Exploiting Trace Based Parallelism in Geophysical Calculations

Peilin Jia

EPCC–SS94–12

Exploiting Trace Based Parallelism in Geophysical Calculations

Peilin Jia

EPCC–SS94–12

September 1994

Abstract

Almost all geophysical data is recorded as traces; signals recorded in the time domain from individual transponders such as geophones, hydrophones, electrodes and search coils. Subsequent processing to produce a data volume ready for interpretation is trace based and, due to the large numbers of traces which are usually recorded, has a high potential for parallelisation by data decomposition using the trace as the basic unit of data.

Modular applications builders such as AVS allow scientists to build an application by interactively linking program units in a graphical user interface. The Euphrates software system facilitates the automatic parallelisation by data decomposition of applications described as a map of processing modules such as might be constructed in AVS. Parallelisation using data decomposition is well understood, the major innovation of Euphrates is to support the processing of arbitrarily large data sets on DM-MIMD parallel computers with limited memory.

This project investigates the general characteristics of trace based problems and the best way to parallelise them with Euphrates. The new parallel software engineering problems which arise when tackling arbitrarily large images are explored and discussed. Inversion of the wave transform, a trace based calculation useful in processing of electromagnetic prospecting data, has been incorporated into the Euphrates framework in order to provide a concrete example. This is achieved through the addition of new processing modules and the extension of the functionality of the Euphrates framework.

1 Introduction

Geophysicists aim to estimate properties of the interior of the earth by making measurements at the surface of the earth. Commonly the response of the Earth to some known signal is measured, processed and then interpreted to yield an estimate of the subsurface structure. Applications include estimation of the properties of the Earth's core, oil and mineral exploration, and near surface investigations of waste disposal sites.

Most geophysical data is recorded as traces; signals recorded in the time domain from individual transponders. Seismic exploration records sound waves bouncing back from layers in the Earth using geophones or hydrophones while electromagnetic methods employ electrodes and search coils. The processing of this data involves trace based calculations; operations that can be performed either on individual time series or a group of related time series. Traces form the natural basic data unit in either case.

In order to ensure the highest possible signal to noise ratio, it is typical to record many traces, which results in an enormous volume of data. In seismic surveying "some 30 000 [traces] for each square kilometer of a 3D survey" [Petroleum, 1990] might typically be recorded. Subsequent processing is complex and time-consuming. However the trace based characteristic of most geophysical calculations means that multi-processing computers can perform a major role in reducing the time required for processing. Traces within a dataset may be divided among cooperating processors, each processing one portion of the data set, thus reducing the computing time dramatically.

The Euphrates project, a collaboration between Edinburgh Parallel Computing Centre (EPCC) and the petroleum industry, was initiated to allow non-specialists to fully exploit the advantages of parallel computing. Euphrates allows scientists to interactively prototype a processing sequence using the user friendly interface of modular application builders (MABs), such as AVS [**?**] and Explorer [Incorporated, 1991]. From within these MABs, the user can interactively prototype the processing sequence by linking up the modules provided by the Euphrates library using MAB networking tools. Once the prototype application is completed, it is translated into Euphrates Map Language (EML) a MAB independent format and then 'canned' into a stand-alone application. Once canned, the stand-alone application may then be submitted for execution over a large data set as a batch job on a DM-MIMD parallel computer.

The domain of computational problems Euphrates attempts to address has certain characteristics in terms of data set form and update algorithm. The problem usually involves a global 3-dimensional data set as the input, which is fully populated and has rectilinear shape. During the process of calculation, a series of operations are performed on all data elements. To update individual elements of the data set, the calculation either involves only the old value of the element or the values of some nearby elements. Parallelisation of such computations by regular domain decomposition is elegantly supported by utility libraries such as PUL-RD (Parallel Utility Library for Regular Domain Decomposition) [Chapple and Trewin, 1994]. Euphrates operates by generating source code which links against separate libraries which provide the required application functionality and a framework in which to parallelise the application for execution on a DM-MIMD supercomputer [Thornborrow *et al.*, 1993]. Utility libraries such as PUL-RD and PUL-GF (Parallel Utility Library for Global Files) are convenient tools in exploiting such parallelism and form the backbone of the Euphrates framework.

A major innovation in Euphrates is that it will support the processing of arbitrarily large data sets. This is particularly important when trying to process real-world data sets on DM-MIMD parallel computers which, although computationally powerful, may have limited amounts of main memory per node and no virtual memory facilities. Unlike regular domain decomposition, the processing of arbitrarily large data sets is a novelty in the parallel computing world, and is the principle new parallel software-engineering problem solved by Euphrates.

In this project, a particular trace base problem, inversion of the wave transform, is investigated as an example for solving similar problems within the Euphrates framework. The purpose of the project is not only to solve this specific geophysical problem, but also to explore the functionality required of the general support for the trace based processing of arbitrarily large images.

2 The Euphrates Project

Euphrates is one of a series of collaborative projects between EPCC and the petroleum industry. A previous project developed a number of new 3D seismic processing techniques and implemented these on a Meiko Computing Surface sited at EPCC [Wilson *et al.*, 1991]. The goal of the Euphrates project was to take these developments and present them in a form suitable for use in operating companies.

2.1 The Euphrates Philosophy

The approach taken by Euphrates was to build a system in which a scientist could interactively prototype a processing sequence, running small jobs to test that the desired effect was being achieved. The scientist would then submit a batch job, possibly to a parallel supercomputer, which would then execute on a much larger, production, data set. Afterwards the output from this production run might well be visualised back on the original workstation — see figure 1. This system had to be suitable for use by geoscientists who had no wish to become programmers, let alone parallel processing specialists. In addition it should be possible for programmers to add new modules to the system without having to become parallel processing specialists.

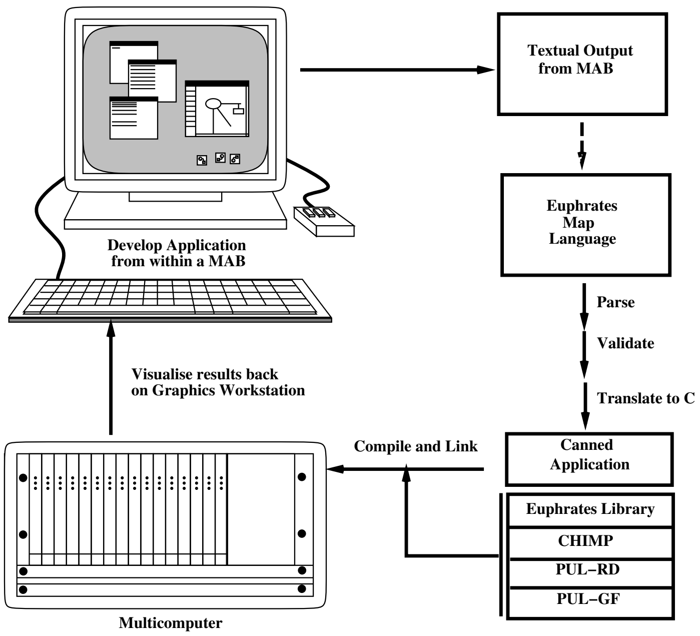


Figure 1: Linking MABs to Multicomputers in Euphrates. The Euphrates framework supports the generation of parallel code from the applications users build using either Euphrates Map Language or MAB Euphrates Library. The process of generating the applications is called the Euphrates Cycle, which describes a chain of processes including linking modules within MAB, translating the application into EML, further translating the EML to parallel C or executable code, running the code on multi-processors, and displaying the results back in MAB's.

In order to achieve this with the limited time and resources available it was decided to take advantage of existing MABs. A number of modules were prepared for such systems to allow users to interactively prototype their processing sequence. The process of extracting an application from the execution environment of a MAB is known as *canning*, and it consists of first translating the prototype application from the MAB-specific description into Euphrates Map Language (EML) and secondly translating EML into a *canned* application. Finally a batch job performing the desired function is created from the EML map description. This is achieved by generating source code which links against separate libraries which provide the required application functionality and a framework in which to parallelise this for execution on a DM-MIMD supercomputer.

The latest version of Euphrates works with AVS as a convenient graphic user interface and final visualisation tool. EML functions as an intermediate language between the map language of AVS and the desired parallel code. This allows Euphrates to be extended to work with other MABs with the minimum of effort.

2.2 Implementation

To ensure a user-friendly interface, hiding the generation of complicated parallel code, the framework of Euphrates handles several major jobs, such as parsing command lines and translating EML to subsequent parallel code, and linking different parallel libraries. The principal parallel libraries Euphrates employs are PUL-GF and PUL-RD, which support parallel file access and data decomposition respectively. In addition, Euphrates uses a library which contains the C or Fortran code for all the application functions. The use of the PUL libraries aids portability allowing Euphrates to be ported easily to networks of workstations such as Suns and RS6000s, and more closely coupled machines such as Meiko Computing surfaces, CM5 and Cray T3D.

In common with many geophysical applications, the functionality initially targeted featured processing of a regular mesh of values by finite difference operations. Previous work at EPCC indicated that the most effective method for parallelising such operations was to introduce data parallelism.

As outline above, Euphrates automatically generates a parallel application from a description in a high-level language (EML). Clearly this task is too difficult to be tackled in the general case since this would amount to the construction of an all-purpose parallelising compiler for DM-MIMD computers. Instead the approach taken was to define a class of operations which would be supported and a module developers interface. Any operation which falls into this class, and which is programmed in accordance with the module developers interface, will be automatically, and correctly, parallelised by Euphrates.

2.2.1 Characterising Operations

Operations on meshes can be discussed in terms of the *task* which must be performed at each site on the mesh and the *perspective* of the operation as a whole. Three properties of tasks can be identified which are key indicators of the efficiency which can be expected from a parallel implementation. These properties are the *spatial dependence* of each task, which describes the dependence of a task at one site upon information from other sites; the *activity* of the operation which describes the distribution of tasks across the mesh; and *precedence* which describes the order in which tasks must be executed. Operations which have local spatial dependence, global activity and no precedence relations are the easiest to parallelise and are termed *regular* since a regular geometric decomposition may be expected to provide an efficient parallelisation.

The perspective of an operation may be *global* or *local*. Global operations, such as a global sum or maximum, require all tasks to be completed before another operation can begin. This is not the case for operations with a local perspective and so a sequence of these operations may be applied to a large data set on a tile by tile basis allowing overlap of file I/O and computation and greatly reducing memory requirements.

2.2.2 Designing for Performance

Euphrates chose to tackle regular operations only, since these were the easiest to parallelise, and use could be made of the PUL-RD utility [Chapple and Trewin, 1994] which supports regular decomposition. Only fully occupied 3D rectangular meshes are currently supported, although voxels may have any aspect ratio.

Regular operations may be effectively parallelised by the Single Programme Multiple Data (SPMD) approach where each processor executes the same program on a different part of the data set. Euphrates follows this approach and, in order to minimise context switching, combines the entire functionality of the application into one application process per processor.[1]

In order to maximise translation speed and minimise the volume of generated code the application modules and the supporting parallel utilities are accessed by making calls to pre-compiled libraries.

2.2.3 Providing Arbitrary Connectivity

The order in which calls to application modules are made is determined during the translation from EML to C source code by analysing the topology of the input map. Arbitrary connectivity can be achieved by ensuring that the modules are sorted into an order where each module only uses data output from a module which occurs earlier in the list. This process also checks for cyclicity; maps which contain feedback loops cannot be supported since there is no user interaction and such an application, once set running, would run forever. The introduction of some flow control mechanisms, perhaps following the approach taken by the Khoros system, is an interesting alternative.

2.2.4 Processing Arbitrarily Large Images

In addition to the needs for user-friendliness it must be possible to process very large[2] seismic images within Euphrates. The need to process large images on transputer and i860 based systems with limited amounts of memory available at each node, and no virtual memory system, led to a decision not to support operations with a global perspective. Such operations would have required Euphrates to implement an equivalent to a global memory paging system for DM-MIMD systems. This was considered inappropriate since current and future DM-MIMD systems seem likely to based around commodity processors and will either have virtual memory systems on each node (as in the case of a network of RS6000s) or may have a virtual memory system (as in the case of the Meiko CS-2).

Instead Euphrates pre-calculates the memory requirements of an application in terms of the maximum number of images which will exist at any one time during one run of the application. This analysis is performed during the translation of EML into DM-MIMD source code. Given this information, and an upper limit on the amount of memory available at each node, it is possible to process arbitrarily large images using regular operations with local perspective. This may be achieved by dividing the input images into regions and then processing each region in turn. This decomposition is termed the *primary decomposition* in order to distinguish it from the *secondary decomposition* in which a primary region is distributed over a number of processors. Both the primary and secondary decomposition are performed at run-time, allowing the same DM-MIMD executable to run on systems with varying numbers of processors and with varying amounts of memory per processor, without requiring retranslation or recompilation.

2.2.5 The Module Developers Interface

The module developers interface is remarkably simple. Developers are required to provide a C function call, and a plain text file which describes the arguments to this call. In this plain text file arguments are associated with ports and are described as either input or output, parameter or data. Parameters may be of any type, but arguments corresponding to data ports may only be of one type; a C data structure which describes an array, and an arbitrary region of this array. Modules must not alter input data arrays, and must write data to fill the indicated region of output data arrays.

In order to allow overlap of file I/O and computation, modules are categorised to distinguish those which input and output data from disk. Module developers must provide three separate functions for I/O

[1] Each processor may run processes associated with the PUL and CHIMP utilities, as well as the Euphrates application process.
[2] A typical 3D seismic survey of around 10,000 Km lines will occupy around 920 Gbytes.

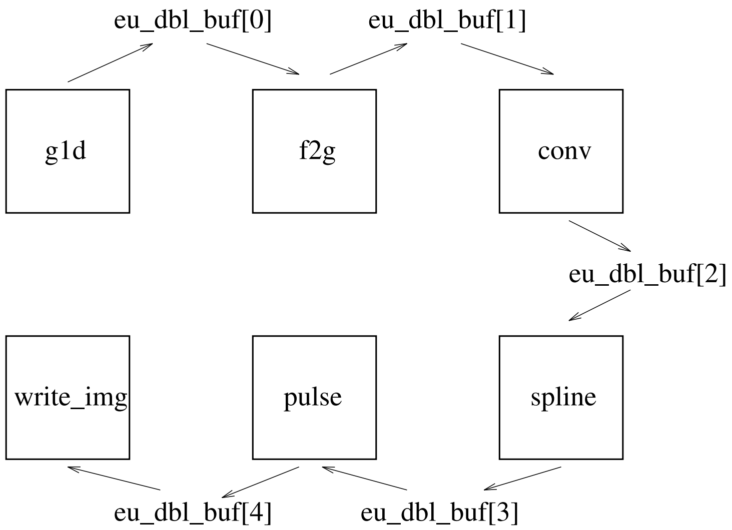


Figure 2: The Buffer Maintaining Scheme of Euphrates Framework

modules, which will be referred to as init, start, and end. In outline the init function sets up some data structures and opens the file and then then start and end functions each read in one region of the data set for processing at a time.

In between the modules, the Euphrates framework maintains a series of buffers that contain the data structure relevant to the image as well as the data within the image. Between every two compute modules, the framework creates an intermediate buffer to reflect the update of the data structures by the compute module. Buffers also get re-used to minimise the memory consumption. Figure 2 demonstrates the buffer maintaining scheme. Normally, the input module and the output module are each associated with one buffer, and a compute module is associated with two. Further details are available in the Euphrates user guide [**?**].

3 The Wave Transform

Inversion of the wave transform, which proves to be a critical step in processing data from multi-channel transient electromagnetic (MTEM) prospecting, is a typical trace based problem. This section gives the relevant geophysical background for the purpose of and method used in inverting the wave transform, and explains the basic processing operations that are used at each step in the inversion.

3.1 Electromagnetic Prospecting and The Wave Transform

Reflection seismology is the dominant exploration methodology in hydrocarbon prospecting. Seismic wave propagation is sensitive to velocity and density contrasts but variations in pore fluid content do not enter the governing equations directly, and in general porosity variations can only sometimes be inferred from variations in seismic velocity.

Multi-channel transient electromagnetic (MTEM) prospecting is directly sensitive to contrasts in bulk resistivity due to, for example, a change in pore fluid from brine to oil. Recent hardware advances allow acquisition of the closely spaced MTEM data required to image hydrocarbon reservoirs for the first time. However it is currently unclear how best to process such data once acquired.

In regions of complex geology it is well known that the expense of inverse modelling — the iterative refinement of a model through forward modelling until a good fit with the experimentally acquired data is

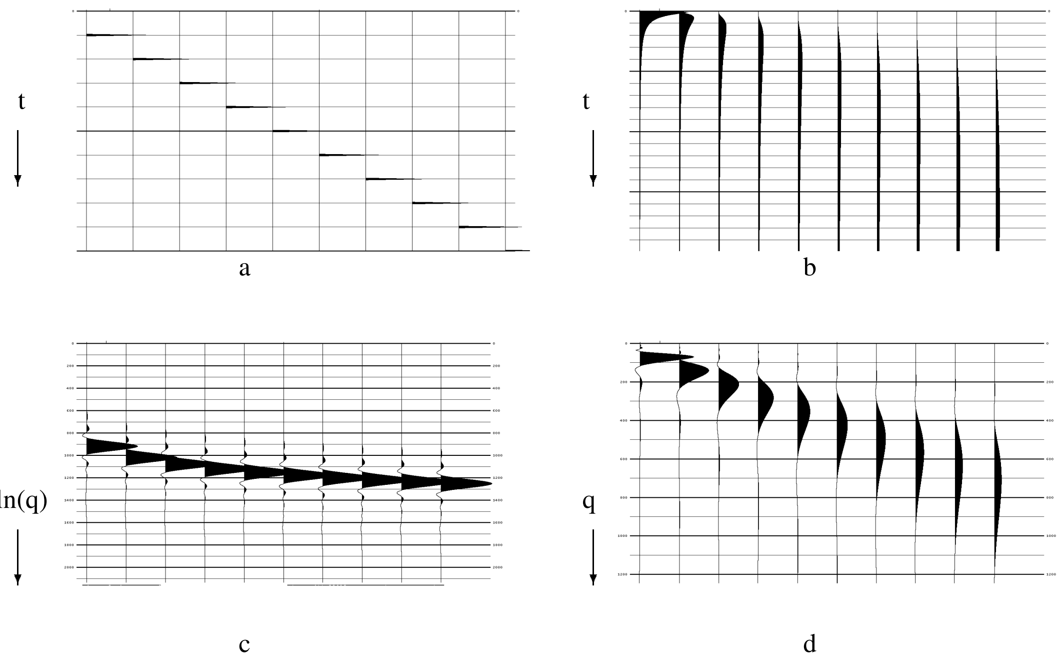


Figure 3: Wave Transform Inversion of the 1D Green's Diffusion Function

achieved — is enormous. There is clearly a need to develop a satisfactory methodology for processing and interpretation of MTEM surveys over complex geology similar to the well-developed and well-understood methodology for processing reflection seismic data.

The wave transform relates the response of a diffusive medium, and the response of an equivalent wave propagation medium, provided the sources, geometry and boundary conditions in both domains match. The wave transform may be inverted to calculate an equivalent wavefield from diffusive transients and then seismic data processing and interpretation techniques may be applied to the equivalent wavefield to recover the equivalent velocity $c(r)$. For EM data a resistivity profile $\sigma(r)$ may then be calculated using the relationship $\mu\sigma(r) = c(r)^{-2}$ [Lee and Xie, 1993].

Inversion of the wave transform is well know to be an inherently ill-posed, and consequently computationally expensive, problem. Inversion schemes introduce differing constraints on possible solutions in order to improve resolution and tractability.

3.2 Wave Transform Inversion by Deconvolution

The wave transform relates $F(r,t)$, a diffusive field and $U(r,q)$, the equivalent wavefield through an integral transform

$$F(r,t) = \frac{1}{2\sqrt{\pi t^3}} \int_0^\infty q \exp(-\frac{q^2}{4t}) U(r,q) dq.$$

The action of the transform is to progressively smooth out a pulse as it travels. Figures 3a and 3b depict respectively the 1D Green's functions for propagation in wave and diffusive media; these are an impulse travelling with constant velocity, and a pulse whose profile gradually changes, dispersing and broadening with increasing time.

The 1D Green's function for diffusion may easily be calculated by applying the wave transform to the 1D Green's function for a propagating wave. Inversion of the wave transform involves the recovery of the impulse from the smooth diffusive transient. Because the action of the forward transform is to smooth out any sharp irregularities, the action of the inverse transform is to amplify any irregularities.

This causes wave-transform inversion to be an ill-posed problem; a small perturbation of the input, such as the addition of some random noise, is amplified causing a dramatic change in the output. Thus small errors in the measured data can produce large changes in the solution. In order to obtain a useful processing system some kind of constraints must be applied to regularise the inversion, so that stable, but approximate, results are obtained.

By moving to a logarithmic sampling in both time and the time-like variable q the wave transform may be re-formulated as a convolution. In effect, by moving to logarithmic time sampling, we are parameterising the inversion and the regularisation which is being imposed by this scheme so that equal resolution is achieved in logarithmic time. Deconvolution, the inversion of convolution equations, is a subject much studied in exploration seismology and one advantage of formulating the wave transform as a convolution is that it allows the inversion to be performed using standard seismic deconvolution techniques. This deconvolution method is also more efficient than some previous approaches, such as discretising the wave transform as a matrix equation and applying regularised inversion. In addition, wave transform inversion by deconvolution has been demonstrated to give superior results to other techniques using known wave/diffusion pairs, synthetic diffusive transients and field data [Gershenson, 1993].

In outline, the processing steps involved in wave transform inversion by deconvolution are as follows. Making the substitutions $q = \exp(v)$ and $t = \frac{1}{4}\exp(2u)$ in the wave transform above results in a convolution equation

$$G(u) = R(v) * W(v),$$

where

$$G(r,u) = \sqrt{\pi}\exp(u)F(r, \tfrac{1}{4}\exp(2u)),$$
$$R(r,v) = U(r, \exp(v)),$$

and

$$W(v) = 4\exp(-2v)\exp(-\exp(-2v)).$$

After pre-processing the first step is to calculate $G(u)$ from the measured transient electromagnetic data $F(t)$. This is achieved by multiplying the pre-processed data by a factor which varies like the square root of time and then resampling onto a logarithmic time scale, using a splining program. Next we convolve $G(u)$ with a suitable deconvolution operator M which is calculated as the (approximate) inverse of W, the convolutional kernel, using one of a number of standard techniques. This convolution yields $R(v)$ which is then resampled, again using spline interpolation, to give the final output.

4 Building Trace Based Applications with Euphrates

Although the initial focus of Euphrates was 3D image processing, it is well suited to trace based problems and, with a few modifications and extensions of the original Euphrates design, many trace based geophysical applications can be parallelised within Euphrates. This section explores the characteristics of trace based applications, the demands they make on the Euphrates framework, and modifications and extensions to the original Euphrates design. which are required to support them.

4.0.1 File Formats

The initial focus of Euphrates was 3D image processing, and so the VoxelView file format was selected as an appropriate file format for storing the large 3D data sets Euphrates was designed to process. This was extended slightly by storing a collection of relevant images, such as images measuring different characteristics of the same physical entity, in an $imagebank$. This is a directory structure within which each image resides in a separate sub-directory. The top level directory also contains an ASCII file name _imagebank, which stores details of the images in the bank, such as the number and type of images.

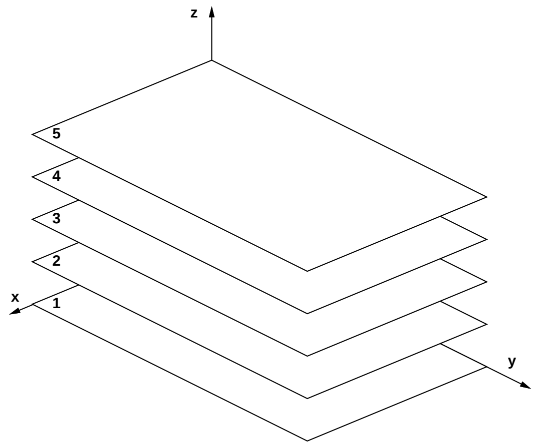


Figure 4: Data Stored as Slices in VoxelView Format

Individual images are stored in the VoxelView[3] file format. An image directory thus contains a set of data slice files, each named simply with number (1, 2, 3 ...), and an ASCII file named _dimension , which describes the details of the data volume, as shown in Figure 4. Usually, in the case of 3-D data sets, each slice contains data of the same z coordinate, with x dimension as the fastest increasing one.

In trace based problems the basic unit of data is now a trace. However the same data format can be adapted to store traces effectively with only a few modifications in the _imagebank file. This allows re-use of previously prepared software modules and compatibility with other ordinary images. The time dimension of a trace based data set can be stored as the fastest increasing dimension in the data file, thus each trace fills a contiguous region of the file. The remaining two dimensions can be used to record, for example, the shot and receiver number associated with each trace.

Information including the length of the traces, number of total traces, offset between adjacent traces, the starting time, time interval, and the sampling method along each trace has to be recorded in the _imagebank file. Addition of new fields including trace related information is also necessary in the image and imagebank data structure employed in the Euphrates framework. The Euphrates framework maintains an updated copy of these structures at each stage of data processing, which will be discussed in the later sections concerning framework modifications.

4.0.2 Memory and Data Decomposition

One of the major innovation of Euphrates is to be able to process arbitrarily large data sets. This facility requires some effective method of splitting a dataset into regions which can then be efficiently and correctly processed. In ordinary image processing by Euphrates, two stages of data decomposition are usually involved. The first stage, the primary decomposition, splits data into blocks that could be fit into the core memory of multicomputer. The secondary decomposition similarly divides data into sections to be distributed over different processors.

For ordinary 3-D rectilinear images each dimension has the same significance, and the data decomposition method is only restricted by the available memory on each processor. Many trace based operations, such as resampling or convolution, involve the input trace only. Thus splitting a trace among processors wouldn't be a wise idea. For trace based images, data decomposition also has to ensure the contiguity of the time dimension to reduce the extra amount of communication during processing. In inverting the wave transform, the framework distributes the traces generated by the input module equally over several processors. Each subgroup of traces enters the chain of modules in parallel, and undergoes a series of transforms, which leads to the final result. The alternative is to store traces as pointers to vectors, which

[3] VoxelView is a volume visualisation product of VitalImages Ltd. No association or endorsement is intended.

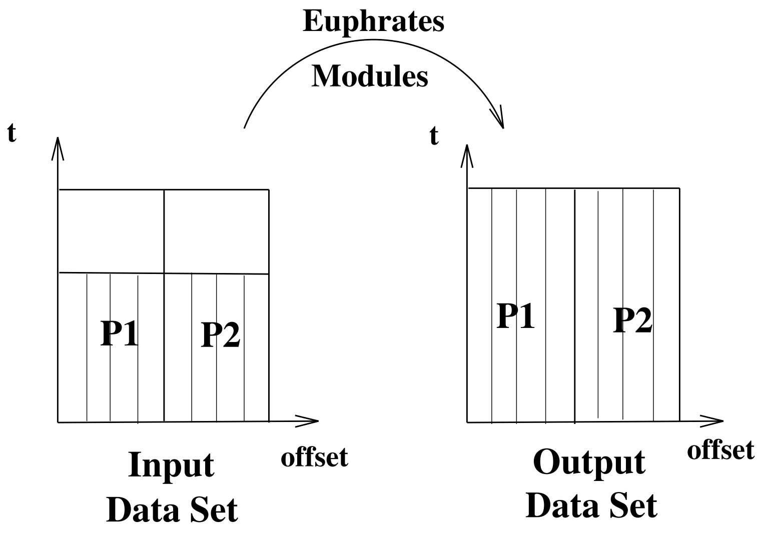


Figure 5: Traces in Transform

ensures the contiguity and special significance of the time dimension, and allows more freedom in the initial data decomposition. This alternative will be discussed in the section of future work on Euphrates.

In previous versions of Euphrates, images were only allowed to shrink or stay the same size as they were processed [?]. However, in trace based applications, images can grow or shrink, depending on the operations they undergo. Thus the problem of specifying the maximum expansion of traces arises during the course of primary decomposition; the necessary amount of memory has to be precalculated to match the available amount of core memory. In this project, the maximum expansion of traces is specified as a parameter of the input module. By adding the time dimension of the data generated and the index of maximum expansion, the input module creates a data set with the sum as the length of the time dimension, while the real time dimension is recorded as a field in the image structure created by the framework. As shown in Figure 5, this method circumvents the memory problem of expanding traces by allocating more memory to the starting buffer. Alternatively, expansion and shrinkage of traces through each module could be specified in the module declaration, and could be collected by the compiler during run time. This would be a more flexible method, and will be discussed in the section about future work.

4.1 Framework and Existing Module Modification

4.1.1 Data Structures

To record sufficient information for trace based images, additional fields such as length of traces, number of traces, starting time, and sampling interval have been added to the eu_img and eu_img_bnk structure. The modified structures are shown in Figure 6. All the images in the bank have the same sampling method and interval and image type (trace based images or non-trace, ordinary images), so these three fields are added to the bank structure. The fields recording the starting time, length of traces, and the interval and velocity constant used in generating the traces may vary according to the image, and are therefore stored in the eu_img structure.

4.1.2 Data Decomposition

During data decomposition, individual traces have to be kept intact, which results in subgroup of traces residing on different processors. The decomp.c file in the Euphrates S library has been modified to ensure that the default decomposition is done on either y or z direction, depending on the dimensions, and keep the fastest increasing direction, x, contiguous in memory.

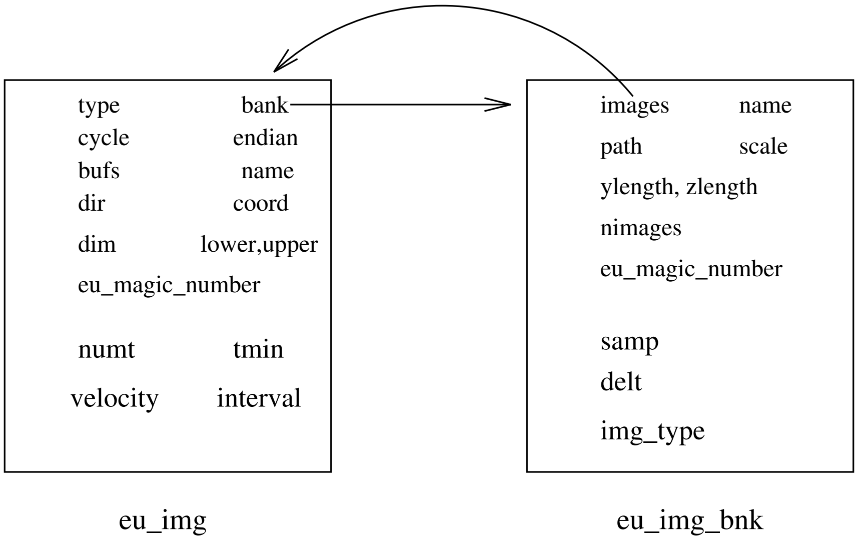


Figure 6: The New eu_img and eu_img_bnk Structure

4.1.3 I/O Modules

Certain functions related to the I/O modules have to be modified to reflect the correct trace based structure of images and image bank. In the initialisation stage of the input and output module, eu_setup_img_bnk is called to read the _imagebank file and construct the current image bank structure from the file. In the start stage of the input module, eu_update_img_bnk_P is called to update the current _imagebank file. Due to the modification in trace based data structures, these I/O related functions also have to change correspondingly to be able to write or read the correct entries in the file.

4.2 Implementation of Wave Transform Inversion

Inverting the wave transform needs four basic type of operations: data input, scaling, convolution, and resampling. These are linked as shown schematically in Figure 7. The process starts with a data input module, which is temporarily replaced by a module generating synthetic data from 1-D Green's diffusion function, called **g1d**. The diffusion field F generated by **g1d** records a number of time series with specified spatial offset. The data then flows into a rescaling module to obtain time series G, as a function of variable v, where $v = \frac{1}{2}\ln(4t)$. Deconvolution of the output data is achieved by convolving the time series with a deconvolution filter, generating output R. Further resampling R onto exponential v transforms the series into U, a wave field of variable q, where $q = \exp(2u)$. The output U generated is a reasonable approximation of the equivalent wave field of the original 1-D Green's diffusion field. By comparing the peaks of the traces in U with the analytical equivalent wave pulses, the accuracy of the method of inverting the wave transform by deconvolution can be verified.

4.3 EML Description of the Application

When using a modular application builder such as AVS, convenient networking tools ease the prototyping of applications by providing a visualisation mechanism; the application, however, can also be described directly using Euphrates Map Language. Figure 8 shows the module linking in constructing an application for inverting the wave transform.

Modules are classified according to their functions as of source image, source, compute, and sink, which correspond to functions of inputing data, inputing parameters, applying computing operations, and outputing data. As shown in Figure 8, **g1d** is an input module, which takes various parameters and

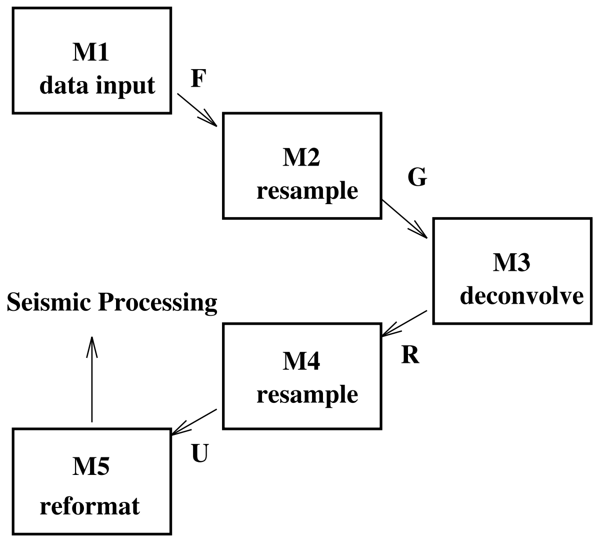


Figure 7: Module Linking in the Inversion of the Wave Transform

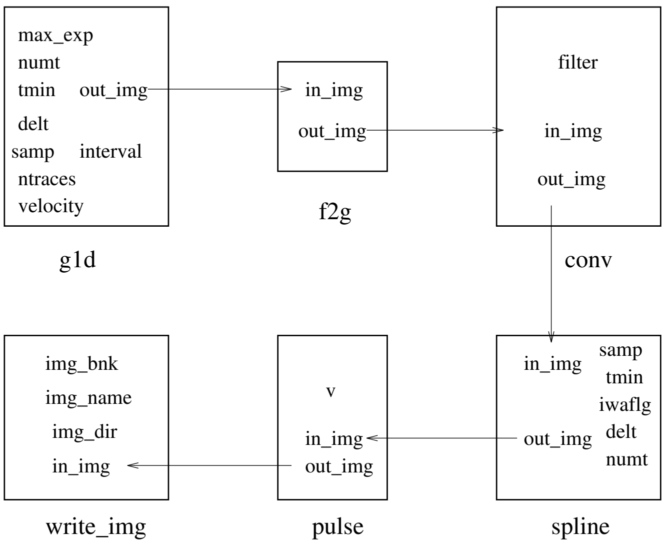


Figure 8: EML Description of the Modules Used to Invert the Wave Transform

outputs a volume of data. The output port is then linked to the input port of the **f2g** module, the output of which is subsequently linked to the input of the **conv** module, and so on. Thus, the simple textual description of the application in EML functions as the equivalent of the visual layout of modules in AVS.

4.3.1 Module Definitions

g1d is an input module similar to read_img_char, a source_image module that consists of three separate functions: init, start, and end. This common structure of source_image and sink modules allows read and write to be done in a non-blocking manner, although at present stage non-blocking read and write are not implemented. The init of **g1d** is much simpler than that of read_img_char in terms of its functionality. It merely sets up the image bank structure and the structure of the first image to be generated. Whereas in read_img_char, the directory path, file name, etc. have to be verified in addition, before returning a file pointer to be read. The start procedure of **g1d** actually generates the diffusion data using 1-D Green's diffusion function and store the information about the image type (EU_TRACE), method of sampling(EU_LOG_SAMP or EU_CONST_SAMP), and sampling interval into the image bank structure. Additional information about the starting time, velocity, and interval between traces are stored in the new image structure. The newly generated data of the diffusion field is stored in the out_buf_rgn_ptr associated with the image.

f2g functions as a rescaling module, multiplying the output traces $F(t)$ from **g1d** by a factor of $2\sqrt{\pi t}$. Like most compute modules, **f2g** consists of two parts: a loop to compute the output traces from the corresponding input traces and a section to update the image and image bank structure associated with the output buffer. It takes no parameters.

conv takes the name of a file which contains the deconvolution filter as a parameter, and convolve that filter with the input traces $G(v)$ to produce the output traces $R(v)$. This is the only module in the inversion of wave transform that outputs traces longer than the input traces. The length expands by the length of the filter minus one. Luckily, the input module took the maximum expansion of traces into account, so no memory problem would arise, although the method of updating data remains the same. Future design should allow the size of the data buffer to vary according to the requirements of different modules. Traces should be able to expand or shrink without consuming extra "padded" buffer space. This scheme will be explained in the section of further work on Euphrates.

spline takes as parameters the length, starting time, sampling interval, and sampling method of output traces, and a weighted average flag for interpolation. The module interpolates the value of function $R(v)$ at the new sampling points specified by those parameters. The sampling method of new traces could be either even or logarithmic sampling in time. In addition to the usual two parts of the code, **spline** module also has a sub-routine **get_x** to reconstruct the array of time points according to the specified sampling method, interval, and the starting time. In the inversion of wave transform, the input $R(v)$ needs to be resampled into $U(q)$, where $q = \exp(v)$. So the **samp** parameter to the module should be logarithmic (EU_LOG_SAMP). The **get_x** sub-routine can also be extended to take more than two sampling methods, to allow more flexibility in a wider range of applications.

pulse turns the input traces into spikes according to the parameter v, which specifies the requirements for the peak value to be picked along each trace. Options are positive maximum, negative maximum, and absolute maximum. The output traces from this module thus consist of integers 0 and 1, with 1 indicating the peak position and 0 otherwise.

4.4 Module Binding in AVS

To incorporate Euphrates modules into AVS, AVS wrappers of these functions needs to be generated by AVS module binding tools. The resulting AVS modules prototype a series of sequential codes, which could be run on a small set of data for testing. Although the sequential code bears little resemblance to the ultimate parallel version, the visualisation function of AVS could be a great benefit in testing modules.

#Proc	g1d	f2g	conv	spline	I/O Write	Total Compute	Speed Up
1	9.5	6	74.5	79	30.5	170	1
2	6	3.5	44.5	50.5	26	106	1.6
4	7	4.5	49.5	43	30.5	106.5	1.6

Table 1: Computing Time Comparison at Each Stage of Transform

Attempts have been made in this project to extend AVS Euphrates Library, and they should be completed in the future work of Euphrates project.

5 Performance Results

The application on a cluster of Sun4 workstations shows the result in Table1. By moving the application from one processor to two, each stage of computation shows some speed up of around 35%. The I/O write time, however, stays the same. The total computation time, from entering the data input module **g1d** to exiting the **spline** module, doesn't have a speed up of 50%. Because the synchronisation between two processors usually consumes extra amount of time. This phenomenon is even more obvious when the application is tested on four parallel processors. For some stages of the computation, some of the processors could finish the job in nearly 25% of the time as one processor would. But the slowest processor causes the bottleneck during synchronisation, and so no further speed up is achieved.

6 Further Work

6.1 Run-time Detection of Data Expansion

Unlike ordinary 3-D images processed by Euphrates, which can only shrink or stay the same size, trace based images can grow or shrink depending on the different operations on the data. So far, the expansion of traces during data processing is handled by adding a new parameter max_exp to the initial data input module. This method allocates enough space for the maximum possible buffer space from the starting module. Although this method temporarily solve the memory requirement problem, it is not flexible and efficient in the long run.

Either a compile time or a run time solution would be possible to handle the situation. The compile time solution involves some mechanism in module instance declaration, specifying the extent to which the traces will grow or shrink when passing through the module. The framework then generates code so that at run time, the maximum size to which the traces will grow is used as the buffer size.

A run time solution to detect the expansion and shrinkage of each modules would be more complete. Each module has an information function, which processes the metadata – intermediate data structures associated with the module – and calculates the expansion of a trace as it passes through the module. The information functions are called in the same order that the compute functions are later called, and they calculate the maximum trace length and allocate the buffers for the subsequent compute modules. The compute functions are then called to process the data. Figure 9 shows the calling sequence of the information modules and their corresponding compute modules. This method allows more flexibility to the framework, and also helps handling metadata, including those indicating the substitution of variables during trace processing.

6.2 Mechanism for Specifying Data Decomposition

Currently, data decomposition is handled by the eu_decomp function. The available memory each processor has is a major consideration in primary decomposition, and the shape of stencil in secondary

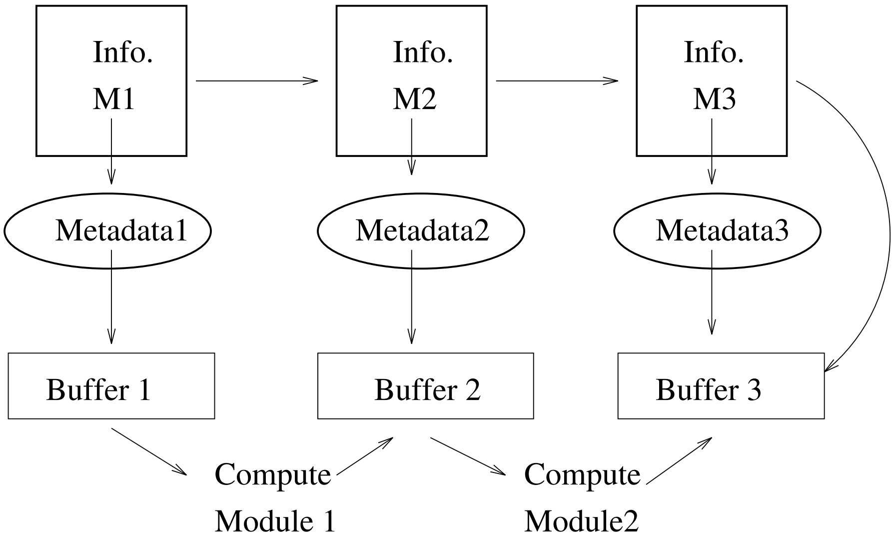


Figure 9: The Calling Sequence of Information Modules

decomposition. Trace based calculations, however, impose specific requirements for data decomposition; the time dimension of the data set should be kept contiguous. Other requirements might arise in building different types of geophysical applications. Thus, a method to specify the direction of data decomposition would provide more flexibility and robustness to the Euphrates framework.

6.3 Vector Data Type

The storage of trace based images as normal 3-D images has the advantage of preserving the old data format and inheriting the old functions in reading data files and reconstructing the image structures. This format could be awkward when required to handle time series sampled in 3 spatial dimensions. In its geophysical meaning, the time dimension is intrinsically different from spatial dimensions, and a clearer, more elegant vector data type would suit the storage of time series better, as shown in Figure10.

Each time series could be stored as a vector, and the principal array which is decomposed would then consists of pointers to vectors. Although the whole vector needs to be swapped instead of a mere pointer during boundary swapping at early stage of data processing, PUL-RD can easily handle this situation. Mechanisms, such as stencils, which facilitates the communication between neighbouring data points, can then be utilised to process adjacent traces. This scheme ensures the uniform rectilinear shape of the data set, without the constraint by the time dimension.

6.4 Need for PUL-PF

Euphrates is designed to process huge volumes of data on parallel machines. Currently PUL-GF is used to support file access, allowing many processes to write to a single image file. However this introduces a bottleneck; as the power of the computing platform increases the time required for computation will decrease but the time spent reading and writing of data files will remain the same. PUL-PF, which implements parallel access of files on multiple servers, would remove this bottleneck, allowing I/O performance to grow with the size of the parallel platform.

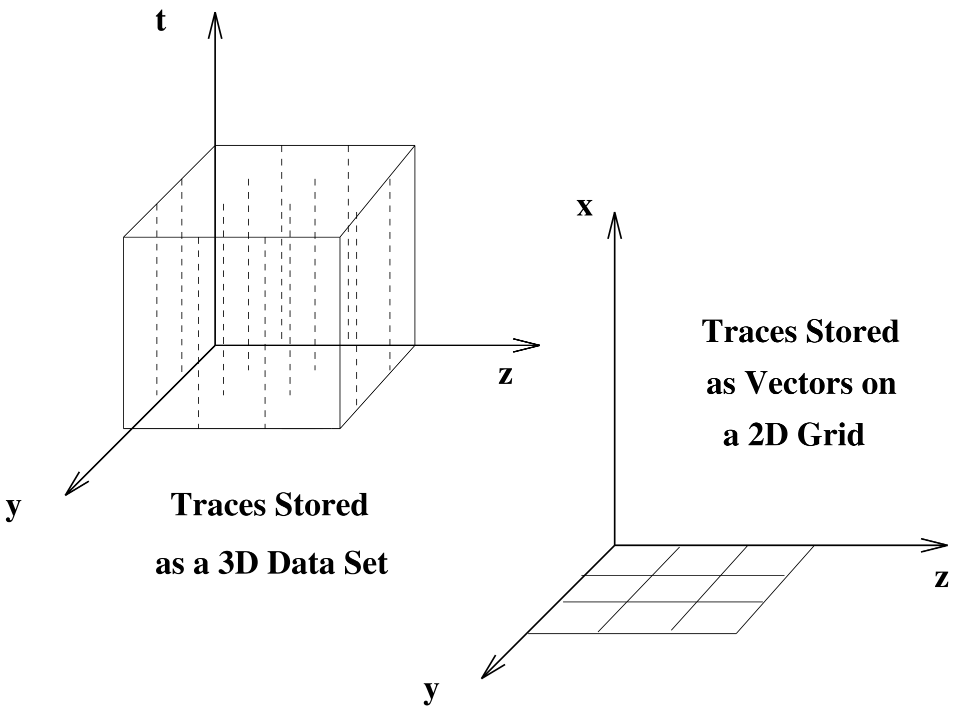


Figure 10: The Storage of Traces as Vectors

7 Conclusions

Parallelisation using data decomposition is well understood, however most applications to geophysical calculations suffer from two weaknesses that prevent their use in a real world situation. Firstly geoscientists are seldom parallel programming specialists and consequently require a user-friendly interface, secondly geophysical data sets, and in particular seismic data sets, are often very large — too large to be processed by programs which assume all the data will fit into memory at one time.

The Euphrates software system sidesteps the user interface problem by allowing scientists to develop their applications within a familiar, user-friendly environment, such as the AVS modular application builder. A description of the application as a map of interconnected modules, each of which performs a single processing operation on the data as it flows through the map, is then output as an ASCII file. From this description Euphrates will then automatically produce source code for a parallelised version of the application.

The second problem, the processing of arbitrarily large data sets on DM-MIMD parallel computers with limited memory, is more complex and is the primary focus of this report. Processing operations where the computation of each output trace is independent, and requires only the data from the input trace, can be easily parallelised. In addition Euphrates will support the application to arbitrarily large data sets of computations where the calculation of an output trace requires data from a number of neighbouring traces. In this respect Euphrates is highly innovative.

Trace based problems provide a good test of the robustness of Euphrates. Although inverting the wave transform is only part of the geophysical problem concerning the estimate of oil reservoir structures, it brings insight into the feasibility of solving general trace based problems with Euphrates. Thus this project starts a series of extension towards a more flexible and dynamic Euphrates system.

This project has demonstrated that the solution of a wide range of geophysical problems can be incorporated into Euphrates, by adding software modules for basic operations, such as summing, sorting, resampling, and convolving. An obvious continuation of the project would be refinement of the test application, inversion of the wave transform, by applying more powerful inversion methods. These would be more computationally intensive, hence file I/O would become less of a bottleneck and we would expect speed up to be observed on greater numbers of processors.

Acknowledgements

Hearty thanks to my supervisors Ian Flockhart and Andrew Wilson for their instructions in this project, suggestions and patient help in this report, and encouragement throughout the summer programme.

A An Informal Derivation of the Wave Transform

The wave transform is derived by applying the Laplace transform to the diffusion and wave equations and then by warping the real axis in one of the Laplace transform domains until the two transformed equations match. A relationship in the Laplace transform domain can then be made between the solutions of the two equations. Taking the inverse Laplace transform establishes the relationship in the time domain which we call the wave transform.

Begin with the vector diffusion equation

$$\nabla^2 F(r,t) - \mu\sigma(r)\frac{\partial F(r,t)}{\partial t} = S(r,t), \tag{1}$$

and the vector wave equation

$$\nabla^2 U(r,q) - \frac{1}{c(r)^2}\frac{\partial^2 U(r,q)}{\partial q^2} = T(r,q). \tag{2}$$

In these equations $F(r,t)$ is a diffusive field, and $S(r,t)$ is a source term representing, in this case, either a current or magnetic dipole source. Similarly $U(r,q)$ is a wavefield with $T(r,q)$ as source term. The independent variable q in the vector wave equation (2) equation is a time-like variable although in fact it has dimension $\sqrt{t}$.

The next stage in the derivation is to take the Laplace transform of each equation so that time (and time-like) derivatives become powers of the transform parameters. Here the Laplace transform parameters are s and p for the diffusion and wave equations respectively.

$$\begin{aligned} \nabla^2 \hat{F}(r,s) - s\mu\sigma(r)\hat{F}(r,s) &= \hat{S}(r,s), \quad (3)\\ \nabla^2 \hat{U}(r,p) - \frac{p^2}{c(r)^2}\hat{U}(r,p) &= \hat{T}(r,p). \quad (4) \end{aligned}$$

By setting $p = \sqrt{s}$, which is equivalent to warping the real axis in the Laplace transform domain, and requiring that $(c(r))^{-2} = \mu\sigma(r)$, where $c(r)$ is the velocity term in the wave equation, the two equations become the same. At this stage in a more rigourous proof, conditions would also be imposed requiring that source terms, model geometry and boundary conditions in the diffusive and wave domains correspond, in addition to the correspondence of the velocity term $c(r)$ which has already been specified. These conditions would then allow application of a uniqueness result concerning solutions of partial differential equations with the conclusion that

$$\hat{F}(r,s) = \hat{U}(r,\sqrt{s}). \tag{5}$$

By the definition of the Laplace transform this implies that

$$\hat{F}(r,s) = \int_0^\infty U(r,q) \exp(-\sqrt{s}q) dq. \tag{6}$$

Using the result that for real $q > 0$

$$\mathcal{L}\{\frac{q}{2\sqrt{\pi t^3}} \exp(-\frac{q^2}{4t})\} = \exp(-\sqrt{s}q) \tag{7}$$

[Erdélyi, 1954, equation (1), page 245] we take the inverse Laplace transform, from transform parameter s to time t, of equation (6) and obtain the result

$$F(r,t) = \frac{1}{2\sqrt{\pi t^3}} \int_0^\infty q \exp(-\frac{q^2}{4t}) U(r,q) dq, \tag{8}$$

which, following [Oliver, 1994], we call the wave transform.

It is perhaps unclear how to assign credit for the first derivation of the wave transform. Certainly it appears that more than one group may have independently derived equivalent results.

[Bragg and Dettman, 1968b; Bragg and Dettman, 1968a] present an early statement of the transform in a generalised form in a short communication to the American Mathematical Society. Almost concurrently [Filippi and Frisch, 1969a; Filippi and Frisch, 1969b] present their results on the other side of the Atlantic. Soon after these short, abstract, mathematical notes an equivalent result appears, albeit in the frequency domain and in a slightly less obvious form, with application to inversion of electromagnetic induction data [Weidelt, 1972, equation 5.1].

The running then seems to have been taken up by the mathematicians of the then communist block [Lavrent'ev *et al.*, 1980; Filatov, 1984]. The wave transform seems to disappear from the western literature until taken up in a series of papers by Ki Ha Lee and co-authors. These demonstrated its usefulness both in forward modelling [Lee *et al.*, 1989] and inversion [Lee, 1988; Lee and Xie, 1993]. This work should not be confused with the 'electromagnetic migration' work of Seunghee [Lee *et al.*, 1987] and [Zhdanov and Frenkel, 1988]. Work continues on wave transform inversion in the frequency [Levy *et al.*, 1988; Gibert and Virieux, 1991; Gibert *et al.*, 1994] and time [Gershenson, 1993] domains.

The wave transform is now becoming more widely known and publicised [De Hoop, 1992] and is attracting interest in diverse fields which deal with diffusion problems, including non-destructive testing of metal components [Zorgati *et al.*, 1991; Zorgati *et al.*, 1992], transmission tomography of thermal waves [Mandelis, 1991], the investigation of hydrocarbon reservoirs through pressure pulse transients [Oliver, 1994] and detection of water-borne metallic hazards [Gershenson, 1993].

References

[Bragg and Dettman, 1968a] L. R. Bragg and J. W. Dettman. Related partial differential equations and their applications. *SIAM journal on Applied Mathematics*, 16(3):459–467, May 1968.

[Bragg and Dettman, 1968b] L. R. Bragg and J. W. Dettman. Related problems in partial differential equations. *Bull. Amer. Math. Soc.*, 74:375–378, 1968.

[Chapple and Trewin, 1994] S. R. Chapple and S. M. Trewin. Pul-rd prototype user guide. *EPCC-KTP-PUL-RD-PROT-UG 1.6*, 1994.

[De Hoop, 1992] A. T. De Hoop. Transient electromagnetic and seismic prospecting – an analysis of their similarities. In *54th Meeting, EAEG*, pages 422–423, June 1992. Abstract number GO12.

[Erdélyi, 1954] A. Erdélyi. *Tables of Integral Transforms*. McGraw-Hill, 1954. Vols No I and II.

[Filatov, 1984] V. V. Filatov. Construction of focusing transformations of transient electromagnetic fields. *Geol. i Geofiz. (Soviet Geology and Geophysics)*, 25(5):89–95, 1984.

[Filippi and Frisch, 1969a] P. Filippi and U. Frisch. Relation entre l'équation de la chaleur et l'équation des ondes de helmholtz. *Comptes Rendus Acad. Sci.*, 268(A):804–807, 1969.

[Filippi and Frisch, 1969b] P. Filippi and U. Frisch. Relation entre l'équation de la chaleur et l'équation des ondes. In *Deuxieme Colloque sur le traitement du signal et ses applications*, pages 159–174, May 1969. In French.

[Gershenson, 1993] M. Gershenson. Simple interpretation of time-domain electromagnetic sounding using similarities between wave and diffusion propogation. In *SEG Annual International Meeting 93*, pages 1342–1345, 1993. Extended Abstract number SS2.34.

[Gibert and Virieux, 1991] D. Gibert and J. Virieux. Electromagnetic imaging and simulated annealing. *Journal of Geophysical Research-Solid Earth And Planets*, 96(NB5):8057–8067, 1991.

[Gibert *et al.*, 1994] D. Gibert, B. Tournerie, and J. Virieux. High-resolution electromagnetic imaging of the conductive earth interior. *Inverse problems*, 1994. In press.

[Incorporated, 1991] Silicon Graphics Incorporated. IRIS EXPLORER. Technical report, Silicon Graphics Incorporated, 1991.

[Lavrent'ev *et al.*, 1980] M. M. Lavrent'ev, V. G. Romanov, and S. P. Shishatskii. Ill-posed problems of mathematical physics and analysis. *Nauka*, 1980.

[Lee and Xie, 1993] K. H. Lee and G. Q. Xie. A new approach to imaging with low-frequency electromagnetic fields. *Geophysics*, 58(6):780–796, June 1993.

[Lee *et al.*, 1987] S. Lee, G. A. McMechan, and C. L. V. Aiken. Phase-field imaging: the electromagnetic equivalent of seismic migration. *Geophysics*, 52(5):678–693, May 1987.

[Lee *et al.*, 1989] K. H. Lee, G. Liu, and H. F. Morrison. A new approach to modeling the electromagnetic response of conductive media. *Geophysics*, 54(9):1180–1192, September 1989.

[Lee, 1988] K. H. Lee. A new approach to interpreting electromagnetic-sounding data. Technical report, Lawrence Berkeley Laboratory, 1988. Annual report 1988, LBL-26362.

[Levy *et al.*, 1988] S. Levy, D. Oldenburg, and J. Wang. Subsurface imaging using magnetotelluric data. *Geophysics*, 53:104–117, 1988.

[Mandelis, 1991] A. Mandelis. Theory of photothermal wave diffraction tomography via spatial Laplace spectral decomposition. *J. Phys. A: Math. Gen.*, 24:2485–2505, 1991.

[Oliver, 1994] D. Oliver. Application of a wave transform to pressure transient testing in porous media. *Transport in Porous Media*, 1994. in press.

[Petroleum, 1990] Shell Internationale Petroleum. Seismic surveying. *Shell Science and Technology*, December 1990.

[Thornborrow *et al.*, 1993] C. Thornborrow, A. J. S. Wilson, and C. Faigle. Developing modular application builders to exploit MIMD parallel resources. In *Visualisation 1993*, pages 134–141. IEEE, October 1993.

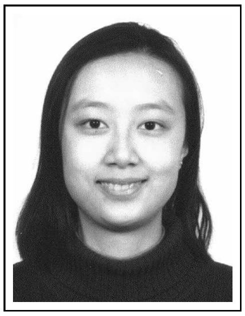
[Weidelt, 1972] P. Weidelt. The inverse problem of geomagnetic induction. *Zeit. fur Geophysik*, 38:257–289, 1972.

[Wilson *et al.*, 1991] A. J. S. Wilson, J. G. Mills, and M. G. Norman. *Bodyscan*: A transputer based 3D image analysis package. In *Applications of Transputers 3*, volume 2, pages 130–135. IOS, August 1991.

[Zhdanov and Frenkel, 1988] M. S. Zhdanov and M. A. Frenkel. Phase-field imaging - the electromagnetic equivalent of seismic migration - discussion. *Geophysics*, 53(6):863, 1988.

[Zorgati *et al.*, 1991] R. Zorgati, B. Duchene, D. Lesselier, and F. Pons. Eddy-current testing of anomalies in conductive materials (I) qualitative imaging via diffraction tomography techniques. *IEEE Transactions On Magnetics*, 27(6):4416–4437, 1991.

[Zorgati *et al.*, 1992] R. Zorgati, D. Lesselier, B. Duchene, and F. Pons. Eddy-current testing of anomalies in conductive materials, (II) quantitative imaging via deterministic and stochastic inversion techniques. *IEEE Transactions on Magnetics*, 28(3):1850–1862, May 1992.



Peilin Jia is a student of Computer Science at Yale University. After graduation in 1996 she hopes to pursue a career in law.
This project was supervised by Ian Flockhart (EPCC) and Andrew Wilson (Department of Geology and Geophysics, University of Edinburgh).

Contacts:
Peilin Jia
P.O. Box 229 Yale Station
New Haven, CT 06520
U.S.A.
Email: jia@cs.yale.edu

Ian Flockhart
University of Edinburgh,
Edinburgh Parallel Computing Centre,
The Kings Buildings,
West Mains Road,
Edinburgh EH9 3JW, UK.

Email: ianf@epcc.ed.ac.uk
Telephone: +44 31 650 4889
Fax: +44 31 650 6555

Andrew Wilson
University of Edinburgh,
Department of Geology and Geophysics,
The Kings Buildings,
West Mains Road,
Edinburgh EH9 3JW, UK.

Email: Andrew.J.S.Wilson@uk.ac.ed
Telephone: +44 31 650 8533
Fax: +44 31 668 3184
URL: http://www.glg.ed.ac.uk/ajsw