ANALELE ŞTIINŢIFICE ALE UNIVERSITĂŢII „AL. I. CUZA"
Tomul XXXIV, s. I a. Matematică-Informatică, 1988, f. 4

NUMERICAL METHODS FOR SOLVING THE HEAT PROPAGATION PROBLEM ADAPTED TO THE PARALLEL COMPUTATION

BY

CONSTANTIN CHIRUŢĂ

Let us consider the simple problem of heat propagation (parabolical equation) :

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \text{ with } 0 \leqslant x \leqslant 1 \text{ and } t \geqslant 0\,; \tag{1}$$

with the initial conditions $u(x, 0)=f(x)$, for $0\leqslant x\leqslant 1$ and the boundary conditions $\begin{cases} u(0, t)=g_0(t) \\ u(1, t)=g_1(t) \text{ for any } t>0. \end{cases}$
In the domain under discussion, we shall consider the rectangular net made up of the knots $x_i=ih$, and $t_j=jk$, writing $u(x_i, t_j)=u_{i,j}$. In each of the knots we shall approximate the derivative $\frac{\partial u}{\partial t}$ with one of the formulae :

$$\frac{\partial u}{\partial t}(x_i, t_j) \cong \frac{u_{i,j+1}-u_{i,j}}{k}, \tag{a}$$

$$\frac{\partial u}{\partial t}(x_i, t_j) \cong \frac{u_{i,j}-u_{i,j-1}}{k}, \tag{b}$$

$$\frac{\partial u}{\partial t}(x_i, t_j) \cong \frac{u_{i,j+1}-u_{i,j-1}}{2k}, \tag{c}$$

and the derivative $\frac{\partial^2 u}{\partial x^2}$ with the most frequently used formula :

$$\frac{\partial^2 u}{\partial x^2}(x_i, t_j) \cong \frac{u_{i-1,j}-2u_{i,j}+u_{i+1,j}}{h^2}, \tag{d}$$

although in this instance, as well we could consider several other approximations e. g. with the formulae :

$$\frac{\partial^2 u}{\partial x^2}(x_i, t_j) \cong \frac{u_{i-1,j}-u_{i,j-1}-u_{i,j+1}+u_{i+1,j}}{h^2} \tag{d'}$$

or

(d'') $$\frac{\partial^2 u}{\partial x^2}(x_i, t_j) \simeq \frac{1}{2}\left[\frac{u_{i-1,\,j+1} - 2u_{i,\,j+1} + u_{i+1,\,j+1}}{h^2} + \frac{u_{i-1,\,j} - 2u_{i,\,j} + u_{i+1,\,j}}{h^2}\right].$$

(From (a) and (d') there results an explicit method and from (a) and (d'') there results the Crank-Nicolson implicit method).

We thus get three ways of finding the solution to problem (1) in the knots of the net we shall be dealing with, namely:

We shall get the first one considering the approximations (a) and (b):

$$\frac{u_{i,\,j+1} - u_{i,\,j}}{k} = \frac{u_{i-1,\,j} - 2u_{i,\,j} + u_{i+1,\,j}}{h^2}$$

or

$$u_{i,\,j+1} = \frac{k}{h^2}u_{i-1,\,j} + \left(1 - 2\frac{k}{h^2}\right)u_{i,\,j} + \frac{k}{h^2}u_{i+1,\,j};$$

and writing $r = k/h^2$ we get:

(2) $$u_{i,\,j+1} = ru_{i-1,\,j} + (1-2r)\,u_{i,\,j} + ru_{i+1,\,j}$$

which is an explicit method, which is proved to be unstable for $r > 1/2$ (The stability of the method is satisfied if the difference between the analytical solution and the numerical solution is bounded with the increase of j, for a constant k and any i).

We shall see that it is ideal for the parallel computation since it can be applied simultaneously in each knot of a horizontal line in the domain. But in order that the stability condition be satisfied, we must have $k \leqslant h^2/2$, which means that the k step must be very small, thus implying very much computation.

The second method is obtained considering the approximations (b) and (d):

$$\frac{u_{i,\,j} - u_{i,\,j-1}}{k} = \frac{u_{i-1,\,j} - 2u_{i,\,j} + u_{i+1,\,j}}{h^2}$$

or, with the same notation:

(3) $$-r\cdot u_{i-1,\,j} + (1+2r)\,u_{i,\,j} - r\cdot u_{i+1,\,j} = u_{i,\,j-1}$$

that is an implicit method which turns out to be stable for all values r and the system we have obtained must be solved so as to find the values in the knots.

Along each horizontal line in the net we shall get a non-homogeneous linear system of 3-wide band matrix, for whose solution we could use a parallel computer with l processors, especially with iterative methods of solving the systems, in the following manner:

If we must solve a system $Ax = b$, with $A = (a_{ij})_{n\times n}$, det $(A) \neq 0$, so that the diagonal is dominant, Jacobi's method:

$$x_i^{(m+1)} = \frac{b_i}{a_{ii}} - \sum_{\substack{j=1\\ j\neq i}}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(m)},$$

with $i = \overline{1, n}$, is easily paralleled because $x_i^{(m+1)}$ can be computed simultaneously $(i = \overline{1, n})$.

The Gauss-Seidel method is more difficult to deal with, but in this instance, too, we could try a parallelizing procedure, namely: Given that $l < n$ and l divisor of n, therefore $p = n/l$ an integer (otherwise we shall consider $p = [n/l]$).

The method is the following:

$$x_i^{(m+1)} = \frac{b_i}{a_{ii}} - \sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{(m+1)} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(m)}, \text{ with } i = \overline{1, n},$$

and its adaptation to the parallel computation could take the following course:

The first processor computes $x_i^{(m+1)}$ depending on $x_i^{(m)}$, with $i = \overline{1, n}$

The second processor computes $x_{p+1}^{(m+1)}$ depending on $x_i^{(m)}$,

. . . .

The l-processor computes $x_{p(l-1)+1}^{(m+1)}$ depending on $x_i^{(m)}$.

The components already calculated in the $(m+1)$ iteration are used by all the processors for the continuation of the computation, namely:

The first processor computes $x_2^{(m+1)}$ using $x_1^{(m+1)}, x_{p+1}^{(m+1)}, x_{2p+1}^{(m+1)}, \ldots, x_{p(l-1)+1}^{(m+1)}$ already calculated, as well as the other $x_i^{(m)}$ components from the previous iteration.

The second processor computes $x_{p+2}^{(m+1)}$ using the above, etc., and the l-processor computes $x_{p(l-1)+2}^{(m+1)}$ in the same manner.

In short, the technique can be written as such:

$$x_i^{(m+1)} = \frac{b_i}{a_{ii}} - \sum_{s=0}^{l-1}\left(\sum_{j=ps+1}^{p(s-q+1)+i-1} \frac{a_{ij}}{a_{ii}} x_j^{(m+1)} + \sum_{\substack{j=p(s-q+1)+i\\ j\neq i}}^{p(s+1)} \frac{a_{ij}}{a_{ii}} x_j^{(m)}\right)$$

where $p = n/l$, and q $(q = \overline{1, l})$ is the number of the processor which works on the i line of the system of equations to solve, with $i = \overline{1, n}$. We could proceed similarly if we considered the (b) and (d'') approximations, getting an implicit method, as well.

The third method is obtained considering the (c) and (d) approximations (the Richardson algorithm):

$$\frac{u_{i,\,j+1} - u_{i,\,j-1}}{2k} = \frac{u_{i-1,\,j} - 2u_{i,\,j} + u_{i+1,\,j}}{h^2}$$

or

(4) $$u_{i,\,j+1} = 2ru_{i-1,\,j} - 4ru_{i,\,j} + 2ru_{i+1,\,j} + u_{i,\,j-1}, \quad \text{where } r = k/h^2,$$

which is an explicit and stable method, with the only particularity that the values along the first horizontal line of the net must be obtained through some other methods (for instance through the first one given above), therefore it can also be easily applied through a parallel computation.

(From the (c) and (d') approximations we shall get the explicit and stable method of Du Fort and Frankel.)

Herewith we shall show how we could use the implicit methods in the parallel computation without resorting to methods of solving the non-homogeneous system of algebraic equations with band matrix, which implies more computation. Making use of the j and $j+1$ lines in the net the (3) formula becomes:

$$-ru_{i-1,\,j+1}+(1+2r)\,u_{i,\,j+1}-ru_{i+1,\,j+1}=u_{i,\,j} \tag{5}$$

In order to facilitate the application of this method, the so-called asymmetrical techniques of Sauliev have been used, namely:

$$-ru_{i-1,\,j+1}+(1+r)\,u_{i,\,j+1}=(1-r)\,u_{i,\,j}+ru_{i+1,\,j} \tag{6}$$

which is explicit if applied from left to right and

$$-ru_{i+1,\,j+1}+(1+r)\,u_{i,\,j+1}=(1-r)\,u_{i,\,j}+ru_{i-1,\,j} \tag{7}$$

which is also explicit if applied from right to left. These two methods (6) and (7) are also called *semi-explicit* methods. We can now join the use of these two methods in two adjacent knots, using them simultaneously in the group of adjoining knots $\{ih,\ (j+1)\,k\}$ and $\{(i+1)\,h,\ (j+1)\,k\}$. Thus, in the point $\{ih,\ (j+1)\,k\}$ the solution is computed with the relation:

$$-ru_{i+1,\,j+1}+(1+r)\,u_{i,\,j+1}=ru_{i-1,\,j}+(1-r)\,u_{i,\,j} \tag{8}$$

while in the point $\{(i+1)\,h,\ (j+1)\,k\}$ the solution is computed with the relation:

$$-ru_{i,\,j+1}+(1+r)\,u_{i+1,\,j+1}=(1-r)\,u_{i+1,\,j}+ru_{i+2,\,j} \tag{9}$$

Rewriting relations (8) and (9) under the form of matrix we get:

$$\begin{bmatrix}1+r & -r\\ -r & 1+r\end{bmatrix}\begin{bmatrix}u_{i,\,j+1}\\ u_{i+1\,j+1}\end{bmatrix}=\begin{bmatrix}1-r & 0\\ 0 & 1-r\end{bmatrix}\begin{bmatrix}u_{i,\,j}\\ u_{i+1,\,j}\end{bmatrix}+\begin{bmatrix}ru_{i-1,\,j}\\ ru_{i+2,\,j}\end{bmatrix} \tag{10}$$

where the matrix $\begin{bmatrix}1+r & -r\\ -r & 1+r\end{bmatrix}$ is non-singular and can be easily inversed to give the inverse matrix $\dfrac{1}{1+2r}\begin{bmatrix}1+r & r\\ r & 1+r\end{bmatrix}$.

So, the matricial relation (10) can be written explicitely as follows:

$$\begin{bmatrix}u_{i,\,j+1}\\ u_{i+1,\,j+1}\end{bmatrix}=\frac{1}{1+2r}\begin{bmatrix}1+r & r\\ r & 1+r\end{bmatrix}\left\{\begin{bmatrix}1-r & 0\\ 0 & 1-r\end{bmatrix}\begin{bmatrix}u_{i,\,j}\\ u_{i+1,\,j}\end{bmatrix}+\begin{bmatrix}ru_{i-1,\,j}\\ ru_{i+2,\,j}\end{bmatrix}\right\} \tag{11}$$

and after calculation, we get:

$$\begin{bmatrix}u_{i,\,j+1}\\ u_{i+1,\,j+1}\end{bmatrix}=\frac{1}{1+2r}\begin{bmatrix}r(1+r)\,u_{i-1,\,j}+(1-r^2)\,u_{i,\,j}+r(1-r)\,u_{i+1,\,j}+r^2u_{i+2,\,j}\\ r^2u_{i-1,\,j}+r(1-r)\,u_{i,\,j}+(1-r^2)\,u_{i+1,\,j}+r(1+r)\,u_{i+2,\,j}\end{bmatrix} \tag{12}$$

that is:

$$u_{i,\,j+1}=\frac{1}{1+2r}\,[r(1+r)\,u_{i-1,\,j}+(1-r^2)\,u_{i,\,j}+r(1-r)\,u_{i+1,\,j}+r^2u_{i+2,\,j}] \tag{13}$$

and:

$$u_{i+1,\,j+1}=\frac{1}{1+2r}\,[r^2u_{i-1,\,j}+r(1-r)\,u_{i,\,j}+(1-r^2)\,u_{i+1,\,j}+r(1+r)\,u_{i+2,\,j}]. \tag{14}$$

(13) and (14), when used alternatively, provide an explicit and stable method, well-suited for the parallel computation. The problem arises to determine the u values in the points adjoining the 0 and 1 boundaries of the domain, but they can also be determined with (6)-type formulae for the boundary on the left, that is

$$u_{1,\,j+1}=\frac{1}{1+r}\,[ru_{0,\,j+1}+ru_{2,\,j}+(1-r)\,u_{1,\,j}] \tag{15}$$

and with (7)-type formulae for the boundary on the right, that is:

$$u_{m-1,\,j+1}=\frac{1}{1+r}\,[ru_{m,\,j+1}+ru_{m-2,\,j}+(1-r)\,u_{m-1,\,j}] \tag{16}$$

$u_{0,\,j+1}$ and $u_{m,\,j+1}$ being known from the boundary condition ($x_0=0$ $x_m=1$).

We have applied these methods to problem (1) considering the initial condition:

$$f(x)=\begin{cases}2\,.\,x & \text{for } x\in[0,\ {}^1/_2]\\ 2\,(1-x), & \text{for } x\in[{}^1/_2,\ 1],\end{cases}$$

and the boundary condition:

$$g_0(t)=g_1(t)=0,\qquad \text{for any } t>0.$$

For this problem (studied by Smith (1965) and thoroughly presented by Ixaru [3]), the analytical solution is known:

$$u(x,\,t)=\frac{8}{\pi^2}\sum_{n=1}^{\infty}\frac{1}{n^2}\sin\left(\frac{n\pi}{2}\right)\sin\,(n\pi x)\,e^{-n^2\pi^2 t} \tag{17}$$

which, obviously, tends to 0 for $t\to\infty$.

We have chosen this problem to more easily simulate the parallelism on a serial computer (FELIX C-256), because $f(x)$ is a symmetrical function depending on $x=1/2$, and the solution preserves this symmetry for any $t\geqslant 0$.

We have confined ourselves to the square $x\in[0,\ 1]$, $t\in[0,1]$ considering $h=0{,}1$; $k=0{,}001$; hence $r=0{,}1<0{,}5$ necessary for the stability of method (2). The times we get are given in the table:

Methods	Necessary Time in Seconds	
	Serial	Parallel
(2)	39	25
(6) and (7)	39	25
(13) and (14)	41	26

We have also compared the results we obtained with the analytical results coming out of (17), for a few points in the domain at issue :

t	Method	$x=0,3$ or $x=0,7$		$x=0,5$	
		Numerical Solution	Analytical Solution	Numerical Solution	Analytical Solution
0,1	(2)	0.24723	0.24441	0.30562	0.30211
	(6) and (7)	0.25027		0.30842	
	(13) and (14)	0.24909		0.30708	
0,3	(2)	0.033895	0.033953	0.041897	0.041968
	(6) and (7)	0.036638		0.045144	
	(13) and (14)	0.036197		0.044618	
0,5	(2)	0.0047393	0.0047165	0.0058581	0.00583
	(6) and (7)	0.0053632		0.0066085	
	(13) and (14)	0.0052597		0.0064834	
0,7	(2)	0.00066266	0.0006552	0.00081909	0.00080986
	(6) and (7)	0.0007851		0.00096738	
	(13) and (14)	0.00076428		0.000942	
0,9	(2)	0.000092654	0.000091016	0.00011453	0.0001125
	(6) and (7)	0.00011493		0.00014161	
	(13) and (14)	0.00011213		0.00013822	

We have chosen $x=0,5$ mostly because $f(x)$ is not derivable in this point, and $x=0,3$ or $x=0,7$ so as to draw a comparison between the approximation in these points and the points with $x=0,5$.

REFERENCES

1. C h i r u ț ă, C. — *Metode numerice de rezolvare a ecuațiilor diferențiale cu derivate parțiale adaptate calculului paralel.* Paper held at the 6[th] Edition of the Informatics Colloquim INFO—IAȘI '87, October, 9[th]—10[th], 1987, p. 146—152.
2. E v a n s D. J. — *Parallel Algorithms in Computational Linear Algebra.* Parallel Computer and Computation, C.W.I. Newsletter, nr. 9, 1985, p. 55—79.
3. I x a r u, Gr. I. — *Metode numerice pentru ecuații diferențiale cu aplicații*, Editura Academiei R.S.R., 1979, 208 p.
4. K o t o v, V. E. — *Elementî parallelinovo programmirovania.* Radio i sviazi, 1983, 240 p.

Received 9.IV.88

Faculty of Mathematics
University „Al. I. Cuza" Iași,
Iași R-6600, Romania

ANALELE ȘTIINȚIFICE ALE UNIVERSITĂȚII „AL. I. CUZA"
Tomul XXXIV, s. I a. Matematică-Informatică, 1988, f. 4

SYNTAX ORIENTED DATA PROCESSING

BY

LUCA DAN ȘERBANAȚI, VALENTIN CRISTEA, ADRIAN POREMBSCHI, MIHAELA SABIN, DANIEL SABIN, ADRIAN PRIVACHE

Introduction. Developing a program, the programmer aims for it to accomplish a certain computation. By writing code, he assembles various computer operations. In other words, while being preoccupied by syntax, the programmer must constantly think in terms of semantics. By being fully aware of the difference between a symbol and its meaning, between an operator and an operation, between a program and a processor which executes it, or in general terms, between syntax and semantics, the programmer has in mind, explicitly or not, the relations between the syntactical structures of a program and the semantical ones.

As far back as 1961, I r o n s suggested in [1] the existence of such a relation in the compilation process. Today, a widely accepted idea is that access to semantics is accomplished exclusively by way of syntax. This being the case we can use it in the design and implementation of many data processings.

This paper sets out to classify the most frequently encountered syntax oriented data processings, to introduce models for these processings, as well as solutions for their implementation.

1. Syntax and semantics. We shall consider, in the following, that the main formal operational model in the study of the programming languages syntax continues to be the context-free grammar (*CFG*). In what concerns semantics, no such formal model, similar to the one mentioned above, exists, at least not in terms of suggestiveness, conciseness or ease in handling, although several such proposals for models have been made ([2], [3], [4], etc.).

Irrespective of the formal model chosen for the semantics, one can easily ascertain that the syntax of a context-free language (*CFL*) is allways richer in information than any semantics of the language and any such associated semantics actually makes the language "poorer" in terms of diversity of expression. For example, modifying the effect of a program (in other words its meaning) can be obtained exclusively by modifying the program text, thus its syntactical structures. In contrast, different syntactical programs (i.e. with nonidentical texts) can have identical effects. In formal terms, if L is a language (i.e. a subset of the set of symbols over an alphabet) and D the domain of sentence evaluation, a language semantics is an application : $L \rightarrow D$.