# PAPER M

# COMPUTATIONAL SPEEDUP OF THE STRING ALGORITHM

**Luis L. Canales**
*Seismic Tomography Project*

## ABSTRACT

The string algorithm has been speeded up 4 fold. The program was analyzed with the UNIX profiler and found to spent 93% of the time in the ray tracer, which is based in the Runge-Kutta method. Inside the ray tracer, the interpolation to obtain the velocity at the required point used 77% of the time.

A fine mesh grid was incorporated in the algorithm to avoid the interpolation. With the fine grid, instead of interpolating to obtain the velocities, we just look for the grid point nearest the required location. This fine grid is normally output by the String inversion, so subsequent iterations do not require additional effort to produce it.

The grid is very fine, compared with the ray coverage and it is very important to smooth the velocity grid between iterations. Fortunately smoothing is very inexpensive as compared to ray tracing.

## INTRODUCTION

The String algorithm basically works by generating a fan of rays from a source position. The number of rays is large enough so that there is good sampling of the velocity values in the grid. A ray normally passes between two receivers, whose measured arrival times are interpolated to give an estimate of the time at the ray. The time residual is then back projected to give corrections for the velocity at the grid points touched by the ray.

It turned out after using the UNIX profiler that the program spent most of the time (93%) in the ray tracing part of the algorithm. Further analysis showed that the interpolation needed to compute the velocity at the locations needed by the algorithm used 77% of the time. This is because the rays do not necessarily travel along grid points, especially when using a coarse velocity grid.

A fine grid was introduced so that instead of interpolation we just do a simple look-up to return the velocity value of the nearest neighbor. This essentially reduces the amount of computation to one fourth of the original time. In order to use the nearest neighbor we have to make sure that the velocities are smooth, but smoothing is very inexpensive as compared with ray tracing.

## SMOOTHING

Smoothing is done in the velocity grid. A very satisfactory way to smooth the data is to use the Hanning smoother. This has the virtue that the smoothing function does not have impulsive derivatives. Impulsive derivatives of high order can seriously affect the effectivity of the Runge-Kutta method for solution of differential equations, so normally the velocity grid needs to be smooth.

The Hanning function consists of a three point weighted running average with weights ( .24, .56, .24 ), centered in the middle point. The amount of smoothing depends on how many times the filter is passed trough the data. Normally two to four times in the z-direction and four to sixteen times in the x-direction.

Even though we are using a fine grid for the velocities, the ray coverage is still very coarse. It turns out that the amount of smoothing needed to account for the ray coverage coarseness is more than enough to smooth the velocity grid, so we do not lose any efficiency on account of the smoothing.

## RESULTS

We tried the modification on some synthetic data. Figure 1 shows the velocity perturbation produced by the iteration. The coarseness of the ray coverage can be seen. Normally before plotting and before a new iteration the velocity field have to be smoothed. Figure 2 shows the velocity field after smoothing. There was smoothing before and after computing the perturbation.

To validate the change on the algorithm, we run an iteration with the old algorithm, it is shown in figure 3, after smoothing. The smoothing was equivalent to that of figure 2. As you can see there is hardly any difference between the two velocity fields, showing that indeed we did not lose anything in the speedup.

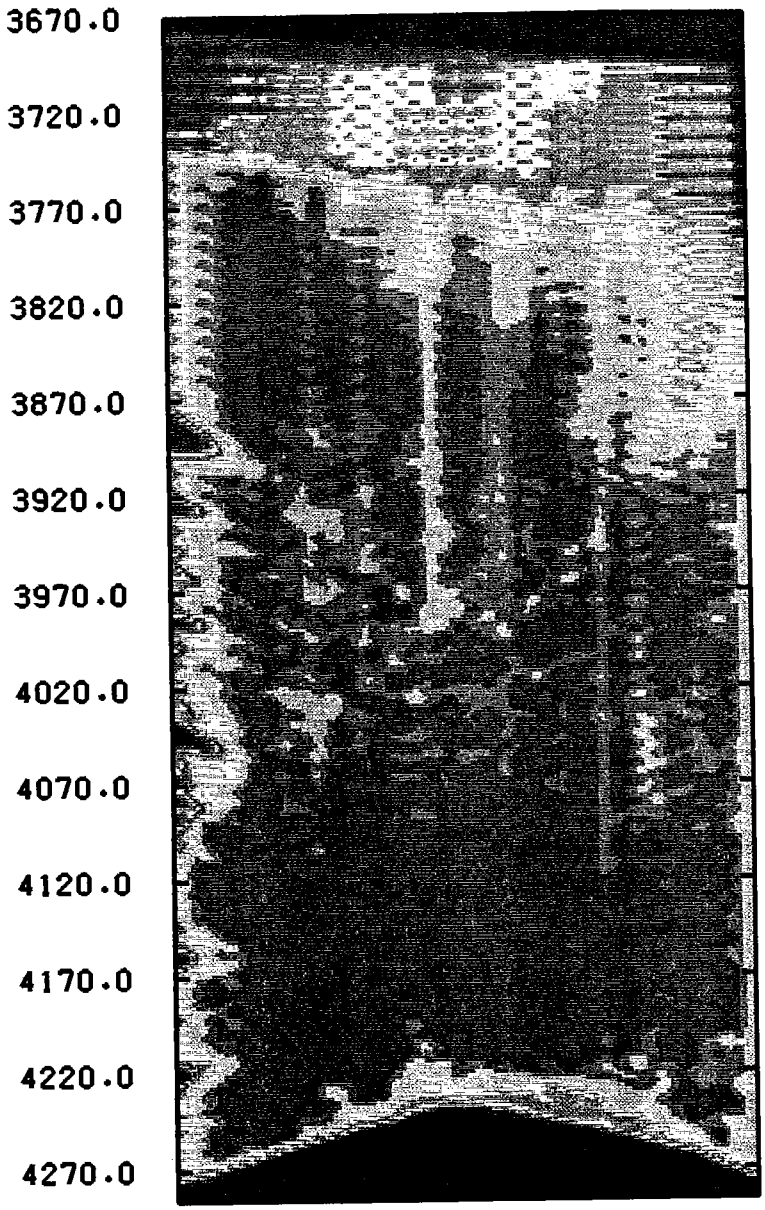


**Figure 1**
**String iteration with fine grid, no interpolation without smoothing**

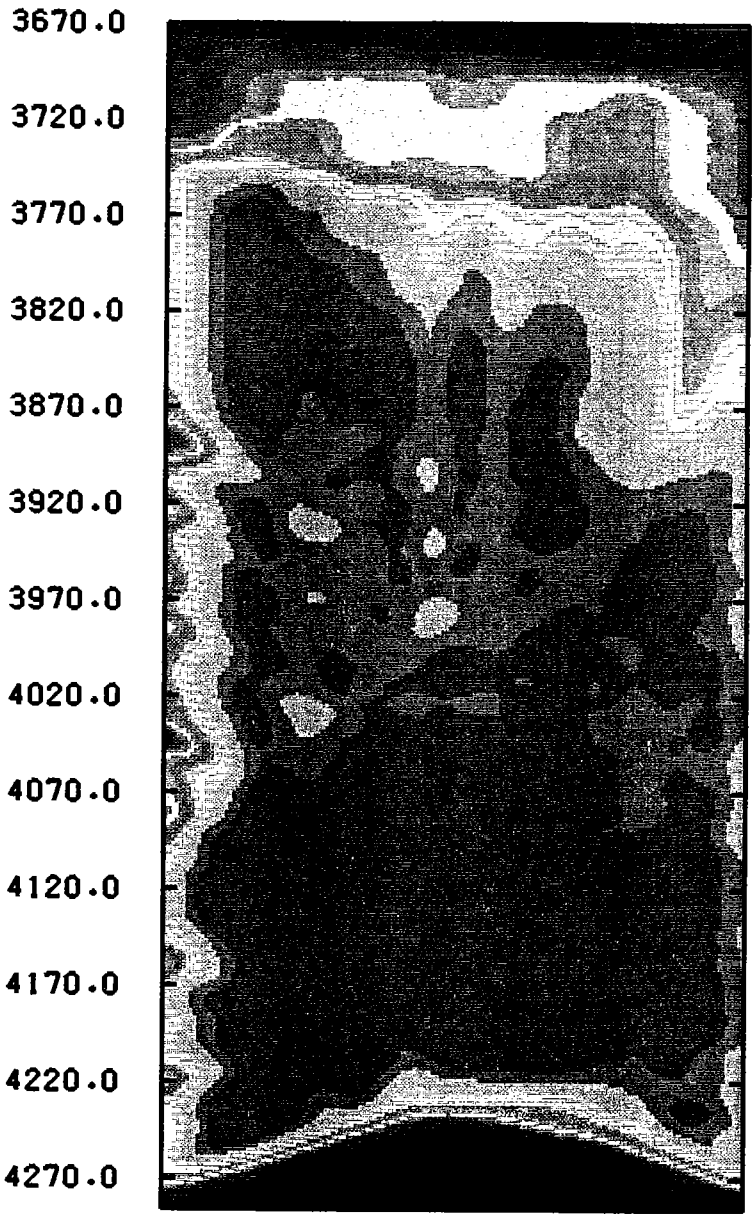


**Figure 2**
**String iteration with fine grid and no interpolation after smoothing**

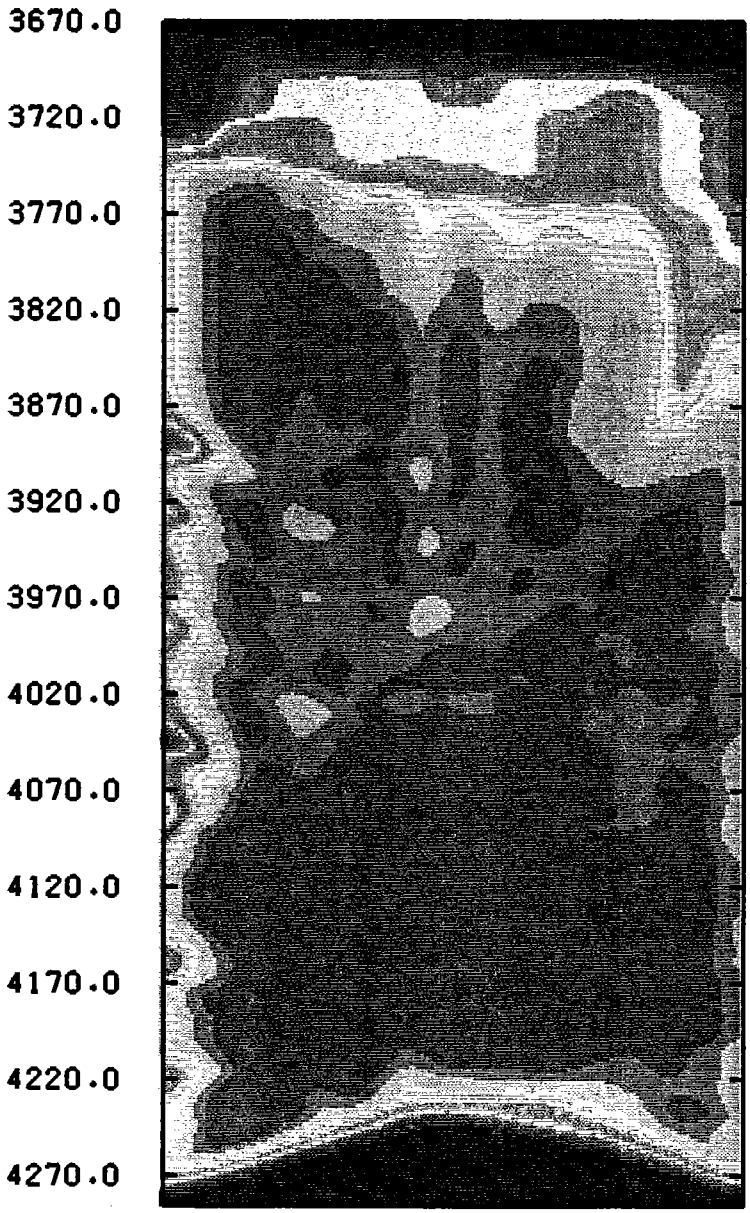


**Figure 3**
**String iteration with coarse grid and interpolation after smoothing**

## CONCLUSIONS

We have shown that by using a fine velocity grid we can eliminate the grid interpolation needed by the ray tracing part of the string algorithm. Instead of interpolating we just look for the nearest neighbor, thus speeding the algorithm four times. A grid giving about 10 points per wavelength has been very satisfactory.

To avoid inaccuracies we need to smooth the velocity field between iterations, but it turns out that the smoothing needed by the coarseness of the ray coverage is enough, so we do not really lose any computational efficiency in doing this step.

## ACKNOWLEDGEMENTS

The author gratefully acknowledges the Gas Research Institute for its support of the Stanford Tomography Project.