General Mathematics Vol. 16, No. 4 (2008), 35–40

Remarks on some parallel computation for spline recurrence formulas

Ioana Chiorean

Abstract

It is known that many problems may be solved more accurate by using spline functions instead of finite differences methods. In this way, the computational effort is, also, reduced, even if serial algorithms are used, due to the tridiagonal matrices involved. But this effort can be even more improved by parallel calculus. The purpose of this paper is to give some parallel computation approaches for the recurrence formulas which appear in generating the cubic spline functions.

2000 Mathematical Subject Classification: 65D05, 65D07, 37B20, 65Y05.

1 Statement of the problem

Let us have given the monotone spline knotset

$$x = \{x_i \in [a,b] \mid i = \overline{0, N+1}\}$$

for $[a, b] \subset \mathbb{R}$, where

$$a = x_0 < x_1 < x_2 < \cdots < x_N < x_{N+1} = b,$$

the stepsize $h_i = x_i - x_{i-1} > 0$, $i = \overline{1, N+1}$ and the data

$$F = \{f_i \mid i = \overline{0, N+1}\}.$$

In this paper, we consider $f_i = f(x_i)$, the value of a function $f : [a, b] \to \mathbb{R}$ in the knotpoint x_i, for every $i = \overline{0, N+1}$.

In general, it is known (see [2]) that the spline function of order l (with the defect one) is a piecewise polynomial function $s(x) = s_{l1}(x) \in C^{l-1}[a, b]$, which is a polynomial of the l-th degree on each interval $[x_{i-1}, x]$ and

$$s(x_i) = f(x_i) = f_i. \tag{1}$$

In what follows, we consider $l = 3$.

Under this circumstances, we want to determine the cubic polinom s on every interval $[x_{i-1}, x_i]$.

2 Some computations

According with [3], if s is a cubic polinom on $[x_{i-1}, x_i]$, one can write

$$s''(x) = M_{i-1}\frac{x_i - x}{h_i} + M_i\frac{x - x_{i-1}}{h_i}, \quad x \in [x_{i-1}, x_i] \tag{2}$$

where

$$M_i := s''(x_i).$$

Integrating twice relation (2), and using the interpolation relations (1), the following expression on $[x_{i-1}, x_i]$ is obtained:

$$(3) \quad s(x) = M_{i-1}\frac{(x_i - x)^3}{6h_i} + M_i\frac{(x - x_{i-1})^3}{6h_i} + \left(f_{i-1} - \frac{M_{i-1}h_i^2}{6}\right)\frac{x_i - x}{h_i} + \left(f_i - \frac{M_i h_i^2}{6}\right)\frac{x - x_{i-1}}{h_i}.$$

On the next interval, $[x_i, x_{i+1}]$, function s can be obtained by replacing i with $i+1$ in (3). Taking into account the continuity conditions of the first derivatives, s' in x_i, we get the following equality for computing the unknown values M_i, $i = \overline{1, N}$:

$$(4) \quad \frac{h_i}{6}M_{i-1} + \frac{h_i + h_{i+1}}{3}M_i + \frac{h_{i+1}}{6}M_{i+1} = \frac{f_{i+1} - f_i}{h_{i+1}} - \frac{f_i - f_{i-1}}{h_i}$$

which, after some computations in the r.h.s. term, generates:

$$(5) \quad \frac{h_i}{6}M_i + \frac{h_i + h_{i+1}}{3}M_i + \frac{h_{i+1}}{6}M_{i+1} = \frac{1}{h_i}f_{i-1} - \frac{h_i + h_{i+1}}{h_i h_{i+1}}f_i + \frac{1}{h_{i+1}}f_{i+1}$$

for $i = \overline{1, N}$.

(5) is a system of N equations with $N+2$ unknowns, with a tridiagonal matrix. In order to solve it, we may use the boundary conditions, if we consider that $M_0 = f_0$ and $M_{N+1} = f_{N+1}$ are given. Then, in matricial form, (5) can be written:

$$(6) \quad \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ \frac{h_1}{6} & \frac{h_1 + h_2}{3} & \frac{h_2}{6} & 0\dots & 0 & \\ 0 & \frac{h_2}{6} & \frac{h_2 + h_3}{3} & \frac{h_3}{6} & \dots & 0 \\ \vdots & & & & & \\ 0 & 0 & \dots & \frac{h_N}{6} & \frac{h_N + h_{N+1}}{3} & \frac{h_{N+1}}{6} \\ 0 & 0 & \dots & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ \vdots \\ M_N \\ M_{N+1} \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ \frac{1}{h_1} & -\frac{h_1+h_2}{h_1h_2} & \frac{1}{h_2} & 0\dots & 0 & \\ 0 & \frac{1}{h_2} & -\frac{h_2+h_3}{h_2h_3} & \frac{1}{h_3} & \dots & 0 \\ \vdots & & & & & \\ 0 & 0 & \dots & \frac{1}{h_N} & -\frac{h_N+h_{N+1}}{h_Nh_{N+1}} & \frac{1}{h_{N+1}} \\ 0 & 0 & \dots & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_N \\ f_{N+1} \end{bmatrix}$$

or

$$A \cdot M = B \cdot f. \tag{7}$$

The computation effort, in a serial algorithm, is of order $O(N^3)$, because the final matrices are not quite tridiagonal, due to the values at the boundaries.

3 Parallel approaches

(7) can be written

$$M = A^{-1} \cdot B \cdot f \tag{8}$$

so, in order to compute $M = (M_0, M_1, \dots, M_{N+1})$ we have to obtain the matriceal product $A^{-1} \cdot B$, with A and B matrices of order $N+2$. Due to the fact that the $(N+2)^3$ multiplications involved in the result matrix are very time-expansive, we try to compute all of them, simultaneously, considering (acc. with [1]) that we have enough processors.

Denoting by $C = A^{-1} \cdot B$, the information will be organized in the following manner: we generate two arrays, first with $N+2$ copies of matrix

A^{-1} and the other one with $N+2$ copies of row $B^T_{1\cdot}, B^T_{2\cdot}, \dots, B^T_{N+2,\cdot}$, and we make a single parallel multiplication operation between these two arrays:

$$\begin{array}{c} \uparrow \\ N+2 \\ \downarrow \end{array} \underbrace{\begin{bmatrix} & & & \\ A^{-1} & A^{-1} & \dots & A^{-1} \\ & & & \end{bmatrix}}_{\leftarrow (N+2)\cdot(N+2) \rightarrow} * \underbrace{\begin{bmatrix} B^T_{1\cdot} & B^T_{2\cdot} & \dots & B^T_{N+2,\cdot} \\ & \dots & & \\ B^T_{1\cdot} & B^T_{2\cdot} & \dots & B^T_{N+2,\cdot} \end{bmatrix}}_{\leftarrow (N+2)\cdot(N+2) \rightarrow} \begin{array}{c} \uparrow \\ N+2 \\ \downarrow \end{array}$$

Then, by means of double recursive technique, the additions in the final matrix are computed.

Remarks.

1. In general, by $X_{i\cdot}$ we understand the generation, from array X, of another array which contains the elements of the i-th row of X.

2. The operator "$*$" denotes, here, the parallel multiplication operation between the two arrays, which means that all the corresponding elements are multiplied in the same time.

4 Conclusions

By means of parallel calculus, the effort of computation is reduced by a factor of order $O(N^2)$, if our parallel system contains enough processors. If not, we may still improve this effort, by decomposing the matrices in smaller one, and applying parallel calculus to these smaller peaces.

References

[1] Chiorean, I., *Calcul paralel*, Ed. Microinformatica, 1995.

[2] Kobza, J., *Proc. of Algoritmy*, Conf. on Scientific Computing, 2000, pp. 58-67.

[3] Micula, Gh., *Funcţii spline şi aplicaţii.*

Ioana Chiorean

"Babes-Bolyai" University

Cluj-Napoca, Romania

e-mail: ioana@cs.ubbcluj.ro