2ABZ}{(A-B)^2} \qquad (7)$$

$$c=\frac{(A+B)\,Z+B\bar{f}_A+A\bar{f}_B}{(A-B)^2} \qquad (8)$$

$$d=\frac{2Z+\bar{f}_A+\bar{f}_B}{3\,(A-B)^2} \qquad (9)$$

$$a=\bar{f}_A-bA-cA^2-dA^3 \qquad (10)$$

5. Find λ that makes h (λ) minimum by

$$\frac{dh}{d\lambda}=b+2c\lambda+3d\lambda^2=0 \qquad (11)$$

Substituting equations 7, 8 & 9 into equation 11, we obtain

$$\lambda=A+\frac{(\bar{f}_A+Z\pm Q)\,(B-A)}{\bar{f}_A+\bar{f}_B+2Z} \qquad (12)$$

6. Find both h (λ) at λ computed in step 5 and f ($x_i$) at initial $x_i$ + λ * $S_i$.
7. Repeat the previous procedure until the following criterion is satisfied

$$\left|\frac{h(\lambda)-f(x_i)}{f(x_i)}\right| \le \epsilon \qquad (13)$$

Where ε is a small number whose value depends on the accuracy desired.

8. If equation 13 not satisfied check the value of f ($x_i$). If it is less than zero put A = λ or B = λ. If the value of f ($x_i$) is greater than or equal zero, go back to step 3 until equation 13 is satisfied.

Since the search for the minima of f $(x_i)$ takes place in a certain range of $x_i$, then this range can be partitioned to many ranges. The following formula can be used to provide the initial value for all variables in each range:

$$X_j = C + \frac{j-1}{n} * H$$

where
H is an incremental value
j is the range number
C starting point
n number of variables
such that

$$X_j \geq C \wedge X_j \leq D$$

where C and D define the range of f $(x_i)$ where the optimization will take place.

## CONCURRENCY IN QIOM

### 1. Concurrency in the bounded region

The region is divided into subranges where the QIOM is used in each region concurrently as shown in Fig. 1.

The QIOM algorithm can be applied to find the local minima in each subrange defined between two adjacent cells in $X_j$.

This procedure can be implemented using Occam language as follows.

```
PAR I = 0 FOR J
QIOM (I)
```

Where QIOM is a process that find the local minima between $X_i$ and $X_i + {}_1$. The global minimum is the lowest of all of the local minima achieved.

### 2. Concurrency is QIOM algorithm

The QIOM algorithm has several independent equations which can be evaluated concurrently.

**Table 2**

| | | |
|---|---|---|
| $X_1$ | −1.053 | 1.941 |
| $X_2$ | 1.028 | 3.854 |
| F (X) | 0.4866 | 1.9855 |

**Table 3**

| $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | $X_6$ | F (X) |
|---|---|---|---|---|---|---|
| 5.329 | 4.664 | 0.432 | 12.078 | 0.7524 | 0.8769 | 39.35131 |

## CONCLUSIONS

An optimization acceleration technique is implemented using Occam language under one Transputer T400. It showed a significant improvement in speed at real time compared to sequential approaches as shown below:

— FORTRAN 77 implemented on 286 processor with math. coprocessor took about 3 minutes to obtain all possible minima.

— Occam implemented on T400 Transputer took about 0.005 seconds as a maximum time for any major region.

This indicates that if an array of Transputers were used, then the global minima can be obtained in less than 0.005 seconds.

A concurrent processing structure appropriate to QIOM is shown in Fig. 1. The hardware architecture proposed is based on the transputer technology. Each rectangular block represents a discrete processor and they intercommunicate via high speed serial channel.

## REFERENCES

1. **Allan Burns, 1988,** Programming in Occam 2 . Addison - Weseley Publishing Co.

2. **Rao, S.S., 1984,** Optimization: Theory and Application: New Delhi, Wiley Eastern.

3. **Ahmed Hamad L. Al Mohanadi** and **Makkey, M.Y., 1991,** Global Minimization Using Cubic Method . 13th IMACS World Congress on Computation and applied mathematics, Dublin, Ireland.

4. **Bowler, K.C., Kenway, R.D.** and **Pawley, G.S., 1989,** An introduction to OCCAM2 Programming 2nd Edition, Chartwell-Bratt.

5. **Price, W.L., 1978,** Global Optimization Algorithm for a CAD workstation, Journal of Optimization Theory and Applications, vol. 55, no. 1, October.

6. **Ballard, D.H., Jellnek, C.A.** and **Schlnger, R., 1974,** An Algorithm for solution of constrained generalized Polynomial Programming Problems, Computer Journal, vol. 17, p. 261-266.