Report for IBM
1985

TWO - POINT RAY - TRACING ON A MICROCOMPUTER

V. Pereyra

Escuela de Computacion
Facultad de Ciencias
Universidad Central
Caracas VENEZUELA

1. Introduction

In many applications in Seismology, underwater acoustics, and recently in computer graphics solid modeling, it is necessary to trace optical rays through some general medium. If this medium is inhomogeneous, the ray trajectories are solutions of a set of nonlinear ordinary differential equations of the second order, which in many cases are subject to boundary conditions in two or more points.

In the past few years we have produced a number of software packages for tracing seismic, optical or acoustical rays in two or three dimensional, fully inhomogeneous media, with discontinuous interfaces [1 - 5].

Those programs have been developed for, and are operational in main frame, and large to small minicomputers. They are written in very transportable FORTRAN, and are based on widely distributed packages for solving general two-point boundary-value first-order systems of non-linear ordinary differential equations [6 - 8].

The question has arisen, on how much of this effort can be transported down to a microcomputer based operation.

In this paper we report on the results obtained so far on work done in that direction. The microcomputer chosen for the development is an IBM PC/XT, with 320K bytes in RAM, equipped with a black and white 640 x 200 graphics monitor (in high resolution; or colour 320 x 200 in medium resolution), a dot matrix graphics printer, one floppy, and one hard disk unit (10 mbytes).

In § 2 we describe our first exploratory steps, and our current system design choices. In § 3 we develop the mathematical procedure and the corresponding numerical algorithm. In 4 we give some preliminary results, and finally in § 5 we comment on future developments and make some closing remarks.

2. First steps

Our initial inclination was to see if any of the original, already written packages, could be fit directly, or with minor modifications into the microcomputer. As a matter of fact, that had already been done, using a more powerful machine (NICAT).

The most recent issues of our packages for large computers have between 3000 and 4000 FORTRAN statements, and there are separate versions for two and three space dimensions, although they share a big piece of code consistent of the two-point ODE solver PASVA4 in a modified format.

With some difficulty, we managed to compile all the subroutines of the 2D package, although several of them had to be fragmented in order to fit the available space (we were using at the time a smaller memory machine,with only 128K). However, we could never link this huge program and eventually gave up. The lessons learned were: (a) Subroutines with more than about 200 FORTRAN statements cannot be compiled in a machine of that size. (b) Compiling and linking long FORTRAN programs on a machine equipped only with floppy disks takes forever, and it is not a media adequate for program development. (c) The FORTRAN compiler employed (Microsoft, January 1982) is not suited for software development, because of implementation limitations, and the total abscense of debugging aids.

Thus, we decided to do a complete retooling of our system, sacrificing some aspects and defering others to a second stage, in order to have , as soon as possible, some kind of working version that would give us an indication of the plausibility and possible limitations of the project.

Having taken that decision, things started moving more quickly. In the literature quoted above,the reader can find ample details on the algorithms, both mathematical and numerical, and of course the physical basis of the problem, and some of its applications, at least on large computers. Additionally, in 3 we describe in detail the new simplified approach. Here we would like to run rapidly through the main differences between this and earlier approaches.

The most important decision made, was to do away with the general purpose two-point solver PASVA4, by far the bulkiest part of the packages. Of course, with it we were loosing many years of software development, and a careful, well tested , very general code. However, one of the questions we wanted to answer was : is it possible to preserve as much generality as we can in the physical problem, simplify drastically the numerical solver, and still obtain acceptable results?

As we have already indicated, PASVA4 is a general purpose two-point boundary value problem solver for first order nonlinear systems of ordinary differential equations, which can

have discontinuous right hand sides, and be mixed with algebraic equations, and have correspondingly, additional unknown parameters. The numerical method used, is based on a simple finite difference approximation, which can be enhanced by deferred corrections. The code is able to select automatically quasi-optimal non-uniform meshes, to estimate the errors accurately, and thus to produce as precise a result as is requested by the user in a most efficient form.

The boundary value nature of the auxiliary conditions, and the chosen method of discretization, leads to the solution of large sparse, structured, systems of non-linear equations. The largest part of the code is actually dedicated to the solution of this problem by a modified Newton type method, and a specially designed Gaussian elimination that preserves sparseness.

These features are most desirable for a general purpose code, and they are good to have in a general ray-tracing routine, specially for difficult problems.

Continuing however in the spirit of finding a compromise between performance, reliability and simplicity, we have chosen to start with a simple discretization on an uniform mesh, with no enhancements or error estimations. Since we want to keep the basic philosophy of global discretizations ("bending of rays"), as opposite to "shooting techniques", we still have large, sparse, structured systems of algebraic equations to solve, for which we will use a straightforward Newton method, and a band solver for the resulting systems of linear equations.

Since we are not forced by a canned solver to reduce the problem to a first order system, we will keep it in its second order form, neglecting in this way the possible simplifications that a wise choice of coordinates allow us in the 2D case. In this manner, two or three dimensions only differ in the number of differential equations and dependent variables. Thus, switching between the two cases can be governed by just one parameter : IDIM, that needs to take upon the values 2 or 3, and therefore an unique code can be written .

3. Problem formulation and discretization

The usual equations for tracing rays on an inhomogeneous three dimensional medium are :

$$(1) \qquad \frac{d}{ds}\left(u(\underset{\sim}{\eta})\frac{d}{ds}\underset{\sim}{\eta}\right) = \nabla u^T(\underset{\sim}{\eta}) ,$$

where $\underset{\sim}{\eta}(s) = (\eta_1, \eta_2, \eta_3)$ represents the ray trajectory. s is arc length along the ray, and $u(\underset{\sim}{\eta}) = 1 / v(\underset{\sim}{\eta})$, with $v(\underset{\sim}{\eta})$ the velocity of propagation of the waves being traced (P, S, ...), and ∇u^T is the gradient vector transposed.

The inhomogeinity is reflected by the fact that v is non-constant. An important quantity to be computed is the travel

time τ, defined by

$$\tau(s) = \int_0^s u(t)\, dt \,. \tag{2}$$

Although in the past we have used the form (1) of the ray equations, we shall now introduce a new variant, obtained considering τ as the independent variable instead of s.

A short computation shows that the new form of the ray equations is:

$$\underset{\sim}{\eta}'' - 2u \langle \nabla v , \underset{\sim}{\eta}' \rangle \underset{\sim}{\eta}' + v \nabla v^T = 0 \,, \tag{3}$$

where $\underset{\sim}{\eta}' = d\underset{\sim}{\eta} / d\tau$, $\underset{\sim}{\eta}'' = d^2\underset{\sim}{\eta} / d\tau^2$.

We will be mainly interested in the two-point ray-tracing problem, in which the initial and final position of the ray are prescribed:

$$\underset{\sim}{\eta}(0) = \underset{\sim}{\eta}_{source} \,, \quad \underset{\sim}{\eta}(T) = \underset{\sim}{\eta}_{receiver} \,,$$

where T (unknown), is the total travel time.

In order to have the problem posed on a known interval, we make a further change of variable, to a normalized travel time:

$$[0,T] \xrightarrow{t} [0,1] \,,$$ and we get finally:

$$\ddot{\underset{\sim}{\eta}} - 2u \langle \nabla v, \dot{\underset{\sim}{\eta}} \rangle \dot{\underset{\sim}{\eta}} + v T^2 \nabla v^T = 0 \,, \tag{4}$$

$$\underset{\sim}{\eta}(0) = \underset{\sim}{\eta}_S \,, \quad \underset{\sim}{\eta}(1) = \underset{\sim}{\eta}_R \,, \tag{4'}$$

with $\dot{\underset{\sim}{\eta}} = d\underset{\sim}{\eta} / dt$, $\ddot{\underset{\sim}{\eta}} = d^2\eta / dt^2$.

The same equations apply in two dimensions, with the difference that $\underset{\sim}{\eta} = (\eta_1, \eta_2)$, and therefore we have only two equations instead of three. By choosing some different variables, it is possible to formulate the two-dimensional problem with only three first order equations (see [1]), but we will not do that here, since we want to preserve the second order formulation.

Equations (4) - (4') are discretized on an uniform mesh $t_i = (i-1)\,h$, $h = 1/(n-1)$, $i = 1, \ldots, n$:

$$r(\underset{\sim}{\eta}_i) \equiv \underset{\sim}{\eta}_{i+1} - 2\underset{\sim}{\eta}_i + \underset{\sim}{\eta}_{i-1} - \tfrac{1}{2} u_i \langle \nabla v_i \,,\, \underset{\sim}{\eta}_{i+1} - \underset{\sim}{\eta}_{i-1} \rangle (\underset{\sim}{\eta}_{i+1} - \underset{\sim}{\eta}_{i-1}) + (Th)^2 v_i \nabla v_i^T = 0 \,, i = 2, \ldots, n-1 \,, \tag{5}$$

$$\underset{\sim}{\eta}_0 = \underset{\sim}{\eta}_S \,, \quad \underset{\sim}{\eta}_n = \underset{\sim}{\eta}_R \,, \tag{5'}$$

where the subindex refers to evaluation at t_i , and $\underset{\sim}{\eta}_i$ will hopefully, approximate $\underset{\sim}{\eta}(t_i)$.

We need to introduce an additional condition to account

for the unknown parameter T. This is the so called arc-length condition, which in this case takes the form

$$(4'') \qquad r_T \equiv \left\| d\,\underset{\sim}{\eta}(0) / dt \right\|_2^2 - T^2 v^2(0) = 0 ,$$

and its discretization

$$(5'') \qquad \langle \underset{\sim}{\eta}_1 - \underset{\sim}{\eta}_0, \underset{\sim}{\eta}_1 - \underset{\sim}{\eta}_0 \rangle - [h\,T\,v(0)]^2 = 0 .$$

This is an $O(h^2)$ approximation, in the sense that if the continuous problem have an isolated solution $\underset{\sim}{\eta}(t)$, the discretized problem has also an isolated solution $\{\underset{\sim}{\eta}_i\}$, and the discretization is stable, then

$$\max_{i=1,\ldots,n} \left| \underset{\sim}{\eta}(t_i) - \underset{\sim}{\eta}_i \right| \approx O(h^2) .$$

System (5) consists of m x (n - 2) + 1 nonlinear equations, where m = 2 or 3. In order to solve this system we introduce a linearized version, that will be the main part of a Newton iteration :

$$\begin{aligned} L_i\,\delta\underset{\sim}{\eta} \equiv\ & \delta\underset{\sim}{\eta}_{i+1} - 2\,\delta\underset{\sim}{\eta}_i + \delta\underset{\sim}{\eta}_{i-1} - \tfrac{1}{2}\,[\langle \nabla v_i , \underset{\sim}{\eta}_{i+1} - \underset{\sim}{\eta}_{i-1} \rangle * \\ & \langle \nabla u_i , \delta\underset{\sim}{\eta}_i \rangle + u_i \langle (H v_i\,(\underset{\sim}{\eta}_{i+1} - \underset{\sim}{\eta}_{i-1})), \delta\underset{\sim}{\eta}_i \rangle + \\ & u_i \langle \nabla v_i , \delta\underset{\sim}{\eta}_{i+1} - \delta\underset{\sim}{\eta}_{i-1} \rangle]\,(\underset{\sim}{\eta}_{i+1} - \underset{\sim}{\eta}_{i-1}) - \\ & \tfrac{1}{2}\,u_i \langle \nabla v_i , \underset{\sim}{\eta}_{i+1} - \underset{\sim}{\eta}_{i-1} \rangle (\delta\underset{\sim}{\eta}_{i+1} - \delta\underset{\sim}{\eta}_{i-1}) + \\ & (T\,h)^2 [\, v_i H v_i + \nabla v_i^T \nabla v_i]\,\delta\underset{\sim}{\eta}_i + 2\,T\,h^2\,v_i\,\nabla v_i^T\,\delta T , \end{aligned} \qquad (6)$$

$$i = 2, \ldots, n-1$$

$$(6') \qquad \delta\underset{\sim}{\eta}_0 = 0 , \qquad \delta\underset{\sim}{\eta}_1 = 0 ,$$

$$(6'') \qquad L_T \equiv 2 \langle \underset{\sim}{\eta}_1 - \underset{\sim}{\eta}_0 , \delta\underset{\sim}{\eta}_1 \rangle - 2\,[\,h\,v(0)\,]^2\,T\,\delta T .$$

From these expressions we observe that the matrix of the linear system (for the unknown corrections $\{\delta\underset{\sim}{\eta}_i\}$), has almost block tri-diagonal form. In fact, if we order the unknowns and equations as $[T, \delta\underset{\sim}{\eta}_1, \ldots, \delta\underset{\sim}{\eta}_{n-1}]$, $[L_T, L_1, \ldots, L_{n-1}]$ then, schematically, we have :

$$(7) \qquad \begin{bmatrix} * & - & 0 & . & . & . & 0 \\ | & \square & \square & & & & \\ | & \square & \square & \square & & 0 & \\ | & & & \ddots & \ddots & & \\ | & & & & \ddots & \ddots & \\ 0 & & & & & \ddots & \\ | & & 0 & \square & \square & \square & \end{bmatrix} \delta\underset{\sim}{\eta} = - \begin{bmatrix} r_T \\ \\ r(\underset{\sim}{\eta}) \end{bmatrix}$$

where the m x m blocks, and the (1 x m) , (m x 1) lines indicate non -zero elements.

Thus, except for the first row and column, this is a block tri-diagonal matrix, with either 2 x 2 or 3 x 3 blocks, depending upon the dimension of the problem. If we write, in an obvious notation:

$$
\begin{array}{c} 1 \\ (8) \\ m \times (n-2) \end{array}
\left[\begin{array}{c|c} L_T^T & L_T^\eta \\ \hline L_\eta^T & L_\eta^\eta \end{array}\right]
\left[\begin{array}{c} \delta T \\ \hline \delta \underset{\sim}{\eta} \end{array}\right]
= -\left[\begin{array}{c} r_T \\ \hline r_{\underset{\sim}{\eta}} \end{array}\right]
$$

$1 \qquad m \times (n-2)$

then we can reduce the solution of this system to that of the solution of two systems with the block tri-diagonal matrix L_η^η, and a few additional operations. In fact, the first equation can be replaced by

$$(L_T^\eta (L_\eta^\eta)^{-1} L_\eta^T - L_T^T)\, \delta T = r_T - L_T^\eta (L_\eta^\eta)^{-1} r_{\underset{\sim}{\eta}} \equiv \tilde{r}_T ,$$

from where we readily obtain δT

$$\delta T = \tilde{r}_T / \rho ,$$

provided that $\rho = L_T^\eta (L_\eta^\eta)^{-1} L_\eta^T - L_T^T \neq 0$.

Finally, replacing this in the second block of equations we obtain $\delta \underset{\sim}{\eta}$

$$\delta \underset{\sim}{\eta} = - [L_\eta^\eta]^{-1} r_{\underset{\sim}{\eta}} - \delta T [L_\eta^\eta]^{-1} L_\eta^T .$$

Thus, the following algorithm can now be extracted from these formulae :

a) Solve $L_\eta^\eta \ \alpha = L_\eta^T$

b) Solve $L_\eta^\eta \ \beta = -r_{\underset{\sim}{\eta}}$

c) Compute $\rho = L_T^\eta \ \alpha + L_T^T$; $\tilde{r}_T = r_T + L_T^\eta \ \beta$

d) If $\rho \neq 0$ then $\delta T = \tilde{r}_T / \rho$ else : STOP 'problem is singular'

e) Compute $\delta \underset{\sim}{\eta} = \beta - \delta T \ \alpha$.

The Newton iteration requires an initial guess for T and $\underset{\sim}{\eta}$ say T^0, $\underset{\sim}{\eta}^0$. With these values, one computes the matrix and right hand side in (8); by means of the algorithm above he obtains the corrections δT , $\delta \underset{\sim}{\eta}$, and then upoates T^0 . $\underset{\sim}{\eta}^0$

$$T^1 = T^0 + \delta T \quad , \quad \underset{\sim}{\eta}^1 = \underset{\sim}{\eta}^0 + \delta \underset{\sim}{\eta}$$

and iterates.

If the initial guesses are in the region of convergence of the Newton iteration and the Jacobian matrix is well conditioned, then this iteration will generally converge quadratically to a solution of the problem.

For solving the block tri-diagonal systems we can use either an elaborated solver as in the PASVA series, or as we choose to do now, a band solver with partial pivoting. This requires some additional storage, but it has the advantage of a fairly compact implementation.

Observe that since we have to solve two systems with the same matrix of coefficients, it is appropriate to use a solver that performs separately the LU decomposition of the matrix, and the solution of the resulting triangular systems, since in this way we need to do the first phase (the most expensive one) only once. Also one can obtain further savings by not updating the Jacobian matrices at every Newton step. This is even more important, and feasible, when calculating families of close-by rays.

4. Some preliminary results

We have made an implementation of the procedure described in 3, in Microsoft Basic.

This implementation, MICRORA2, has simple interactive input of parameters and graphic output. The user has to provide a velocity subroutine (VELMOD, lines 590-630), and one for its first and second derivatives (DVEL, lines 640-720).

The ray coordinates $\eta_1, \eta_2 (\eta_3)$ are stored in the array ETA(I,J), J = 1,2 (3) , I = 1,...,N = # mesh points. The velocity $v(\underline{\eta})$ is stored in an array V(I) , I = 1,...,N, and its first and second partial derivatives in DV(I,J), and DDV(I,J,K) respectively.

Upon starting MICRORA2, the user will be prompted to enter:

Dimension (either 2 or 3)

Mesh points : N

Print option : The largest, the more detail in printing intermediary results is produced.

Plot option : 0 No plot , 1 Plot.

Screen mode : 1 Color , 2 High resolution.

Precision : The program will attempt to make the maximum absolute residual less than this quantity.

Xsource, Ysource (Zsource) Source coordinates.

Xstat, Ystat (Zstat) Coordinates of first receiver.

Delxstat, Delystat (Delzstat) Constant increments for further receivers.

Of Rays.

If IDIM = 2 , then upon convergence of the first ray, the prompt :

Enter Xmin, Xmax, Ymin, Ymax

will appear, allowing the user to enter a plotting window (in user coordinates). Then the rays will be plotted in this window .

If IDIM = 3 , then the rays are saved on a file named PLOTRAY.1, which is in a format adequate for further processing by a modified 3D graphics package (ISO3 , Copyright 1982 by Kern Publications).

In Fig. I, II we give a sample of the results for the following data:

I. $v(\underline{\eta}) = 5 + \eta_2 + 2 * \eta_1$ kms/sec

IDIM = 2 , N = 10 , Precision = 0.01 ,
Xsource, Ysource = (0, 0) ;
Xstat , Ystat = (.5, 1)
Delxstat , Delystat = (0.05, -0.1)
# of rays = 10
(Units in Kms.)

II. $v(\underline{\eta}) = 5 + \eta_2 + 2 * (\eta_1 + \eta_2)$ kms/sec

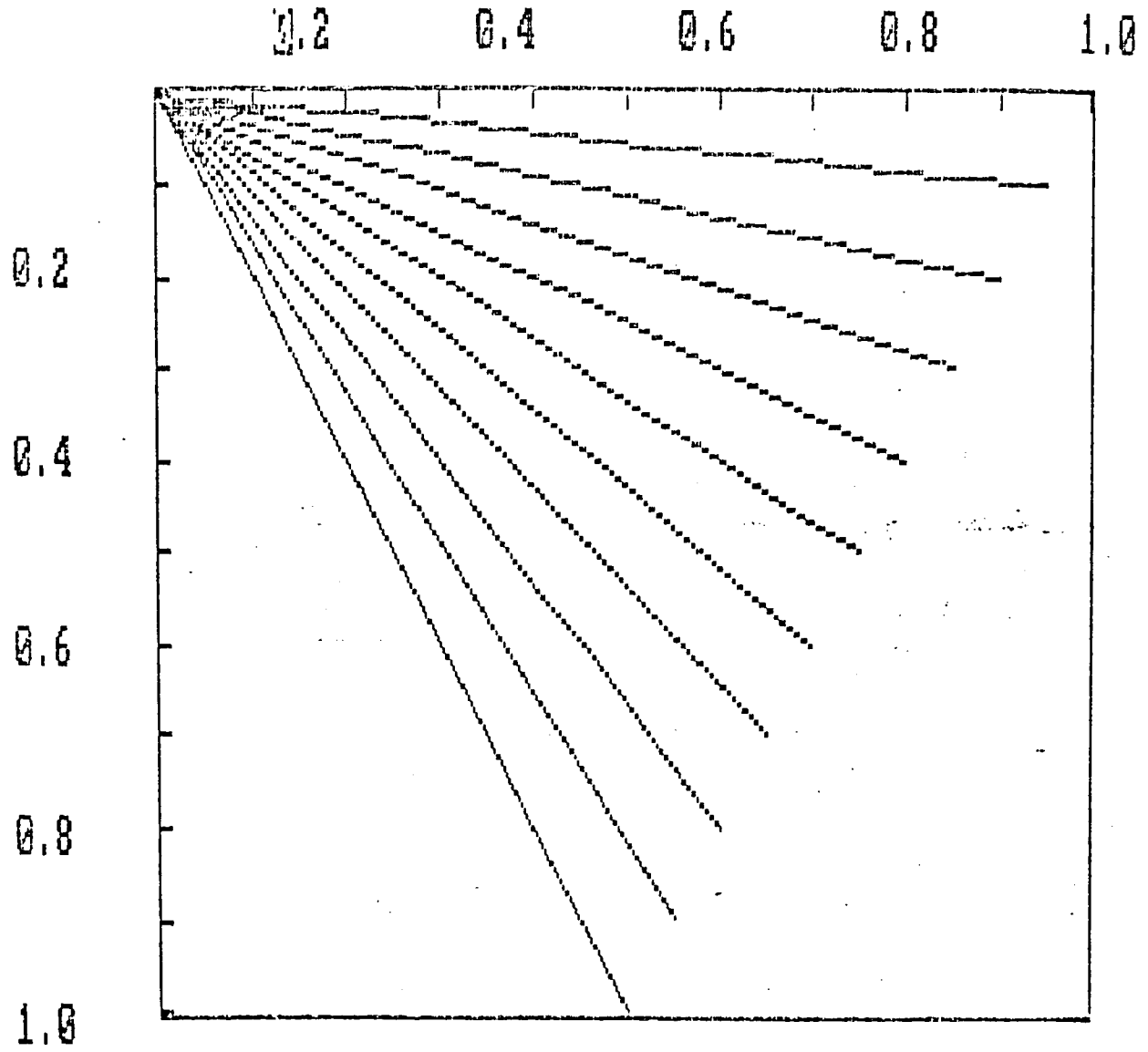

FIG. I

5. Future developments

The preliminary results we have reported show the soundness of the current approach. In order to continue these efforts in the direction of an useful package for general users, we must now proceed on two fronts.

First, a more friendly user interface should be developed. A menu-driven system, with good facilities for data input, graphic output, and interactive modeling, that takes advantage of both the hard and the soft-ware we are using would be indicated.

In this respect, we are planning to add to our station a digitizing tablet and a plotter. Also, it is planned to add an 8087 co-processor, which should considerably enhance the performance of the system.

An important extension to the current version is to piecewise smooth media, in order to include many geologically interesting features. This will require some substantial modifications to the system of equations to be solved, and also we would need to include the additional model information.

The main point here is that when a ray traverses from one medium to another, crossing an interface, it changes direction discontinuously, according to Snell's law [2,5]. It can also reflect, i.e. return to the same media, and in general a ray signature should be given, since in such a complicated medium, there will be multiple solutions to the ray tracing problem between two points.

One thing that we can save is the sparse structure of the system of equations, and with it, the current solver. In fact, it is enough to observe that the discretization of Snell's law couples also three consecutive points, just as the difference equations already used; therefore, if we insist that the intersections of the ray with the interfaces be mesh points, and put in the system as the equation for this points Snell's law, then the overall structure of the system will be just as in (7).

REFERENCES

1. Pereyra, V., W.H.K. Lee, and H.B. Keller "Solving two-point seismic ray-tracing problems in a heterogeneous medium". Bull. Seism. Soc. America 70 : 79 - 99 (1980).

2. Pereyra, V. and G. Wojcik "Ray tracing on inhomogeneous, piece wise continuous media" . In preparation.

3. Pereyra, V. "Two-point ray tracing in heterogeneous media and the inversion of travel time data". Comp. Methods App. Sc. and Eng. (Edit. R. Glowinski and J. L. Lions). North - Holland Pub. Co., Amsterdam, 553 - 570 (1980).

4. Pereyra, V., H.B. Keller, and W.H.K. Lee "Computational methods for inverse problems in geophysics". Physics of the Earth and Planetary Interiors 21 : 120 - 125 (1980).

5. Pereyra, V. and J.A. Rial "Trazado de rayos sismicos en cuenca sedimentarias". Acta Cient. Venezolana 32 : 500 - 502 (1981).

6. Pereyra, V. "PASVA3 : An adaptive finite difference FORTRAN program for first order, nonlinear, ordinary boundary problems". Lecture Notes Comp. Sc. 76 : 67 - 88. Springer - Verlag, Berlin (1979).

7. Lentini, M. and V. Pereyra "PASVA4 : An ordinary boundary solver for problems with discontinuous interfaces and algebraic parameters". Mat. Aplicada e Comp. 2 : 103 - 118 (1983).

8. Pereyra, V. "Deferred corrections software and its application to seismic ray-tracing". To appear in Error Asymptotics and Defect Corrections (Edit. K. Bohmer and H. Stetter), Birkhauser-Verlag, Vienna.

9. Pereyra, V. "Desarrollo de software numerico con aplicaciones en la Industria del Petroleo". To appear in Petroleo y Tecnologia, Maracaibo, Venezuela.