Computations in the model composed during the 1998 consortium meeting

Petr Bulant & Luděk Klimeš
Department of Geophysics, Charles University, Ke Karlovu 3, 121 16 Praha 2, Czech Republic, Fax: +420-2-2191 1214,
E-mails: bulant@seis.karlov.mff.cuni.cz, klimes@seis.karlov.mff.cuni.cz

Summary

The aim of the paper is to demonstrate the input data prepared for the travel–time computation and seismogram generation in the model created during the technical session of the 1998 consortium meeting and to comment and explain the most important input parameters.

1 Introduction

The sample synthetic model was created during the technical session of the SW3D consortium meeting in June 1998 and was named "98". The aim of the work done was not to create a realistic model, but to demonstrate capabilities of program packages `MODEL` and `CRT`. The participants of the meeting were asked how should the model look like and which computations should be performed. The data and MS-DOS batch files for the computations were prepared just during the session. It was thus a good opportunity to learn a lot about both the program packages.

"History files" have recently been introduced to control the computations. Their use is explained, e.g., by Klimeš (1998). Three history files control three computations in model 98: (a) visualization of P and S wave velocity cross–sections through the model, (b) computation of travel times of the refracted wave and (c) computation of synthetic seismograms of four reflected elementary waves.

Only the most important parameters in the input data files and in the history files will be described in this paper. Detailed description of input files describing seismic models may be found in papers by Klimeš (1995) and Bulant & Klimeš (1996), the application of history files to the calculation of synthetic seismograms is explained by Klimeš (1998). For detailed description of other parameters refer to hypertext documentation of package CRT version 5.20, which may be found on the SW3D-CD-2 disk.

In: Seismic Waves in Complex 3-D Structures, Report 7 (Department of Geophysics, Charles University, Prague, 1998), pp. 33–56

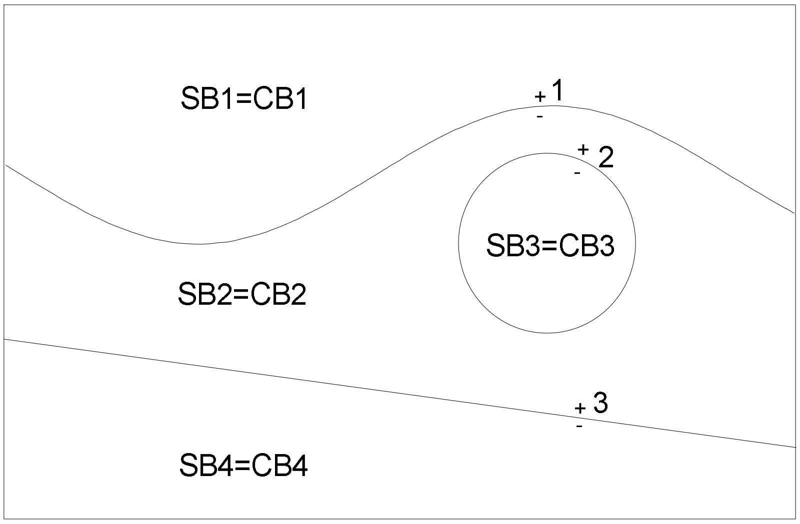


Figure 1: Indices of smooth surfaces, simple blocks and complex blocks in model 98.

2 Model

Model 98 is composed of three layers separated by two interfaces. The upper interface is curved, the lower one is formed by dipping plane. A "salt" body in the middle layer is approximately a sphere. See Figure 1.

The computer representation of the model 98 is contained within file `98-mod.dat`.

file `98-mod.dat` beginning

```
'Simple artificial model created during the 1998 SW3D meeting' /
0  1  1  -3 /         (Cartesian coordinates, velocities, loss factors)
0 5  0 5  0 4         (boundaries of the model)
3 SURFACES
```

The second line might contain just "/" (slash), because the first 3 integers have their default values and the fourth integer is currently not used by any program. Note that some programs can work only in Cartesian coordinates. The usage of velocities and loss factors at this place is highly recommended. The conversion between various powers of velocities or loss factors can be mediated by subroutines specifying the material properties.

The model box is given by $x_1 \in <0\,\mathrm{km}, 5\,\mathrm{km}>$, $x_2 \in <0\,\mathrm{km}, 5\,\mathrm{km}>$ and $x_3 \in <0\,\mathrm{km}, 4\,\mathrm{km}>$, where $x_3 = 0\,\mathrm{km}$ at the top and $x_3 = 4\,\mathrm{km}$ at the bottom. The model box is informative, it may be modified (e.g., extended) for ray tracing, finite differences, or other computations.

Since we have not introduced free Earth surface, there are only 3 smooth surfaces in the model, separating 4 simple blocks. The simple blocks are limited by smooth surfaces. The surfaces limiting a simple block cannot penetrate inside the simple block.

The chosen indices of smooth surfaces, simple blocks and complex blocks are shown in Figure 1. Smooth surfaces 1 and 3 are to be specified as zero isosurfaces of functions like $f(x_1, x_2, x_3) = w(x_1, x_2) - x_3$. The functions are thus negative below the corresponding surfaces, because $x_3 = w(x_1, x_2)$ at the surface and $x_3 > w(x_1, x_2)$ below the surface. The function describing smooth surface 2 is to be specified in the form of $f(x_1, x_2, x_3) = w(x_1, x_2, x_3)$. It is assumed to be negative inside the inclusion and positive outside. It its very important to correctly determine the positive and negative sides of each surface. The signs of the functions are also shown in Figure 1.

file `98-mod.dat` continued

```
4 SIMPLE BLOCKS:
 1 /                    (water)
-1  2  3 /              (middle layer)
-2 /                    (spherical inclusion)
-3 /                    (bottom layer)
```

Simple block 1 is situated at the positive side of surface 1 and is limited by no other surface, see Figure 1. Simple block 2 is situated at the negative side of surface 1, positive side of surface 2 and positive side of surface 3. Simple block 3 is situated at the negative side of surface 2 (inside) and need not be limited by another surface if surface 2 intersects neither surface 1 nor surface 3. For example, if surface 2 were lifted and intersected surface 1, we would have to limit simple block 1 by both surfaces 1 and 2 or limit simple block 3 by both surfaces 1 and 2 in order to prevent intersection of simple blocks 1 and 3.

file `98-mod.dat` continued

```
4 COMPLEX BLOCKS:
 1 /                    (water)
 2 /                    (middle layer)
 3 /                    (spherical inclusion)
 4 /                    (bottom layer)
```

The complex blocks coincide with corresponding simple blocks in this model.

file `98-mod.dat` continued

```
'SURFACE' 1             (curved 2-D interface)
 1 -3 0 0 /                (i.e. W(X1)-X3=0, tension=0)
 6                         (numbers of grid points)
 0   1   2   3   4   5     (X1 grid coordinates)
 1.0 1.5 1.0 0.5 0.7 1.0 (X3 coordinates at grid points)
```

Surface 1 is a 2-D surface, independent of the x_2 coordinate. It is specified in terms of 6 points in the x_1x_3 plane, extending from the left–hand to the right–hand side of the model box. The functions describing the surfaces are interpolated by the natural cubic, bicubic or tricubic splines inside the spline grid. The spline grid may be unique for each function. The analytical continuation is used outside the spline grid, which may cause undesired behaviour of the interpolated function at large distances from the spline grid.

Closed surface 2 has to be specified on a 3-D grid. A spherical surface of radius R, centred at (c_1, c_2, c_3), may be specified in terms of function

$$f(x_1, x_2, x_3) = \sum_{i=1}^{3} \frac{(x_i - c_i)^2}{R^2} - 1 \quad , \tag{1}$$

which may simply be modified for an ellipsoid. Such a quadratic function may be fitted by natural tricubic splines inside a finite box exactly, as described by Bulant & Klimeš (1996). Instead, we decided to enter the gridpoint values of this function and interpolate them by natural tricubic splines for the sake of simplicity. This leads only to a slight distortion of the spherical surface.

file `98-mod.dat` continued

```
'SURFACE' 2          (spherical interface)
 1 2 3 0 /               (i.e. W(X1,X2,X3)=0, tension=0)
 5 5 5                   (numbers of grid points)
 0.0  2.0  3.0  4.0  6.0  (X1 grid coordinates)
-0.5  1.5  2.5  3.5  5.5  (X2 grid coordinates)
-1.5  0.5  1.5  2.5  4.5  (X3 grid coordinates)
```

We chose

$$(c_1, c_2, c_3) = (3.0\,\mathrm{km}, 2.5\,\mathrm{km}, 1.5\,\mathrm{km}) \quad \text{and} \quad R = \sqrt{\frac{2}{3}}\,\mathrm{km} \quad . \tag{2}$$

This value of R looks strange. We likely chose $R = 1\,\mathrm{km}$ first, but the sphere was too big. We then increased the differences of the grid values with respect to the central value of -1 by 50%, which lead to $R = \sqrt{\frac{2}{3}}\,\mathrm{km}$. Note that the first attempt was to use a small, $3\times3\times3$ grid ($x_1 = 2.0\,\mathrm{km}, 3.0\,\mathrm{km}, 4.0\,\mathrm{km}$, $x_2 = 1.5\,\mathrm{km}, 2.5\,\mathrm{km}, 3.5\,\mathrm{km}$, $x_3 = 0.5\,\mathrm{km}, 1.5\,\mathrm{km}, 2.5\,\mathrm{km}$) but the already mentioned analytical continuation outside the grid created spurious parts of simple block 2 at a certain distance from the spherical inclusion. We then extended the grid to cover the whole model box, that is much safer and thus recommended. The grid values below are given by function (1) with (2).

file `98-mod.dat` continued

```
39.5 27.5 26.0 27.5 39.5
27.5 15.5 14.0 15.5 27.5
26.0 14.0 12.5 14.0 26.0
27.5 15.5 14.0 15.5 27.5
39.5 27.5 26.0 27.5 39.5

27.5 15.5 14.0 15.5 27.5
17.5  3.5  2.0  3.5 17.5
14.0  2.0  0.5  2.0 14.0
15.5  3.5  2.0  3.5 15.5
27.5 15.5 14.0 15.5 27.5

26.0 14.0 12.5 14.0 26.0
14.0  2.0  0.5  2.0 14.0
12.5  0.5 -1.0  0.5 12.5
14.0  2.0  0.5  2.0 14.0
26.0 14.0 12.5 14.0 26.0

27.5 15.5 14.0 15.5 27.5
17.5  3.5  2.0  3.5 17.5
14.0  2.0  0.5  2.0 14.0
15.5  3.5  2.0  3.5 15.5
27.5 15.5 14.0 15.5 27.5

39.5 27.5 26.0 27.5 39.5
27.5 15.5 14.0 15.5 27.5
26.0 14.0 12.5 14.0 26.0
27.5 15.5 14.0 15.5 27.5
39.5 27.5 26.0 27.5 39.5
```

Surface 3 is a simple plane, described by a function of the form of $f(x_1, x_2, x_3) = w(x_1, x_2) - x_3$, where function w is bilinearly interpolated from 4 points of intersection with the vertical edges of the model box.

file `98-mod.dat` continued

```
'SURFACE' 3           (dipping interface)
 1 2 -3 0 /  (i.e. W(X1,X2)-X3=0, tension=0)
 2 2         (numbers of grid points)
 0   5       (X1 grid coordinates)
 0   5       (X2 grid coordinates)
 3.0 4.0
 2.0 3.0     (X3 coordinates at grid points)

'END OF SURFACES' /
```

The first complex block is homogeneous (water). The propagation velocities are specified in $\mathrm{km\,s^{-1}}$.

file `98-mod.dat` continued

```
'COMPLEX BLOCK' 1     (water)
'VP ' 1         (P wave velocity)
/               (i.e. VP=W, tension=0)
 1.5
'VS ' 1         (S wave velocity)
/               (i.e. VS=W, tension=0)
 0.0
```

The density is not specified in any block and is thus equal to 1 density unit (default) everywhere.

Complex block 2 is slightly inhomogeneous, with trilinear interpolation of the P–wave velocity between the vertices of the model box. The S–wave velocity is a function of the P–wave velocity and is linearly interpolated, $v_S = 0.6\, v_P$.

file `98-mod.dat` continued

```
'COMPLEX BLOCK' 2     (middle layer)
'VP ' 1         (P wave velocity)
 1 2 3 0 1 /    (i.e. VP=W(X1,X2,X3), tension=0)
 2 2 2          (numbers of grid points)
 0   5          (X1 grid coordinates)
 0   5          (X2 grid coordinates)
 0   4          (X3 grid coordinates)
 2.5 2.6
 2.6 2.5
 3.7 3.6
 3.6 3.6        (velocities at grid points)
'VS ' 1         (S wave velocity)
 4 0 0 /        (i.e. VS=W(X4) with X4=VP, tension=0)
 2              (numbers of grid points)
 0   10         (VP grid coordinates)
 0    6         (VS at the gridpoints)
```

The material properties in complex blocks 3 and 4 are specified analogously to block 2. The spline grid for the spherical inclusion (block 3) covers some neighbourhood of the inclusion but not the whole model box.

file 98-mod.dat continued

```
'COMPLEX BLOCK' 3     (spherical inclusion)
'VP ' 1          (P wave velocity)
 1 2 3 0 1 /     (i.e. VP=W(X1,X2,X3), tension=0)
 2 2 2           (numbers of grid points)
 2.0 4.0         (X1 grid coordinates)
 1.5 3.5         (X2 grid coordinates)
 0.5 2.5         (X3 grid coordinates)
 5.5 5.6
 5.6 5.5
 5.7 5.6
 5.6 5.6         (velocities at grid points)
'VS ' 1          (S wave velocity)
 4 0 0 /         (i.e. VS=W(X4) with X4=VP, tension=0)
 2               (numbers of grid points)
 0   10          (VP grid coordinates)
 0    6          (VS at the gridpoints)

'COMPLEX BLOCK' 4     (bottom layer)
'VP ' 1          (P wave velocity)
 1 2 3 0 1 /     (i.e. VP=W(X1,X2,X3), tension=0)
 2 2 2           (numbers of grid points)
 0   5           (X1 grid coordinates)
 0   5           (X2 grid coordinates)
 0   4           (X3 grid coordinates)
 3.5 3.6
 3.6 3.5
 4.7 4.6
 4.6 4.6         (velocities at grid points)
'VS ' 1          (S wave velocity)
 4 0 0 /         (i.e. VS=W(X4) with X4=VP, tension=0)
 2               (numbers of grid points)
 0   10          (VP grid coordinates)
 0    6          (VS at the gridpoints)

'END OF COMPLEX BLOCKS, END OF THE INPUT DATA FOR THE MODEL' /
```

After the data describing the model are prepared, it is recommended to plot one to several velocity cross–sections through the model as described in Section 3.1 and to check the model consistency by means of program `modchk.for`. The first version of the data specifying the model is usually incorrect!

3 Computations

Following three computations are managed by the *history files*. The history files have been introduced in packages FORMS, MODEL and CRT since version 5.20 and are recommended for the description of calculations. The use of the history files is explained, e.g., by Klimeš (1998).

The three history files, discussed in Sections 3.1—3.3, were tested on a PC under MS DOS, on a PC under LINUX and on an HP workstation under HP UNIX. The history files may be run on all the systems by entering a command of the form of "`perl go.pl historyfile.h`" from the command line. The history files and input data files may be found in subdirectory `98/` of the CRT package on the SW3D-CD-2 disk. When mentioned in the history files, subdirectory `98/` of the CRT package is referred as `crt/98/`, i.e., with respect to the root directory of the SW3D software.

3.1 Velocity cross–sections

Generation of P and S wave velocity cross–sections is managed by history file `98-grd.h`.

file `98-grd.h` beginning

```
# History file '98-grd.h' to generate colour plots of velocity
# crossections across model "98"
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Input files required
 #chk.pl: "crt/98/" "98-mod.dat"
 #chk.pl: "forms/"  "echo.pl"
```

Everything to the right from hash mark "`#`" are comments. The list of files required by the history file has the form of instruction "`chk.pl: "path" "filename"`". It runs Perl script `chk.pl` with input data `"crt/98/" "98-mod.dat"`. Perl script `chk.pl` checks the existence of a file (here `98-mod.dat`) in the current directory. If the path to the root directory of the SW3D-CD-2 software (corresponding to subdirectory `unix/` or `dos/` of the compact disk) is specified and the file exists in its subdirectory (here `crt/98/`), `chk.pl` copies the file into the current directory. If the file is not found, `chk.pl` should stop the execution. However, specification of the path to the root directory of the SW3D-CD-2 software requires to manually edit file `chk.pl` and stopping the execution often does not work in the current version. That is why the instruction is disabled (commented by the hash mark).

file `98-grd.h` continued

```
# Data specifying the model by means of the MODEL package
  MODEL='98-mod.dat'

# Data for gridding and plotting
  N1=501    N2=1       N3=401
  D1=0.01   D2=1       D3=0.01
  O1=0.0    O2=2.5     O3=0.0
  HSIZE=15  YSIGN=-1
  VREF=1.5  VCIRC=-5.6 VMIN=0.
```

The above input data specify the name of the input file `98-mod.dat` defining the model, the coordinates (0.0 km, 2.5 km, 0.0 km) of the origin of the grid, dimensions $501\times1\times401$ of the grid and grid step of 0.01 km between the gridpoints. Horizontal size of the resulting PostScript figure is 15 cm, vertical size is proportional, vertical axis points downwards (YSIGN=-1). Reference velocity, displayed in blue color (default reference colour), is 1.5 km s^{-1}. The extent VCIRC of values corresponding to the whole colour circle RGB is 5.6 km s^{-1}, the colour circle goes from blue to green and red. Zero velocities are rendered as undefined values (VMIN=0), and are displayed in light gray by default.

file **98-grd.h** continued

```
# Two input data files for program 'grid.for'
  echo.pl: "' ' '98-velp.out' 'icbp.out' /" >gridp.tmp
  echo.pl: "/"                              >>gridp.tmp
  echo.pl: "' ' '98-vels.out' 'icbs.out' /" >grids.tmp
  echo.pl: "-1 -2 -3 -4 /"                  >>grids.tmp
```

Perl script echo.pl, which works similarly as UNIX command echo, is used to generate temporary files gridp.tmp and grids.tmp, which are input files of program grid.for. They contain names of output files of the grid.for program in the first line, and the specification of velocities to be gridded in the second line. Thus, P–wave velocities in all the complex blocks (default) will be written to file 98-velp.out, and values of S–wave velocities in complex blocks 1 to 4 will be written to file 98-vels.out.

Note that our history files, unlike SEP (Stanford Exploration Project) history files, may consider Perl scripts, and each instruction to run a program is followed by the input data for the program in the form of "program: data". These modifications enable to include the programs, which have not been designed for the history files (like grid.for or crt.for), in the history files.

file **98-grd.h** continued

```
# Programs to run
  grid:  '98-grd.h' 'gridp.tmp' /                    # Gridding P velocities
  grdps: '98-grd.h' '98-velp.out' '98-velp.ps' / # Plotting P velocities
  grid:  '98-grd.h' 'grids.tmp' /                    # Gridding S velocities
  grdps: '98-grd.h' '98-vels.out' '98-vels.ps' / # Plotting S velocities
```

Program grid.for is run to generate the gridded velocities, and program grdps.for to generate PostScript figures of the gridded values, see Figures 2 and 3.

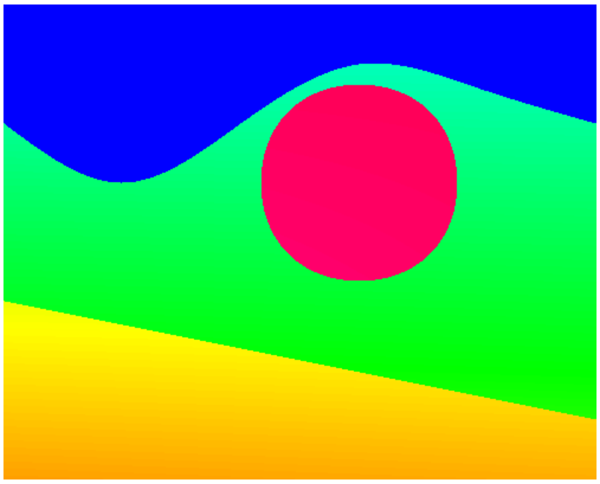


Figure 2: P–wave velocity cross–section through the middle of the model 98. P–wave velocity is lowest at the top water layer and highest inside the circular salt body.

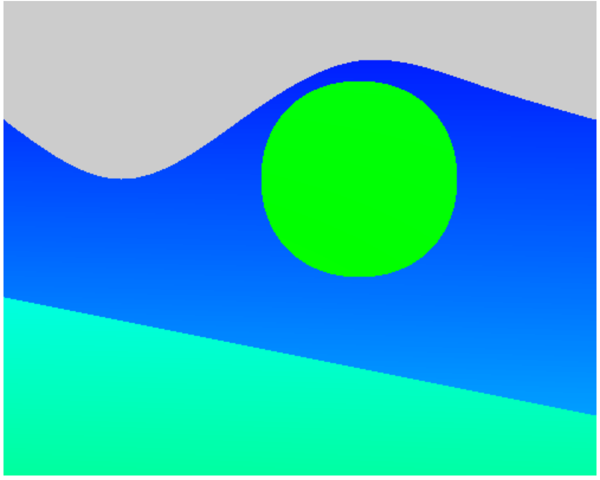


Figure 3: S–wave velocity cross–section through the middle of the model 98. S–wave velocity equals zero in the first layer and it is a linear function of P–wave velocity in other layers.

3.2 Ray–theory travel times on a rectangular grid

The point source is located in the centre of the top of the model, 0.001 km below the top of the model box, see Figure 8. Its name (formal string) and coordinates are stored in data file 98-src.dat.

file 98-src.dat

```
/
'source' 2.500 2.500 0.001 /
/
```

The ray–theory travel times are computed at the nodes of the grid already used for velocity cross–sections. A system of rays of the refracted elementary wave is computed in the first step, and the travel times are interpolated between the rays in the second step.

Main input data for ray tracing program crt.for are located in file 98-crt1.dat.

file 98-crt1.dat beginning

```
/
'98-mod.dat'  (MODEL)
' '           (DCRT)
' '           (INIT)
' '           (CODE)
'98-rpa1.dat' (RPAR)
'writall.dat' (WRIT)
'98-crt1.out' (LOG)
```

Data sets DCRT (numerical parameters for ray tracing), INIT (initial conditions for ray tracing) and CODE (codes of elementary waves) will follow in file 98-crt1.dat. Input data describing the model are in file 98-mod.dat, input data for ray parameters (i.e., data controlling shooting of rays) are read from file 98-rpa1.dat, which is shown below. All points along the rays will be written to the unformatted output files according to the input file writall.dat, the description of the computation will be reported to the output log file 98-crt1.out.

file 98-crt1.dat continued

```
'DCRT'
/                    (KSTORE, NEXPS, NHLF, MODCRT)
0.05 0.1 0.00005 0.00001 0.00001 0.00001 0.000 / (STORE,STEP,UEB,...)
-0.1 5.1 -0.1 5.1 -2 6 / (computational volume)
/                    (no auxiliary surface defined within this data set)
/                    (no end surfaces)
/                    (no storing surfaces)
/                    (all points of intersection)
/                    (all points of intersection)
```

Data set DCRT contains numerical parameters to trace a single ray. The first line of data DCRT contains just an arbitrary comment. Since we use the default values of the integer quantities controlling numerical integration, "/" (slash) is sufficient in the second line.

The third line begins with the step in independent variable (travel time) for storing the computed quantities along a ray. The step determines the length of ray cells for the interpolation. We have chosen STORE=0.05 (the value of STORE=0.2, used during the consortium meeting, was too large). The following quantity is maximum increment

STEP of independent variable for numerical integration, which is expressed in the travel–time units if we use default value of NEXPS=0 (NEXPS=0 is highly recommended). The increment should not exceed the minimum value of $|\mathrm{grad}\ v|^{-1}$ in the model, which is large in this case. Simultaneously, the step along a ray should be sufficiently smaller than the minimum grid interval for spline interpolation of velocities. In our case of a trilinear spline interpolation, the value of STEP=0.1 (in seconds), corresponding to the maximum step about 0.56 km at maximum velocity about 5.6 km s^{-1}, is selected mostly with respect to the dimensions of the simple blocks in the model. If STORE were zero or larger than STEP, STEP should be sufficiently small with respect to the interface curvature.

Upper error bound UEB of travel time per one step of numerical integration cannot be too small with respect to the maximum travel time or model coordinates because of rounding errors. Originally, we set the upper error bound of travel time per one step of numerical integration to UEB=0.00001 (in seconds) on a PC. In our case of travel times up to 2 s or 3 s, this value turned out too small on a RISC workstation and has been increased to UEB=0.00005. Remaining values in the third line of data set DCRT do not influence the computation itself, they just control generation of warnings to the output log file if the error limits are exceeded.

We have increased the computational volume with respect to the model box, in order to cover the corners of the interpolation grid by ray cells without considerably increasing the number of rays, see the description of parameter PRM0(3) of input data 98-rpa1.dat below.

file 98-crt1.dat continued

```
'INIT'
-1 3 /              (single initial point, azimuth.equidist.proj.)
'98-src.dat' /      (file with coordinates of the initial point)
```

Data set INIT, containing the quantities defining the kind of initial conditions, specify INIDIM=-1, which means a point source, whose coordinates are read from the separate file 98-src.dat. The value of INIPAR=3 sets the azimuthal equidistant projection of a unit sphere parametrized by 2 Cartesian coordinates centred at the projection point. This option with just a single singularity is, in general, better than the spherical coordinates which have two singularities. It is especially suitable for reflection seismic studies, when the point source is located at the Earth's surface. The unit vector, tangent to the ray with initial parameters $(0, 0)$ at the source, points downwards in this parameterization.

file 98-crt1.dat continued

```
'CODE'
/
5  1 1 1 1 1 1 1 /  (refracted wave)
/
```

In the input data CODE, we have selected KODTYP=5 (transmission–reflection code) for the first and only elementary wave. For any element of a ray, one integer is used. We chose P wave at most 6 times transmitted at an interface.

The take–off parameters of rays are controlled by input data file 98-rpa1.dat.

file `98-rpa1.dat` beginning

```
'Ray parameters.' /

 105 -1 -2  0   0.001  0.02 2* 1 / (ISRFR,ISRFX,ISRFY,NREC,XERR,AERR)
```

Values `ISRFR=105`, `ISRFX1=-1` and `ISRFX2=-2` determine the reference surface at which the receivers are situated and the coordinates along the reference surface parametrizing the receivers. Since we need no receivers for controlled initial–value ray tracing (`NREC=0`), the values of `ISRFR`, `ISRFX1` and `ISRFX2` have no meaning here, with the following exception: we need nonzero values of `ISRFX1` and `ISRFX2` for controlled initial–value ray tracing. Zero values would lead to simple initial–value ray tracing forming a rectangular grid in the normalized ray–parameter domain, without determining boundaries between ray histories. We thus use here the values of `ISRFR=105`, `SRFX1=-1` and `ISRFX2=-2` corresponding to the next example of two–point ray tracing to the receivers situated at the top of the model box.

Accuracy `XERR=0.001` of two–point ray tracing has no meaning here. Accuracy `AERR=0.02` of the determination of the boundaries between ray histories is very important and will be discussed below.

The last number in the third line, `PRMO(3)=1`, means that boundaries of the computational volume do not influence ray histories. This option reduces number of thin strips between ray histories, in which travel times are missing. On the other hand, this option causes that the edges of the computational volume are not covered by ray cells. The computational volume thus has to be enlarged with respect to the dimensions of the interpolation grid, see the computational volume in section DCRT of file `98-crt1.dat`. We have enlarged the computational volume mostly at the top and bottom of the model box where there are no structural interfaces, and we thus save the computational time.

file `98-rpa1.dat` continued

```
-1.5707 -0.6  1.5707 -0.6  -1.5707  0.6   90 45    /
/
```

The first six numbers are the coordinates of the corners of the ray–parameter domain. We have selected angles from $-\pi/2$ to $\pi/2$ in the x_1x_3 plane in order to cover the whole interpolation grid by ray tubes. Values from -0.6 to 0.6 in the x_2x_3 plane should be sufficient to catch all the rays, which might deviate from the x_1x_3 plane and then return back to the plane containing the interpolation grid. In fact, the configuration of the model does not allow for such rays, and we thus might selected, e.g., values -0.1 to 0.1 instead of -0.6 to 0.6.

The last two numbers in file `98-rpa1.dat`, parameters `ANUM` and `BNUM`, are used to specify the metric tensor in the ray–parameter domain. The ray–parameter domain is covered by basic triangles, which are roughly equilateral in the given metric. Here `ANUM` controls the number of basic rays shot in the x_1x_3 plane across the whole ray–parameter domain. `BNUM` corresponds to the number of basic rays shot in the x_2x_3 plane, and equals half an `ANUM`, because the dimension of the ray–parameter domain in the x_2x_3 plane is about half of its dimension in the x_1x_3 plane.

Let us return to the accuracy `AERR=0.02` of the determination of the boundaries between ray histories. In our case, the width of a thin strip between two different ray histories, in which the travel times can not be interpolated, is kept below approximately

$$\frac{\pi}{90} \times 0.02 \approx 0.0007 \tag{3}$$

radians.

The travel–time computation is managed by history file `98-mtt.h`.

file `98-mtt.h` beginning

```
# History file '98-mtt.h' to compute multivalued ray-theory
# travel times in nodes of 2-D grid in model "98" and
# to generate colour figures of the travel-time maps
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Input files required
 #chk.pl: "crt/98/" "98-crt1.dat"
 #chk.pl: "crt/98/" "98-mod.dat"
 #chk.pl: "crt/98/" "98-src.dat"
 #chk.pl: "crt/98/" "98-rpa1.dat"
 #chk.pl: "crt/"    "writall.dat"
 #chk.pl: "forms/"  "echo.pl"

# Grid dimensions
  N1=251   N2=1    N3=201
  D1=0.02  D2=1    D3=0.02
  O1=0.0   O2=2.5  O3=0.0
```

The ray–theory travel times are computed at the nodes of the grid already used for velocity cross–sections.

file `98-mtt.h` continued

```
# Ray tracing of refracted wave ('crt.for' cannot include 'scropc.for')
  crt: '98-crt1.dat' /
```

The system of rays, creating ray tubes, is traced by program `crt.for` with main input data file `98-crt1.dat`.

file `98-mtt.h` continued

```
# Main input data file for program 'mtt.for'
  echo.pl: "'r01.out'  'r01i.out' 't01.out' /"  >mtt.tmp
  echo.pl: "'98-num.out' /                     " >>mtt.tmp
  echo.pl: "'98-tt.out' 'tt' /                 " >>mtt.tmp
  echo.pl: "'98-hi.out' 'hi' /                 " >>mtt.tmp

# Interpolation of travel times within ray cells
  mtt: '98-mtt.h' 'mtt.tmp' /
```

Perl script `echo.pl` is used to generate temporary input file `mtt.tmp` for program `mtt.for`. The file specifies the names of files with rays computed by `crt.for`, name `98-num.out` of the output file with numbers of travel times found at the gridpoints, name `98-tt.out` of the output file with the travel times interpolated at the gridpoints and name `98-hi.out` of the output file with the values of ray histories corresponding to the travel times at the gridpoints.

file `98-mtt.h` continued

```
# Plotting numbers of arrivals at the gridpoints
  HSIZE=15  YSIGN=-1
  VPLUS=2  CREF=0.166667
  grdps: '98-mtt.h' '98-num.out' '98-num.ps' /
```

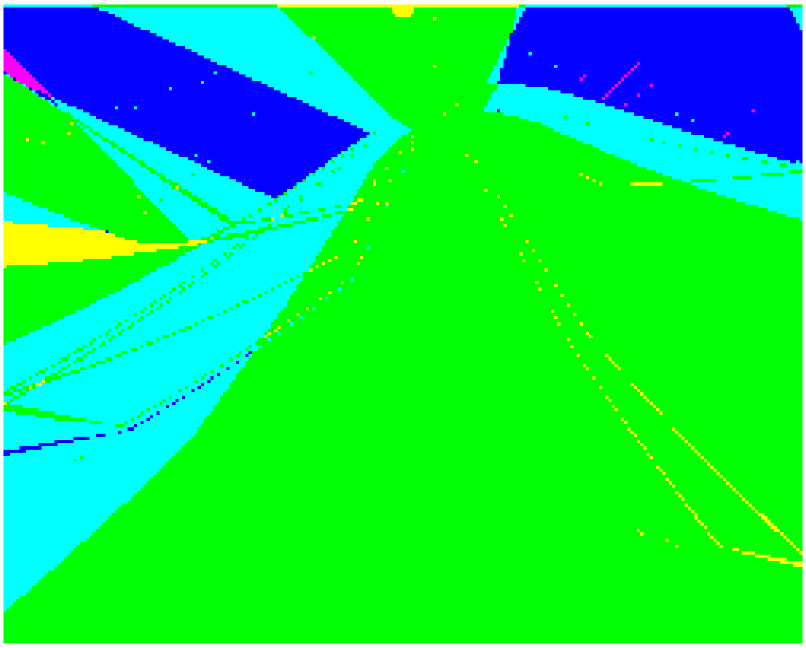


Figure 4: Numbers of computed elementary travel times.

Program `grdps.for` is used to create the PostScript figure of the numbers of computed travel times at the gridpoints, see Figure 4.

file `98-mtt.h` continued

```
# Plotting grid travel times sorted according to their value
  VCIRC=0.2  VREF=0  CREF=0.166667
  mgrd>mgrd.out: '98-mtt.h' '98-num.out' '98-tt.out' '98-tt000.out' /
  grdps: '98-mtt.h' '98-tt000.out' '98-tt001.ps' /
```

The computed multivalued travel–times are sorted according to their values (i.e., first arrival to the gridpoints, second arrival to the gridpoints, ...) using program `mgrd.for`. The sorted singlevalued travel–time maps are then plotted by program `grdps.for` in four PostScript files `98-tt001.ps` to `98-tt004.ps`, see Figure 5. The yellow colour (`CREF=0.166667`) is repeated every 0.2 s (`VCIRC=0.2`) starting at the source (`VREF=0`). Note that the number of arrivals may be influenced by the computer and compiler.

file `98-mtt.h` continued

```
# Plotting grid travel times sorted according to the histories
  mttgrd>mttgrd.out: '98-mtt.h' '98-num.out' '98-hi.out' '98-tt.out' '98-tt100.out' /
  grdps: '98-mtt.h' '98-tt100.out' '98-tt101.ps' /
```

As an alternative, the computed multivalued travel–time map is sorted according to the ray histories and subsequently according to the arrival times into singlevalued travel–time maps using program `mttgrd.for`, and then plotted by program `grdps.for`, see Figure 6. Output PostScript files of `grdps.for` with the sorted singlevalued travel–time maps are named consecutively from `98-tt101.ps` to `98-tt118.ps`. Note that their number may be influenced by the computer and compiler.

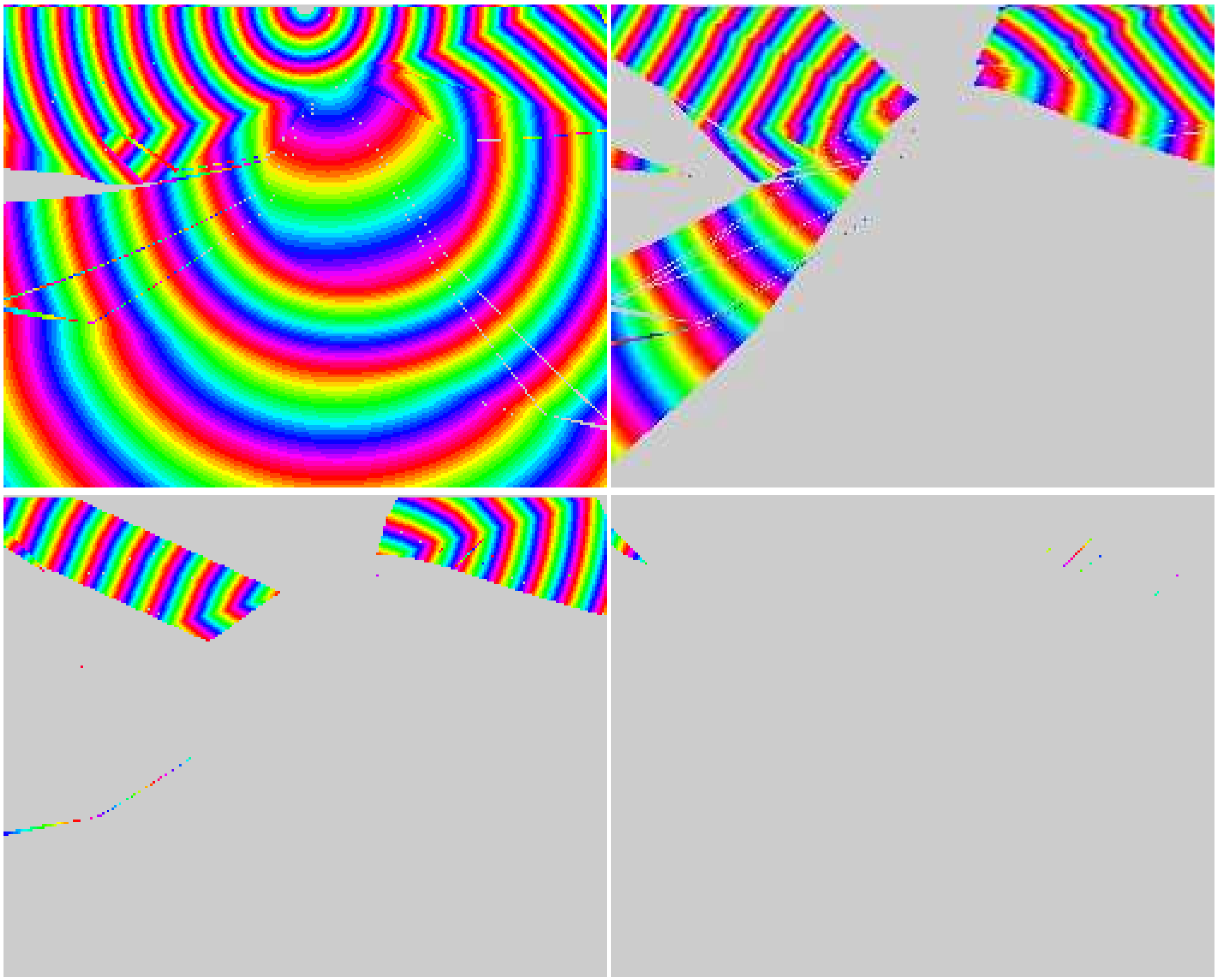


Figure 5: Computed singlevalued travel–time maps. Travel times are sorted according to their value.

3.3 Seismogram generation

The ray–theory synthetic seismograms are computed at 16 receivers located at the diagonal line at the top of the model, for the same point source as was used during the travel–time computation. Coordinates of the point source and of the receivers may be found in data files `98-src.dat` and `98-rec.dat`.

file `98-rec.dat`

```
/
'rec10' 1.0 1.0 0 /
'rec12' 1.2 1.2 0 /
'rec14' 1.4 1.4 0 /
'rec16' 1.6 1.6 0 /
'rec18' 1.8 1.8 0 /
'rec20' 2.0 2.0 0 /
'rec22' 2.2 2.2 0 /
'rec24' 2.4 2.4 0 /
'rec26' 2.6 2.6 0 /
'rec28' 2.8 2.8 0 /
'rec30' 3.0 3.0 0 /
'rec32' 3.2 3.2 0 /
'rec34' 3.4 3.4 0 /
'rec36' 3.6 3.6 0 /
```

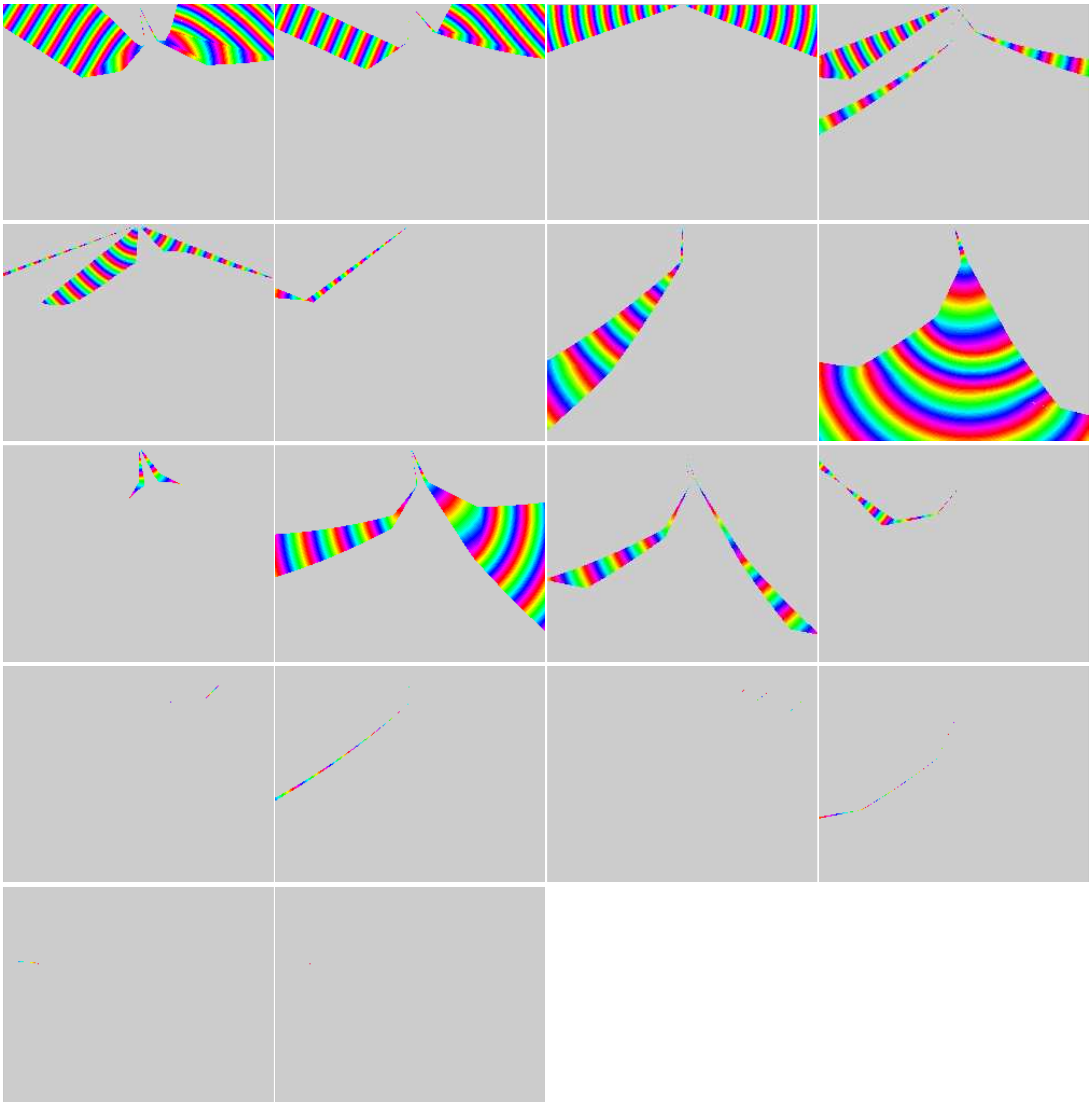


Figure 6: Computed singlevalued travel–time maps. Travel times are first sorted according to the ray histories and then according to their value.

```
'rec38' 3.8 3.8 0 /
'rec40' 4.0 4.0 0 /
/
```

The position of the source and of the receivers in the model may be observed, together with two–point rays, in Figure 8.

Two–point rays of four reflected elementary waves are traced by program `crt.for` with main input data file `98-crt2.dat`.

file 98-crt2.dat

```
/
'98-mod.dat'  (MODEL)
' '           (DCRT)
' '           (INIT)
' '           (CODE)
'98-rpa2.dat' (RPAR)
'writ.dat'    (WRIT)
'98-crt2.out' (LOG)

'DCRT'
/                    (KSTORE, NEXPS, NHLF, MODCRT)
0.2 0.1 0.00005 0.00001 0.00001 0.00001 0.000 / (STORE,STEP,UEB,...)
/                    (computational volume is the model box)
/                    (no auxiliary surface defined within this data set)
/                    (no end surfaces)
105 /                (storing surfaces)
/                    (all points of intersection)
/                    (all points of intersection)

'INIT'
-1 3 /               (single initial point, azimuth.equidist.proj.)
'98-src.dat' /       (file with coordinates of the initial point)

'CODE'
/
5  1 2 1 1 1 1 1 1 / (reflected wave)
5  1 1 2 1 1 1 1 1 / (reflected wave)
5  1 1 1 2 1 1 /     (reflected wave)
5  1 1 1 1 2 1 1 1 / (reflected wave)
/
```

The above data are similar to the data for the travel–time interpolation. According to the input file `writ.dat`, only two–point rays are stored in the unformatted output files. Parameter `STORE` is increased to 0.2 s, because the points stored along the two–point rays are used just for plotting. The top of the model box (surface 105) is specified as the storing surface for the computed travel times and other quantities. Let us note, that the index of the bottom of the model box is 106 and the indices of the sides are 101 to 104. Four reflected elementary waves are specified by four ray codes. All codes are the transmission–reflection codes, describing the P waves reflected at the first, second, third and fourth interface, respectively. The third code describes rays with just six segments, because we consider only the rays satisfying the whole ray code in the computation of the third wave, see the discussion of parameter `IPOINT` in data `98-rpa2.dat` below. The length of other three codes is not crucial, it must only be sufficient to allow rays to reach the receivers.

The take–off parameters of rays are controlled by input data file `98-rpa2.dat`.

file `98-rpa2.dat`

```
'Ray parameters.'

 105 -1 -2 -1   0.001  0.200   / (ISRFR,ISRFX,ISRFY,NREC,XERR,AERR)
'98-rec.dat'

-1.2  -0.6     0.9  -0.6    -1.2   0.6      42 24   1 /
/
-1.2  -0.6     0.9  -0.6    -1.2   0.6      42 24   1 /
/
-1.2  -0.6     0.9  -0.6    -1.2   0.6      42 24  -1 /
/
-0.2  -0.65    0.9  -0.65   -0.2   0.65     42 24   1 /
/
```

We can see, that the above data are similar to the data for the travel–time interpolation. Surface `ISRFR=105` is the reference surface at which the receivers are situated, `ISRFX1=-1` and `ISRFX2=-2` determine the x_1 and x_2 coordinates as the parameters along the reference surface. `NREC=-1` means, that the coordinates of receivers can be found in file `98-rec.dat`.

Accuracy `XERR=0.001` of the two–point ray tracing means, that the distance of the cross–section of a two–point ray with the reference surface from the corresponding receiver should not exceed 0.001 km. If such a two–point ray cannot be found, the ray nearest to the receiver is taken instead and a warning to the output log file `98-crt2.out` is generated. This is usually the case of weakly refracted rays, see Figure 8. Accuracy `AERR=0.2` of the determination of the boundaries between ray histories is not so critical here, and that is why we increased this parameter compared to the travel–time interpolation.

One of the ways, how to set an efficient ray parameter domain, is to plot the ray take–off parameters in the ray parameter domain using program `rpplot.for`. The plots of ray take–off parameters, coloured according to the ray histories, showed that the whole half space need not be illuminated and the extents of the ray parameters can be reduced from $< -\pi/2, \pi/2 > \times < -\pi/2, \pi/2 >$ to, e.g., $< -1.2, 0.9 > \times < -0.6, 0.6 >$ for the first 3 elementary waves and to $< -0.2, 0.9 > \times < -0.65, 0.65 >$ for the fourth elementary wave.

We would like to bring a reader's attention to the parameter `IPOINT=-1` at the end of the 10^{th} line of file `98-rpa2.dat`. Originally, the rays refracted under the first interface were computed during both the computations of the third and of the fourth elementary waves, because these rays satisfy the part `1 1 1` of the ray code, which is common to the third and the fourth elementary waves. We thus set `IPOINT=-1` for the third wave, which admits only the rays satisfying the whole ray code as two–point rays.

Program `rpplot.for` should be used each time, when we wish to reduce the ray–parameter domain and not to shoot rays to all possible directions (to the whole half space in our case). We must be sure, that we shall not omit any two–point ray, before we reduce the ray–parameter domain. As an example of using the `rpplot.for` program, we generate the plot of the ray–parameter domain of the fourth elementary wave running , see Figure 7. Main input data for program `rpplot.for` are stored in file `98-plot1.dat`.

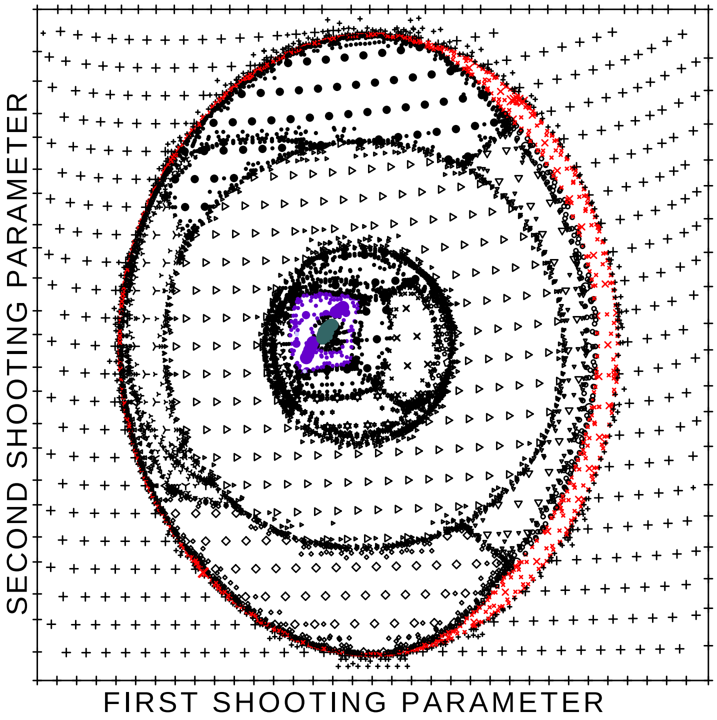


Figure 7: Ray–parameter domain of the fourth elementary wave. Successful rays (rays which crossed the reference surface) are coloured according to their history, other rays are black.

file `98-plot1.dat`

```
1 1 1 0 1 0 1 / IRBAS,IRTWO,IRAUX,ITHOM,ISANG,ISHP,ISUC
0. 1. 0. 1. / PLIMIT
 .2 .3 .1 5* '98-plot1.ps' / HRBAS,HRTWO,HRAUX,HOR,VER,HTEXT,'SYMBOLS','COLORS','PLOT'
's01i.out' 't04.out' 4 / 'INITIALPOINTS','TRIANGLES',IWAVE
/ 'RECEIVERS',ISREC,ICREC,HREC
```

The first four integers in the first line determine, whether corresponding objects (i.e., rays or triangles) are to be plotted or not. We decided to plot basic rays (`IRBAS=1`), to plot two–point rays (`IRTWO=1`), to plot auxiliary rays (`IRAUX=1`), and not to plot homogeneous triangles (`ITHOM=0`). In short, we plot all rays and no triangles. `ISANG=1` means that we plot the ray–parameter domain, the other possibility is to plot the reference surface. `ISHP=0` means, that the objects of all ray histories will be plotted. The other possibility is to plot only the objects of one selected ray history. The last parameter in the first line is `ISUC`, which influences colours of the objects being plotted. For `ISUC=1`, only the rays and triangles of successful histories (i.e., rays passing the reference surface) are coloured and the other objects are black. Note that the default value of `ISUC=0` does not influence the colours.

The second line contains the limits of the plot. For `ISANG=1`, these are the normalized ray parameters from domain $< 0,1 > \times < 0,1 >$, for `ISANG=0` these would be coordinates on the reference surface. The third line begins with three parameters controlling the height of the symbols used for plotting the basic rays, the two–point rays and the auxiliary rays. They are followed by five defaults for the dimensions of the resulting picture, for the height of the characters, and for the selection of the symbols and colours according to the ray histories. The last string in the third line is the name of the output PostScript file.

The fourth line contains the names of CRT output files with rays and triangles and one integer identifying, which elementary wave to plot. In our case, `s01i.out` is the file with initial points of all the rays, `t04.out` is the file with homogeneous triangles of the fourth wave, and the fourth elementary wave is selected. The last line controls plotting receivers on the reference surface and has no meaning here, we thus use default.

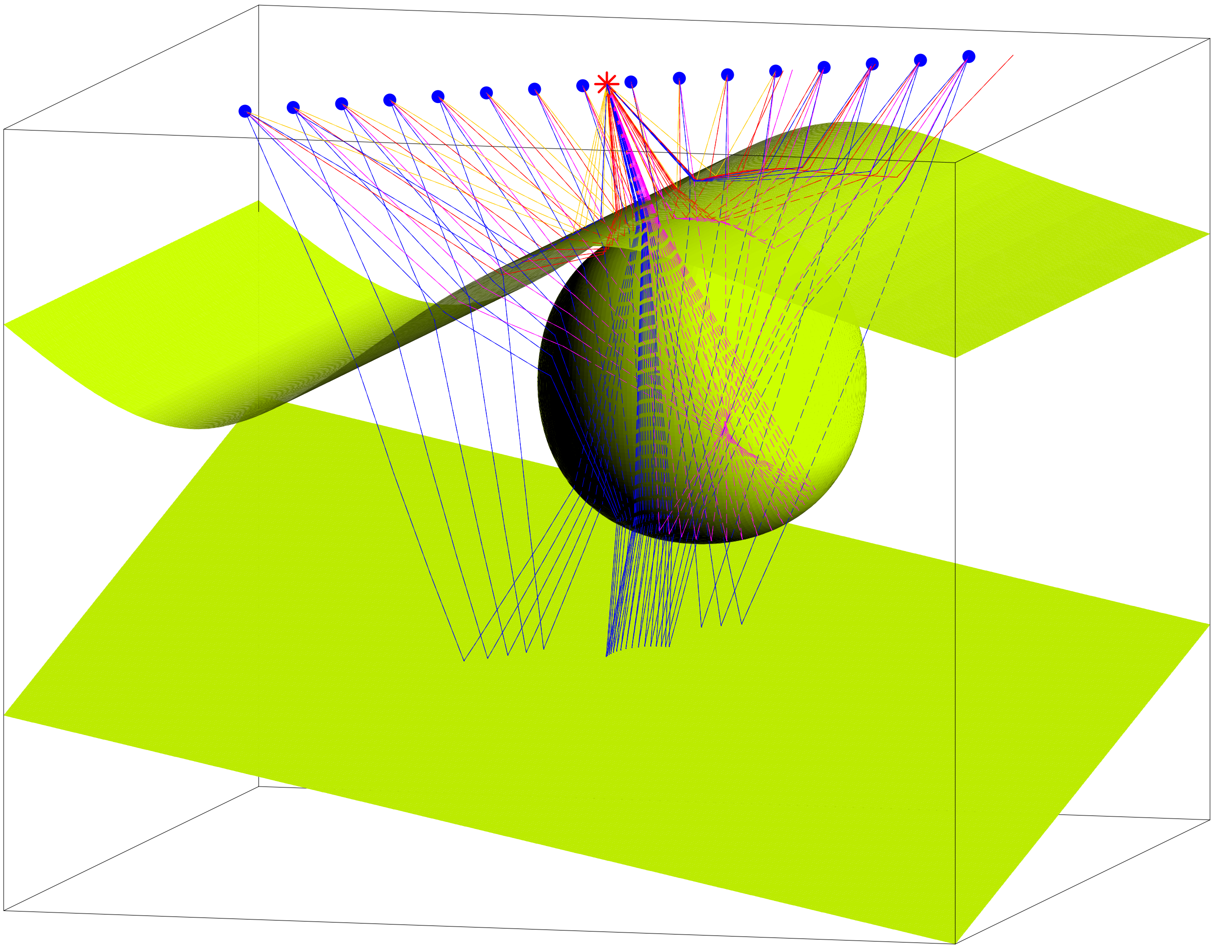


Figure 8: Two–point rays of the four elementary waves in the model 98.

The seismogram computation and plotting is managed by the history file `98-ss.h`.

file `98-ss.h` beginning

```
# History file '98-ss.h' to compute synthetic seismograms in model 98
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Input files required
 #chk.pl: "crt/98/" "98-crt2.dat"
 #chk.pl: "crt/98/" "98-mod.dat"
 #chk.pl: "crt/98/" "98-plot1.dat"
 #chk.pl: "crt/98/" "98-src.dat"
 #chk.pl: "crt/98/" "98-rec.dat"
 #chk.pl: "crt/98/" "98-rpa2.dat"
 #chk.pl: "crt/"    "writ.dat"

# Source and receivers
  SRC='98-src.dat'  REC='98-rec.dat'

# Data describing the source time function
  KSIG=1 # (Gabor signal)
```

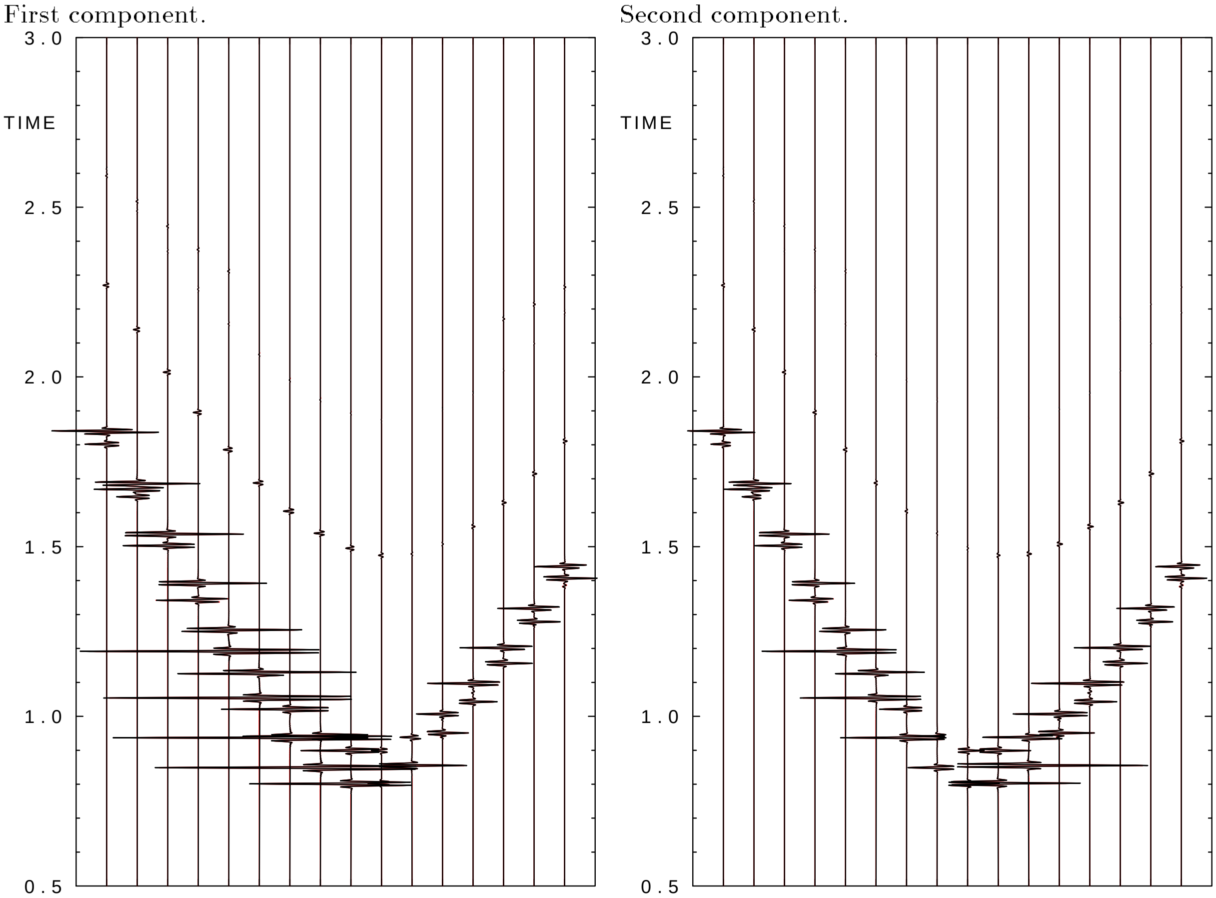


Figure 9: Time–domain synthetic seismograms (red) overlaid by the frequency domain synthetic seismograms (black). The figures are rather small but the corresponding PostScript files may be found in subdirectory **crt/98/** on SW3D-CD-2.

```
  SIGT=0.  SIGF=100.  SIGW=4.  SIGPH=0.  SIGA=1.

# Data describing the filtration of the source time function
  FMIN=4.  FLOW=20.  FHIGH=184.  FMAX=200. # (cosine filter)

# Time step and time interval for the Fast Fourier Transform
  DT=0.001  NFFT=2048

# Data to control optional plotting of program SS
  SIGPLOT='98-sig.ps'  MPTS=100
```

The reference time of the Gabor source time function (i.e., the centre of the envelope) is `SIGT=0`, the reference frequency is 100 Hz, relative width of the signal is `SIGW=4` and phase `SIGPH=0` (symmetric Gabor signal), amplitude of the envelope of the signal is `SIGA=1`. Frequency filter applied to the source time function is zero for frequencies smaller than `FMIN` or greater than `FMAX`. The filter is 1 between `FLOW` and `FHIGH`. Between `FMIN` and `FLOW`, and between `FHIGH` and `FMAX`, cosine tapering is used. Time interval to digitize the source time function and seismograms is `DT=0.001` (10 samples per the reference period) and the number of the time samples for the fast Fourier transform is `NFFT=2048`, which corresponds to the time interval 2.048 seconds. The time interval is centred in the middle between the first and last travel time by default.

Third component.

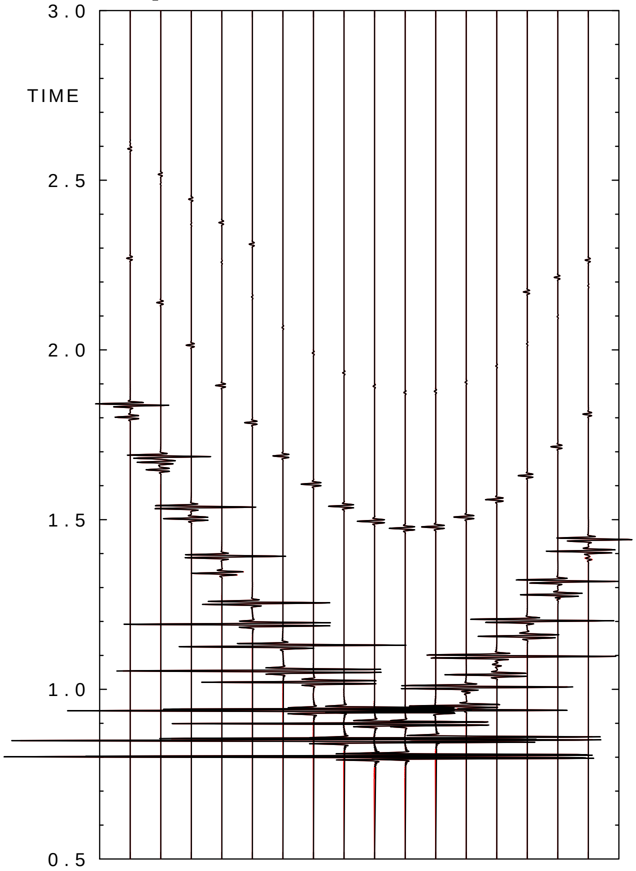


Figure 9 — continuation.

file `98-ss.h` continued

```
# Data to control seismogram plotting (program SP)
  KODESP=2    SPCHRH=0.40
  SPTMIN=0.5  SPTMAX=3.0  SPTLEN=25.0  SPTDIV=5    SPTSUB=5
  SPXMIN=0.8  SPXMAX=4.2  SPXLEN=15.3  SPXDIV=-17  SPXSUB=1
  SPYMIN=0.8  SPYMAX=4.2
  NORMSP=1    SPAMP=-2500
```

Above data control the plot of the synthetic seismograms, see program `sp.for` and also Klimeš (1998). Note that we disabled plotting of coordinate values by negative `SPXDIV` because it is bad in version 5.20 and should be improved.

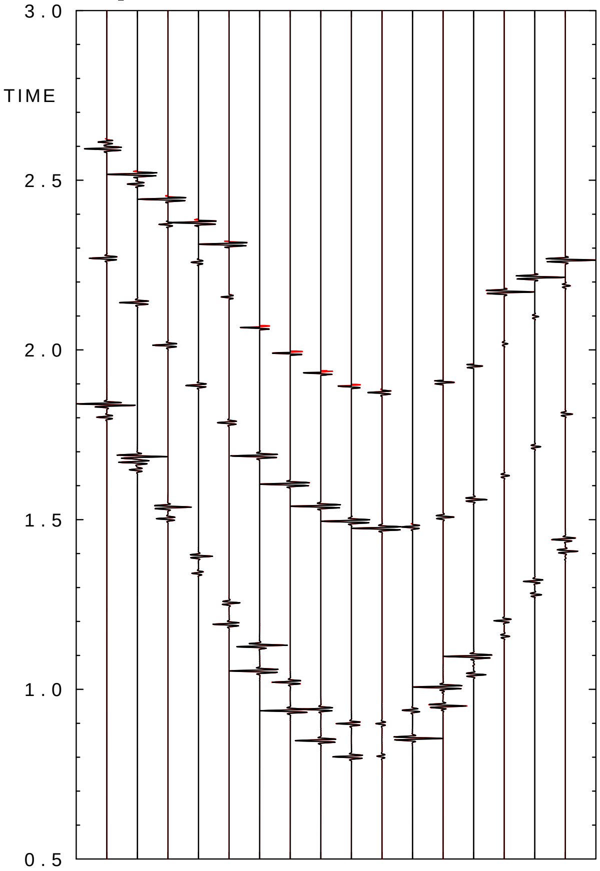
file `98-ss.h` continued

```
# Programs to calculate Green function
  crt:     '98-crt2.dat' /  # two-point ray tracing
                            # (do not link 'crt.for' with 'scropc.for'!)
  rpplot:  '98-plot1.dat' / # ray parameters plotting
  green:   '98-ss.h' /      # Green function
```

Two–point rays of four reflected elementary waves are computed by program `crt.for` with the main input data file `98-crt2.dat`. Output of `crt.for` is converted to the elementary Green functions using program `green.for`.

As it is quite difficult to judge the obtained results only according to the generated synthetic seismograms, we decided to check them in more detail. We have generated the

First component. Second component.



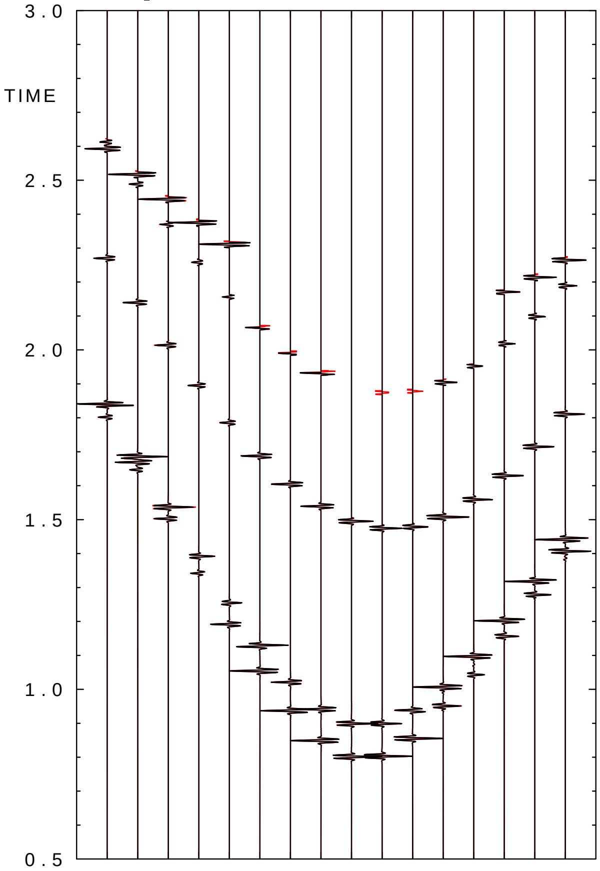


Figure 10: Time–domain synthetic seismograms (red) overlaid with the frequency domain synthetic seismograms (black). Exponential amplitude scaling with travel time, normalized maximum amplitudes at each trace. The figures are rather small but the corresponding PostScript files may be found in subdirectory `crt/98/` on SW3D-CD-2.

3-D plot of the two–point rays computed in the model. Yet unreleased graphic package D3 has been used to generate Figure 8.

file `98-ss.h` continued

```
# Frequency-domain seismograms
  SS='98-ssf.gse'  SP1='98-ssf1.ps'  SP2='98-ssf2.ps'  SP3='98-ssf3.ps'
  greenss: '98-ss.h' /      # response function
  ss:      '98-ss.h' /      # synthetic seismograms
  sp:      '98-ss.h' /      # seismogram plotting
```

Frequency–domain response function is calculated by program `greenss.for`. The "frequency–domain" synthetic seismograms are then computed by program `ss.for` (file `98-ssf.gse`) and plotted by program `sp.for` (pictures `98-ssf1.ps`, `98-ssf2.ps` and `98-ssf3.ps`).

file `98-ss.h` continued

```
# Time-domain seismograms
  SIGGSE='98-sig.gse' SS=' '
  ss:      '98-ss.h' /      # source time function
  SS='98-sst.gse'  SP1='98-sst1.ps'  SP2='98-sst2.ps'  SP3='98-sst3.ps'
  greenss: '98-ss.h' /      # synthetic seismograms
  sp:      '98-ss.h' /      # seismogram plotting
```

Third component.

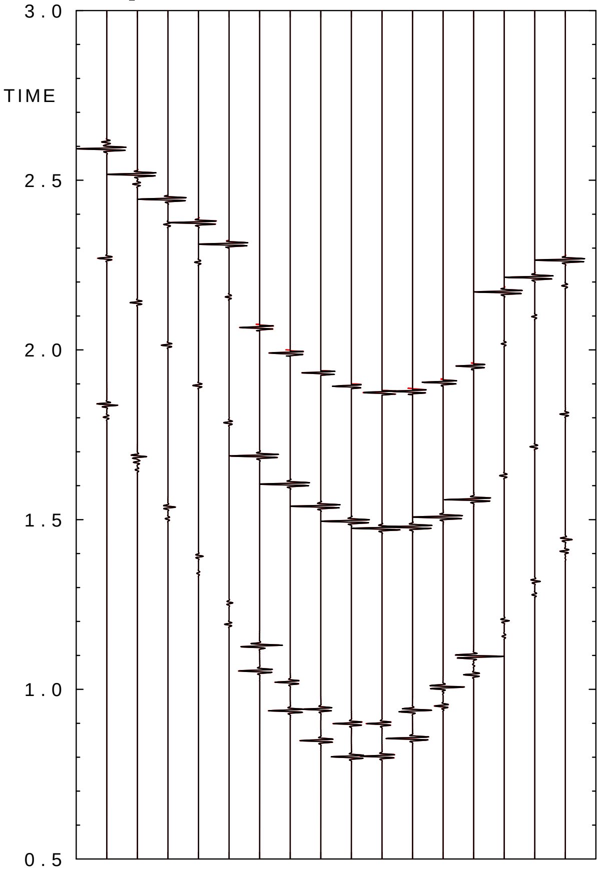


Figure 10 — continuation.

The source time function and its Hilbert transform is written by program `ss.for` to GSE file `98-sig.gse`. Synthetic seismograms are then composed in the time domain by program `greenss.for` (file `98-sst.gse`) and plotted by program `sp.for` (pictures `98-sst1.ps`, `98-sst2.ps` and `98-sst3.ps`). As recalled below in detail, the computation in the time domain is not recommended because it can consider neither an attenuation, responses of stacks of fine layers nor a weak anisotropy. It has been enabled mainly for debugging purposes and is demonstrated here at the request of the meeting participants.

The "time–domain" and "frequency–domain" synthetic seismograms should coincide.

file `98-ss.h` continued

```
# Comparison of the seismograms
  SS='98-sst.gse'   KOLOR=2   # time-domain seismograms (red)
  SS1='98-ssf.gse'  KOLOR1=1  # frequency-domain seismograms (black)
  SP1='98-ssb1.ps'  SP2='98-ssb2.ps'  SP3='98-ssb3.ps'
  sp:      '98-ss.h' /      # seismogram plotting
```

That is why the "time–domain" seismograms, plotted in red, are overlaid by the black "frequency–domain" seismograms in pictures `98-ssb1.ps`, `98-ssb2.ps` and `98-ssb3.ps` shown in Figure 9.

If we wish to make also the later arrivals visible, we can use exponential scaling with time.

file 98-ss.h continued

```
# Exponentially scaled seismograms
  NORMSP=0  SPAMP=-1  SPEXP=5
  SP1='98-sse1.ps'  SP2='98-sse2.ps'  SP3='98-sse3.ps'
  sp:      '98-ss.h' /      # seismogram plotting
```

If we set `SPEXP=5`, the amplitudes are multiplied by the factor e^{5t}, where t is the travel time (in seconds). Here we have set `NORMSP=0` to normalize the maximum amplitude at each trace to the given value of `SPAMP=-1` spacings between the receivers. The resulting plots are quite nice, but they differ considerably from classical seismogram sections, see Figure 10.

Let us emphasize that, in most cases, there is no reason to calculate synthetic seismograms in the time domain. The time–domain synthetic seismograms should be identical to the frequency–domain ones, but the time–domain option has several restrictions: (a) no attenuation, (b) no frequency–dependent Green functions and thus no coupling ray theory for weakly anisotropic models, (c) no responses of stacks of fine layers calculated by a matrix method. It is thus recommended to calculate synthetic seismograms in the frequency domain.

References

Bulant, P. & Klimeš, L. (1996): Examples of seismic models. Part 2. In: Seismic Waves in Complex 3-D Structures, Report 4, pp. 39–52, Dep. Geophys., Charles Univ., Prague.

Klimeš, L. (1995): Examples of seismic models. In: Seismic Waves in Complex 3-D Structures, Report 3, pp. 5–35, Dep. Geophys., Charles Univ., Prague.

Klimeš, L. (1998): Comparison of ray–matrix and finite–difference methods in a simple 1-D model. In: Seismic Waves in Complex 3-D Structures, Report 7, pp. 169–180, Dep. Geophys., Charles Univ., Prague.