Model Order Reduction with Oblique Projections for Large Scale Wave Propagation

Victor Pereyra*

Abstract

There are many applications in which it is necessary to solve large scale parametrized wave propagation problems repeatedly. This is still quite a challenging task, even with the largest available computer clusters. In this paper we will discuss the application of Model Order Reduction (MOR) to problems in seismic petroleum exploration, with the aim of diminishing the necessary computing time by a significant factor. We consider POD and some variants. POD is a Model Order Reduction technique that uses snapshots of a few simulations in order to quickly compute related problems with similar accuracy. The method of lines via a Petrov-Galerkin approximation that uses the snapshots as basis functions is the considered approach. The order reduction comes from projecting the wave equation discretized in space to the subspace spanned by the snapshots. This has been shown earlier to work well in two dimensions. The challenge in three dimensions comes from the size of the spatial meshes required and the fact that the method usually requires a number of snapshots that do not fit in fast memory, even for current high end multicore machines. Parallelization is not an option since it is already used for other aspects of this massive problem.

1 Introduction

Seismic exploration is one of many approaches used to obtain information on the underground to locate and extract oil and gas. The seismic method is used inland and off-shore and consists of introducing man-made sources of energy that travels through the earth. The back scatter is recorded on sensors called geophones or hydrophones. It is this time data that is used to produce depth maps of the earth interior through seismic data processing. With the advent of powerful large scale clusters of computers it has become possible to simulate the full wave propagation at various levels of complexity (acoustic, elastic, anisotropic) in order to migrate the data from time to depth

*Energy Resources Engineering, Stanford University, California.

to produce three-dimensional maps of the material properties that affect wave propagation: density, wave speeds and anisotropic parameters.

A 3D seismic survey consists of many shots, i.e., activation of energy sources, say explosions, vibrator trucks or air-guns and corresponding receiver arrays. Usually, a simple rectangular geometry with equally spaced sources and receivers is used and only one source is activated at a time with a recording time of a few seconds, that depends on the depth one wants to image and the approximate velocity of the involved rock formations. Because of the complexity and the large scale of the 3D meshes involved the simulation of the time domain wave equation is performed using explicit methods on regular meshes, on a box that contains the region of interest. A typical 3D survey may have thousands of shots and for each shot thousands of receivers that record the back scatter, generating enormous data sets that have to be processed.

Repeated wave equation simulation is used for many different facets of this work, from survey planning and illumination studies, to imaging, migration from time to depth, reverse time migration, seismic tomography and quality control.

In recent work we have shown that model order reduction (MOR) is capable of reducing the computational cost of High Fidelity full acoustic wave simulation for shot interpolation and extrapolation in two-dimensions [6, 15, 27], when many problems with different source positions (shots) need to be simulated in a seismic survey or for seismic imaging. The requirement there is accuracy (albeit low, say 3 significant figures) and thus we cannot stray too far from the sources that produced the snapshots for MOR. We have shown that extrapolating one shot in each direction from the basis shot or interpolating two interior shots from two end shots works well when the data thus generated is used in a simulation process [27]. Taking into account the overhead, this doubles the speed at which we can perform this task, where many thousands of shots and corresponding simulations are required.

The challenge now is to extend this process to three-dimensions in a competitive manner. To fix our ideas, let us consider a typical 3D simulation that is performed today on a single multicore machine with 128 Gb of fast memory. The number of nodes in each direction of the spatial mesh are $nx = ny = 1541$, $nz = 613$, for a total of $n = 1,455,679,453$ mesh points. That is to say, the size of one snapshot in double precision would be approximately 11.6 Gb. From our experience in 2D and for the type of source frequencies desired, we expect to need more than 100 snapshots, i.e., just to hold a copy of the whole snapshot matrix in core would require more than 1.2 TB, far more than what is available in a modern single machine. Currently, each High Fidelity simulation of this size takes three hours in a Sandy bridge box (16 cores, 128 GB memory, 277 GHz). This corresponds to 16" of simulation with a time step $dt = 0.00191$ (9424 time steps). We could wait for Moore's Law to catch up, but then larger problems will come around: elastic and

anisotropic wave propagation, higher frequencies and so on, so that is not the answer.

In this paper we explore a number of avenues to solve this problem competitively using some new variants of MOR. We observe that a full seismic survey simulation requires many thousands of shots and that is where parallelism in a distributed system is already employed. So, parallelism is not the answer for a single shot problem that fits nicely in core for a high fidelity simulation.

For the application of interest, the integration domain is a half space that needs to be artificially limited on three sides, where absorbing boundary conditions should be imposed. Thus, the geometry is very simple (a box), although it would be of interest to have topography, i.e., a non-planar surface as the top boundary. For this type of large wave propagation problems it is now routine to use explicit high order methods on uniform meshes, since they are the most efficient and simple ones available.

The acoustic wave equation in three dimensions with a forcing term and absorbing boundary conditions can be writen as:

$$w_{tt} = v^2(x,y,z)\triangle w + bu(t) - 2\epsilon(x,y,z)w_t - \epsilon^2(x,y,z)w,$$

where v is the velocity of propagation and the function ϵ decays rapidly away from the artificial boundaries. First, this equation is discretized in space on a mesh of size $n = nx \times ny \times nz$, and $k << n$ snapshots are collected by running one or several High Fidelity simulations. The snapshots are composed of values of the field variables $w(x,y,x,t)$ at the points of the spatial mesh for selected times, ordered in a vector with indices running first in the z direction and then in the y and x ones. They are written in this vector form as columns of an $n \times k$ matrix S. An orthogonal basis U for its column space can be generated either by a truncated SVD process or by an adaptive QR algorithm (see Section 9 of [15]). We then assume that the solution can be approximated by $w = Ua(t)$. Replacing in the wave equation and discretizing the Laplacian in space we obtain the reduced order system:

$$a_{tt} = U^T AUa + U^T \mathbf{b}u(t) - 2U^T D(\epsilon)Ua_t.$$

Here the matrix A contains the discrete Laplacian plus the last term $-\epsilon^2 w$. We use an $8th$ order discretization of the Laplacian [7], which leads to an $n \times n$ sparse, structured matrix, with only 25 nonzero elements per row that is stored in sparse mode. $\mathbf{b}$ is a vector that describes where the forcing function $u(t)$ is applied. For a point source, $\mathbf{b}$ is all zeroes except at the source index, where it is equal to1. Finally $D(\epsilon)$ is a diagonal matrix. Thus, the main pre-processing task, besides of obtaining the snapshots and the orthogonal basis, is to calculate $U^T AU$. Either of these procedures requires all the snapshots to be present in fast memory to be competitive and since

for the problem sizes we are interested in this is not feasible we will explore in this paper other alternatives:

(a) The reduction to a lower order system can also be attained without orthogonalization, as we explain in the section on Oblique Projection, where a well conditioned basis is created using a progressive adaptive QR algorithm in reduced row space; (b) We can perform MOR using only a limited number of rows of the snapshot matrix (i.e., rows associated with spatial mesh points selected through a $n_r \times n$ matrix C) and (c) We can get snapshots from selected simulations by only integrating part of the total time.

We also consider the use of randomized algorithms [3, 10, 9] to speed up the linear algebra steps that are required to project the high fidelity equations into the reduced ones. Because of the availability of fast least squares solvers for large matrices we will also consider applying them to the oblique projection algorithm.

In order to reduce the computing cost even further we also investigate in Section 5 the use of incoherent encoded sources or super-shots. At this point, the winning combination seems to be MOR with oblique projection (i.e., no orthogonalization) and the use of the compacted snapshots with a limited number of rows or Monte Carlo abbreviated multiplication, to generate the coefficients and reduce the size of the least squares problems that arise. We give some preliminary numerical results for the oblique projection method for large problems in 2D and 3D.

Although most of the components of these approaches are not new, it is startling to find that very little work has been done to combine and apply them to the seismic exploration area, one of the industrial computations that uses a large number of flops on a regular basis.

2 MOR without orthogonality: Oblique Projections

We consider now the use of the snapshots directly in MOR, **without obtaining an orthogonal basis**. We assume then that $w(t) = Sa(t)$ and replace this expression in the original wave equation:

$$Sa_{tt} = ASa + \mathbf{b}u(t) - 2D(\epsilon)Sa_t. \tag{2.1}$$

Multiplying by S^T we get:

$$(S^T S)a_{tt} = S^T[ASa + \mathbf{b}u(t) - 2D(\epsilon)Sa_t].$$

The matrix coefficient of the first term in the righ-hand-side can be obtained by solving the matrix least squares problem:

$$SX = AS,$$

where X is a $k \times k$ matrix. We observe that $(S^T S)^{-1} S^T = S^+$ is the pseudoinverse of S and since $(S^T S)$ is not large we can use a truncated SVD procedure to solve this matrix least squares problem. It turns out that [12] also addresses exactly this problem and have produced a publicly available code that works even in the case that S is rank deficient. Unfortunately, in the case that S is ill-conditioned, this algorithm does not apply directly. One would need to have a procedure to whittle the columns of S into an $\tilde{S}$ that is well-conditioned and then apply the procedure mentioned above. An algorithm to select an independent set of columns from an arbitrary matrix and compute its pseudoinverse or solve a matrix least square problem has also been described in [16, 22], although we do not use it in this development.

3 MOR with a limited number of rows

We want to explore the possibility of only using a limited number of rows in 2.1, which, if possible and acurate, will go a long way to make this approach practical. For this we introduce:

$$r = Cw,$$

where the matrix C, $n_r \times n$, selects the elements of w that correspond to $n_r < n$ spatial positions (we assume that the source positions are included in that set) and packs them on a vector r of size n_r. Now, if we multiply equation 2.1 by C, the reduction proceeds in the usual fashion if we assume $w = Sa$:

$$CSa_{tt} = CASa + C\mathbf{b}u(t) - 2CD(\epsilon)Sa_t.$$

This step essentially selects the n_r equations corresponding to the desired output positions. In the finite element literature this is known as a Petrov-Galergin method [1].

Observe that the product CAS with the full matrix S can be performed columnwise in the High Fidelity (HF) code, as the snapshots are generated and then only n_r rows need to be saved to be read by the MOR code. The only proviso here is that the adaptive selection of a well-conditioned basis needs to be done before calculating this product and saving the snapshots. In fact, since even in the 3D case the sizes will be moderate if n_r is sufficiently small, we can go back to using adaptive QR to obtain an orthogonal basis U in the reduced space. In other words, we can calculate $CS = QR$, where Q is orthonormal and R is upper triangular and well conditioned and from now on S is a selected subset of all the snapshots inspected.

The other two terms in the right-hand-side only require the compacted n_r rows and therefore they can be calculated in the MOR code. This eliminates totally the large dimension n and should help to make possible running the

MOR algorithm within the memory limitations that we have. Replacing we get:

$$QRa_{tt} = CASa + C\mathbf{b}u(t) - 2CD(\epsilon)Sa_t.$$

Finally:

$$a_{tt} = R^{-1}Q^TCASa + R^{-1}Q^TC\mathbf{b}u(t) - 2R^{-1}Q^TCD(\epsilon)Sa_t$$

and

$$w = CSa,$$

where R^{-1} stands for solving an upper triangular system of linear equations. Observe that no abbreviated matrix multiplications are necessary, so in principle, there are no additional approximations. An intriguing thought is: would it be possible to reconstruct the whole field if we save S instead of only CS?

4 Randomized algorithms for large scale linear algebra

In recent times there has been considerable activity in the area of probabilistic algorithms for constructing approximate decompositions for very large matrices, as those that occur in big data mining, computer learning, neural networks, page ranking (Google search) and genome calculations, to name a few. We would be interested in obtaining U, an orthogonal basis for the range of the matrix S of snapshots, for the case in which neither S nor U fit in the fast memory of our computer.

4.1 Randomly sub-sampling rows and columns

Let S be the $n \times k$ matrix of snapshots, so large that it does not fit in fast memory. However, n is such that we can read a few snapshots at a time in fast memory. The following algorithm will enable the projection of the high fidelity system of equations:

1. Sample $s = r + p \leq k$ columns of S at random, with probability of chosing column S_i, $p_i^{col} = \|S_i\|_2^2/\|S\|_F^2$. Here r is the target rank and p is a small over-sampling number. Normalize the numbers p_i^{col} so that they add up to one and use them to define a probability distribution P.
2. Form $Y_{n\times s} = \frac{\|S\|_F}{\sqrt{s}}\left[\frac{S_{i_1}}{\|S_{i_1}\|_2}, ..., \frac{S_{i_s}}{\|S_{i_s}\|_2}\right]$.
3. Sample s rows of Y using the distribution P and form the matrix $W_{s\times s}$, similarly as we did with the columns.

4. Orthogonalize the columns of W to obtain $V_{s\times s}$. Calculate a good basis for the approximate range of S by multiplying by Y. Use YV to reduce the high fidelity equations by assuming that $u(t) = YVa(t)$:

$$YVa_{tt} = AYVa + \cdots,$$

$$a_{tt} = V^T(Y^TY)^{-1}Y^TAYVa + \cdots.$$

The first observation is that $YV_{n\times s}$ is not orthogonal and usually s will be larger than the number of columns that can fit in memory, so in principle it seems that we have not made much progress. However, the critical and potentially expensive solution of the normal equations $(Y^TY)^{-1}$ corresponds to a small $s\times s$ system. Also, for the large matrix-matrix multiplications exist fast approximate randomized algorithms that are trivially parallelizable [2]. So, in principle this would solve our problem. The question is to asses its cost, test it and validate it on realistic problems.

A second observation is that it seems that we could have applied the same idea directly to the original matrix S; however there is no guarantee that the selected set of snapshots has full rank or is well conditioned, so the normal equations are potentially dangerous to use, under those circumstances. Since the resulting problem now is manageable we can use truncated SVD's instead.

With regards to the random sampling of rows or columns we need first to create the cumulative density function C:

$$C_j = \sum_{i=1}^{j} p_i, \quad j = 1, \ldots, n \text{ (or } m),$$

where $C_0 = 0$ and C_n or $C_m = 1$ (columns or rows). Then we get a random number $0 \le r \le 1$ and choose j such that

$$C_{j-1} \le r < C_j.$$

A similar approach is advocated by [3], with the additional advantage that they provide a public implementation in [11]. We are interested here in the fixed-precision problem, where the rank of the approximation is determined by the software, given a computational tolerance. Once a basis has been selected, then we can use this as the S in Section 2.

4.2 Randomized matrix products

As indicated above, there exist fast Monte Carlo algorithms for various matrix operations, which may help in handling very large matrices, both in terms of efficiency and memory requirements [2, 10]. Given a $m \times n$ matrix A and a $n \times p$ matrix B, the key to the method of abbreviated multiplication is the identity:

$$AB = \sum_{i=1}^{n} A^{(i)} B_{(i)},$$

where $A^{(i)}$ is the ith column of A and $B_{(i)}$ is the ith row of B. The algorithm now corresponds to selecting at random c terms in this sum ($c << n$), according to a probability distribution $\{p_i\}_{i=1}^{n}$, and scaling each term in an appropriate manner. In other words, the product AB is approximated by an abbreviated product

$$CR = \sum_{t=1}^{c} C^{(t)} R_{(t)} = \sum_{t=1}^{c} \frac{1}{cp_{i_t}} A^{(i_t)} B_{(i_t)}.$$

The probability distribution could be uniform (that does not require a priori access to the matrices) or the one indicated in the previous section. Drineas et al [2] show in Lemma 11 that

$$\|AB - CR\|_F = \mathcal{O}(\|A\|_F \|B\|_F / \sqrt{c}).$$

4.3 LSRN and a variant

In [12] the authors consider algorithm LSRN for solving large least squares problems using a random projection to produce a pre-conditioner for an iterative method. They demonstrate theoretically the quality of this pre-conditioner, independently of the chosen random projection.

Let the $n \times k$ matrix A be the matrix of coefficients of the least squares problem and ρ its rank. In our case we are interested in problems for which $n >>> k \geqq \rho$, and therefore we will look at the first algorithm LSRN.

Algorithm LSRN

1. Choose an oversampling factor $\gamma > 1$ and set $s = \lceil \gamma k \rceil$.
2. Generate $G = random(s, n)$, i.e., an $s \times n$ random matrix whose entries are independent random variables following the standard normal distribution.
3. Compute $\tilde{A} = GA$.
4. Compute $\tilde{A} = \tilde{U}\tilde{\Sigma}\tilde{V}^T$, where $r = rank(\tilde{A})$, $\tilde{U}$ is a $s \times \rho$ orthogonal matrix (not needed), $\tilde{\Sigma}$ is a $\rho \times \rho$ diagonal matrix and $\tilde{V}$ is orthogonal $k \times \rho$.
5. Let $N = \tilde{V}\tilde{\Sigma}^{-1}$.
6. Compute the minimal length solution to $min_x \|ANy - b\|_2$ using an iterative method. Let $\hat{y}$ be the solution.

7. Calculate $\hat{x} = N\hat{y}$.

One additional difficulty in our problem is that n can be so large that A does not fit in fast memory. Worse still, G is even larger, so the product GA will probably require one of the techniques for approximate multiplication [10]. Also, distributed computing is only allowed at the level of a multi-core box, since the complete simulation or imaging problem requires thousands of shots that are distributed to a network of multi-core machines. In essence, we are seeking to speed up the global calculation by making each simulation (or group of simulations) more efficient, without sacrificing accuracy. If the generation of random numbers is sufficiently fast, an alternative to storing A and G is to process A by columns, re-generating the random matrix G always starting from the same seed.

4.4 Calculating the matrix coefficients for large problems

As we observed in Section 3, there are problems in which a substantial reduction in dimensionality can be attained by working only on a reduced array. However, there are other applications, such as Reverse Time Migration, where the response in the whole spatial mesh is required. In this case and when all the snapshots do not fit in the memory available we need to choose carefully the algorithm and see how we can operate with one single snapshot at a time. At this point we can only say that SVD, the Rosen-Pereyra algorithm [16] and progressive QR will not qualify, since they require the presence of all the snapshots or their orthogonal proxies.

Therefore we take a fresh look into the oblique projection algorithm of section 2. Since the snapshots are not to be orthogonalized, the HF code only needs to produce and output them one at a time. As we will see below, if we use randomized algorithms, it turns out that we will only need to save and process a limited (and small) number of rows of the matrix of snapshots. The MOR code then will read this information and calculate the necessary matrix of coefficients approximately, by using the algorithm of section 4.2. The two most complex coefficients to be calculated are: $S^T S$ and $S^T AS$.

Observe that the dimensions are apropriate for the algorithm of section 4.2: S^T is $k \times n$, with $k <<< n$. So, we proceed to select randomly a priori in the HF code the indices of $c = \gamma k$ columns of S^T (i.e., rows of S) using an uniform distribution. The oversampling factor $\gamma > 1$ will have to be chosen experimentally, once the accuracy impact of this approximate computation is gauged. In any case, we expect $c << n$. From now on we add a ^{c} to a matrix name to indicate that the $ind(i),\ i = 1, ..., c$ rows are stored compacted in positions $i = 1, ..., c$.

Therefore, we need only to write and read a matrix S^c of size $c \times k$, which will not tax the memory. Likewise, only the appropriate c rows of A need to be

generated and saved. Since A's rows only have at most 25 nonzero elements, this provides great savings, both in storage and in operations. Thus, we first perform the random multiplication $A^c S$ and then random multiply the result with S^{Tc}. The least squares part requires the solution of a square system with the approximate normal equations $S^{Tc} S^c$ $(k \times k)$. This can be done with a truncated SVD algorithm, in case the basis is ill conditioned.

Even for $k = 1000$ and $c = 2000$ this is quite a feasible program, within the memory constraints we have. The remaining question is related to the accuracy of the final calculation. The interesting part is that the first product $A^c S$ can be performed exactly by generating only $(c \times 25)$ numbers, a very modest proposition. This needs to be done in the HF code, one snapshot at a time. By saving this product there is no need to generate the matrix A in the MOR code.

4.5 Accuracy of MC product

We offer now some insight on the accuracy that can be achieved with the algorithm of section 4.2. We consider two random matrices A (100×10000) and $B = 10000 \times 100$ and apply the algorithm for all values of c. The Frobenius norm that is plotted is: $\|AB - CR\|_F / \|A\|_F \|B\|_F$. According to the theory, we expect this quantity to be of the order of $1/\sqrt{c}$. We observe in the results that the error decreases rapidly and after a while this decrease tapers down and then continues much more slowly.

In case there is any doubt about the necessity of all the details of the carefully crafted algorithm of [2, 10], we try some reasonable variants. For instance, we see that not allowing repetitions is OK at the begining, but eventually, not only this is worse than Drineas et al, but at the end does not even go to full roundoff accuracy. The sequential, non random algorithm, is not even worth discussing.

For our case, where $n \sim 10^{10}$, we can expect reasonable accuracy relatively earlier. It remains to be proven how far we need to go in order to achieve the ultimate accuracy required by the overall problem. Observe that for the given size of current top box memory of 32×10^9 single precision numbers, we can stretch $\gamma k \approx 10^5$ in the approach above, without running out of memory.

This algorithm rational is based on the fact that these matrices have low rank $(r \leq k)$ and therefore they contain much redundant information with regards to the subspaces that they span. That is why it is plausible that we can obtain a good approximation of the product if we sample a small number c of the large dimension lines and scale the sum of their rank one products appropriately.

In terms of selecting dynamically a well-conditioned subset of snapshots it is worth considering the results and algorithm of [25].

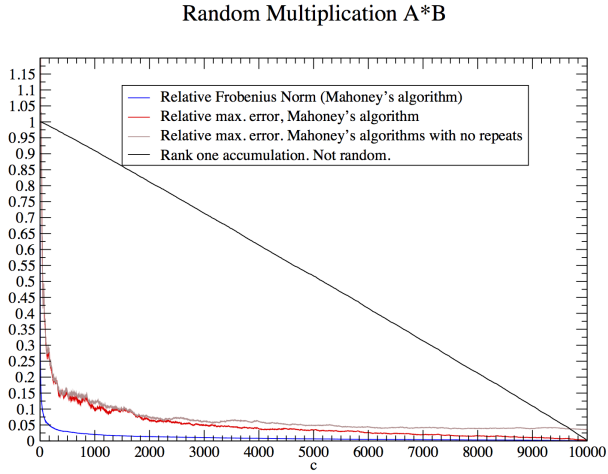

Figure 4.1: A (100×10000), B (10000×100)

4.6 Random vs deterministic and working on a reduced net

One impossible dream, for applications in which we only need the response at a limited number of points (usually a net with dimension one less than the one for the whole model), is to work only in this sub-net. For wave propagation this is impossible, since we need to develop the whole space field in time given that everything is coupled and we need to account for the scattering in depth.

However, the above ideas lead to an interesting alternative that amounts to working on a limited mesh of receivers (the points on the spatial mesh where we extract time history information, i.e., seismograms) in the MOR simulation. This is plausible, since the snapshots would contain the full spatial information from the response in the model.

Unfortunately, this idea would require to select the rows of the matrix of snapshots that correspond to these positions in the abbreviated multiplication process. That is, of course, not random, so the question is how the accuracy of the reduced products would be affected. In order to get an idea about this we have performed an experiment similar to the one above, in which we compare the MC product with one in which we use a deterministic (sequential) selection of columns/rows in the product $A^T A$, for a 100×10000 random matrix, for all values of c. The results can be viewed in Figure 4.2, where we crossplot the maximum relative error of the difference between the

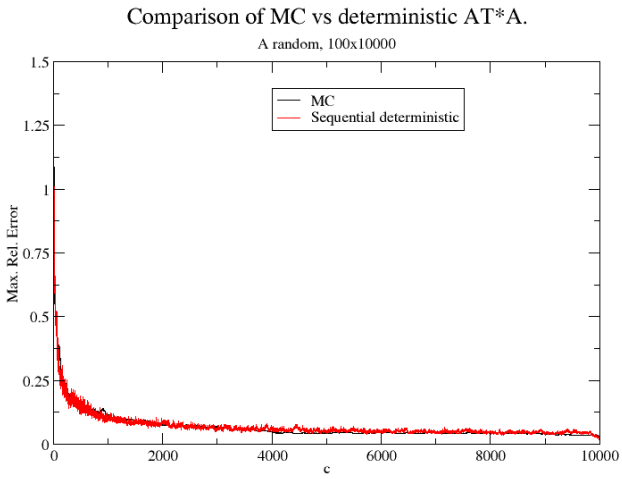

Figure 4.2: Crossplot of maximum relative error for MC and deterministic approximate matrix product

exact and approximate products. The two curves superpose, while the error for the deterministic approach seems to be more oscillatory as a function of c.

In reality, as we showed in Section 3, one could work completely in the subnet, without resorting or thinking of the whole mesh at all. That would not require abbreviated products at all and would virtually reduce the dimensionality of the problem. This is quite a novel idea so it needs to be tested in order to understand its implications.

4.7 Implementation details

The product $S^T S$ fits exactly the case described in detail in section 4.2, with $A = B^T$, so it requires no further explanation.
The product $S^T(AS)$ is somewhat more complicated, so we explain in detail how to perform it without ever having to handle matrices with the large dimension n. We recall that an ^{c} indicates that the matrix is compressed, with the rows $ind(i)$ in the first i positions.

- In the HF code we generate the uniformly random distributed indices $ind(i) \epsilon N,\ i = 1, ..., c$. Here N is the set containing the integers from 1 to n. Save and output only these c rows as S^c (of dimension k). Calculate the first product $A^c S$ exactly, one column at a time as the snapshots are generated and save the result in D^c $(c \times k)$:

$$D^c = A^c S;\ D^c_{jl} = \Sigma_{i=1}^{RowLength(j)} A^c_{j,colind(i,j)} S_{i,l},\quad j = 1, ..., c;\quad l = 1, ..., k.$$

- In MORslantMC (Model Order Reduction with slanted projection and Monte Carlo matrix multiplication), read in the matrix S^c $(c \times k)$. Read also D^c. Finally, perform the second product in the standard MC fashion: $S^T(AS) \sim S^{Tc}D^c = \Sigma_{i=1}^{c} S^{c(i)} D^c_{(i)}$. Observe that at no point we have needed to save a matrix with dimensions including n.

- Once the integration in time is completed we need to reconstruct the time response at the selected space points. Here is where the observation of 4.6 comes handy. In fact, this observation needs to be applied from the begining, instead of the random choice, so we would need to evaluate its effect. The comparison of 4.6 is encouraging.

5 Using encoded sources[1]

We have seen both in 2D and 3D that an important part of the cost of MOR comes from the need of obtaining snapshots from a number of HF calculations. An interesting idea used in 3D full waveform inversion may help in reducing this cost [8]. The idea is to activate multiple sources at once, thus requiring fewer simulations. The question to be answered is if from these combined snapshots, one can obtain an appropriate basis for Model Order Reduction. We call these "supershots".

Encoded simultaneous-source methods have been proposed to improve the efficiency of seismic acquisition and processing [5, 8, 21]. It is argued that incoherently encoded gathers illuminate more of the model than a point-source one and that they reduce the total number of simulations required. It is likely that we may need to use multiple encodings to better represent the relevant subspace of snapshots. Thus, the balance of all these factors need to be explored, in order to see if this idea provides a net gain from our previous implementation. We do that first in 2D using vel3l, a constant velocity three layers model, for a 2" integration with a $5Hz$ source.

After an implementation of a solver where several sources are activated at the same time and added together with randomized signs, we summarize below the best results. The important main one is that with three HF simulations, we can have more sources involved in MOR via this approach, with less full simulations and better accuracy than our previous one source at a time generation of snapshots. We see by comparing the different columns that one can obtain a speedup of 2.86 by this simple device, plus better accuracy. The error headings indicate which shots are included in a super-shot (ss) and similarly for the successive single-shot simulation (succ.).

[1] We thank Josep de la Puente, Barcelona Supercomputer Center, for suggesting this approach and pointing us to Krebs et al paper [8].

Source	Error SS2	Error SS3	Error (succ. 1,9)	Error (succ. 3,7)
1	0.011	0.029	0.004	-
2	0.021	0.018	0.13*	-
3	0.036*	0.0098	0.069*	0.00068
4	0.033*	0.0023	0.057*	0.036*
5	0.028	0.0043	0.051*	0.051*
6	0.029	0.0042	0.046*	0.042*
7	0.015	0.023	0.049*	0.1*
8	0.027*	0.026	0.14*	-
9	0.028	0.022	0.013	-
10	0.038*	0.026	-	
11	0.051*	0.034*	-	
Time HF	1040.64	1040.64	851.43	478.95
Time MORQR	242	364.11	205.9	204
Ratio	4.3	2.86	4.1	2.34
Rank	94	102	93	83

Table 1: Error and time comparisons for several strategies. SS2 (two repetitions) and SS3 (three repetitions) activates sources 1,3,5,7,9,11 at once.

The measure of the error we use is an average, relative RMS between the High Fidelity and the corresponding MOR results for one trace (i.e., a time history at a spatial point):

$$\sqrt{\Sigma_{i=1}^{N}(u_i^{HF} - u_i^{MOR})^2/(N \times max_i|u_i^{HF}|)}.$$

A level of error less than 0.03 corresponds roughly to acceptable small deviations in amplitude in cross-plots. There are three possible sources of error: amplitude, phase and unwanted oscillations. It would be good to be able to discriminate between them, but in the meanwhile it is prudent to check the cross-plots. We have marked with * those shots that may not be accurate enough. The best results for an 11 shot span correspond to three randomized repetitions of a supershot that includes every other shot.

Other strategies, such as activating more adjacent sources at the same time did not give good results, probably because of excesive cross-talk. Using one non-random super-shot (i.e., with all positive sources) gave reasonably good results, but it seems that the repetitions, which require randomization, are important to achieve the necessary variety of snapshots and correspondingly better accuracy. These results were obtained using the QR based algorithm for orthogonalizing the snapshots.

Observe that the supershot approach requires that MOR calculates all the individual shots, including those active in the supershot, since they are all mixed up and cannot be separated. However, since the MOR integration

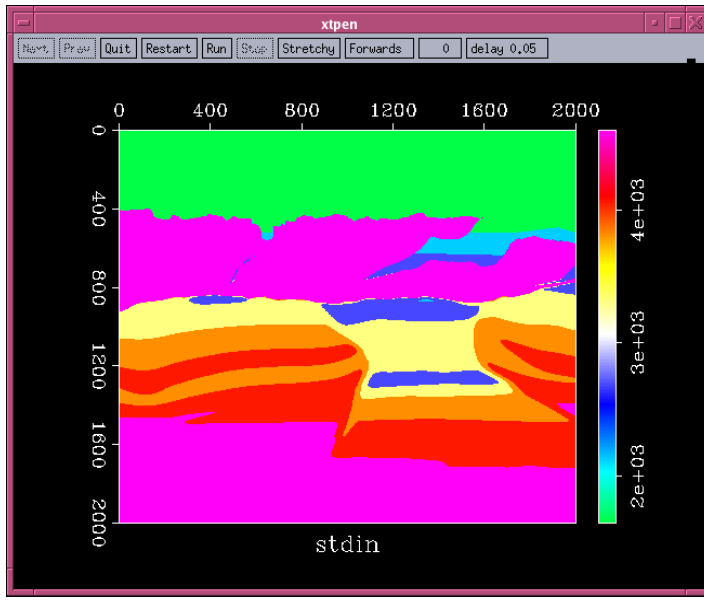

Figure 6.1: Model vel4pc velocity

is so fast, the additional cost is negligible.

6 A large scale 2D numerical example

We consider now *vel4pc*, a 2001×2001 $2D$ model of complex geology. See Figure 6.1 for a depiction of its velocity mesh.

The number of time step stations (where we potentially inspect snapshots) is 400, with $dt = 0.005$", for a total integration interval of 2" and a 5 Hz Ricker wavelet as the source. We consider 10 shots spaced by 6.25 units and extract a trace (i.e., time history) at $[900, 5]$ for comparison of the HF and MOR codes. We exercise the options of using less than the total integration interval in selected snapshots for the code HFS, namely: for the end shots, S_1 and S_{10} we use the first 300 time stations, while for the inner shots we only inspect the first 30 ones, for a total cost of about 2.1 full integrations. With a threshold at the end points of $\epsilon = 0.075$ and $\epsilon_i = 0.9^{i-1} * \epsilon,\ i = 2, ..., 9$ for the interior ones, a well-conditioned basis of 164 snapshots is selected by the adaptive QR algorithm applied to CS. We also consider the matrix C so that only 200100 consecutive points are used in HFS and MOR, starting at 1000500. In Table 2 we list the maximum absolute errors for the difference between the HF and MOR traces and in Figure 6.2 we show crossplots for the best and worst cases (S_1,S_5).

Shot	1	2	3	4	5
Error	0.00017	0.00053	0.00091	0.0012	0.0013
Shot	6	7	8	9	10
Error	0.0013	0.0012	0.00095	0.00057	0.00025

Table 2: Maximum absolute errors for the control trace

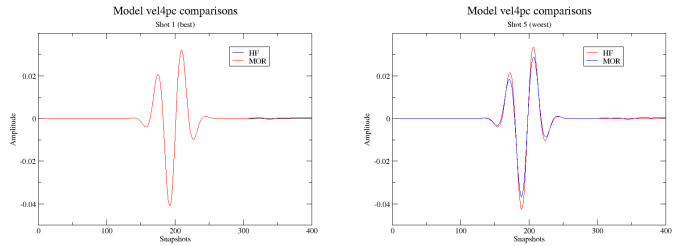

Figure 6.2: Crossplot of best and worst shot

Observe that for this example there is no action after time station 300. If there were significant signal after time station 300 then we could not ignore this final interval when seeking useful snapshots from the end sources, as we have seen in other examples. Again we see that phases are very good, while there is a miss in amplitude matching, which is visible at this scale. We observe in Table 3 that a speed up by a factor of 5 has been achieved by been able to use a significantly larger number of shots to share the overhead. This should be even more pronounced in 3D. We have used no parallelization (one thread only for the three codes).

7 3D layered media example

We consider now a 3D version of model *vel3l, vel3l3D*. The specs are:

Method	Time (sec.)
HF	249579
HFS	48510
MOR	1168
Ratio	5.0

Table 3: Performance

$nx = ny = nz = 176$, $nt = 200$, $n_{sngl} = 500000$,
$dx = dy = dz = 10m$, $dt = 0.005s$.
$threshold = 0.01$, $\delta threshold = 0.9$. 9 shots in a 3×3 square.
Frequency of Ricker source wavelet is $5Hz$.

The matrix C selects 500000 consecutive rows starting at 1982465 and the trace is collected at point 2182405. The sources are spaced by $10m$ in each direction, with the first one at coordinates $(71, 71, 5)$, i.e., at point 2180645. We recall that the order of the mesh in vector form proceeds, from slow to fast indeces (x, y, z).

We collect snapshots from the corner shots $(\#1, 3, 7, 9)$ for all timesteps and also from the interior shots, but only for the first 10 timesteps, for a total of $4, 25$ HF integration intervals. Of, course, as we see in Table 4, only a subset of snapshots are selected. In conclusion, the limiting speedup for this example is $9/4.25 = 2.12$. The threshold is multiplied by $\delta threshold$ for each successive shot, in order to diminish the bias of the basis been more heavily populated with snapshots from earlier shots.

We are interested at this point in assesing how much we can reduce the number of rows (size of matrix C) and the effect of the adaptive snapshot selection strategy, including the short integration for the interior shots, all part of our algorithm. For this example we are reducing the number of rows by a factor of $176^3/500000 = 5451776/500000 = 10.9$.

In Table 4 we describe the results and in Table 5 we post the computing times for the different codes:

HF3D, the High Fidelity code that runs all the shots and produces and output trace for comparison only.

HFS3D, the High Fidelity code that runs the limited integrations, extracts adaptively the snapshots S, produces the QR decomposition of the abbreviated version $CS = QR$, calculates CAS and outputs these matrices to be used by the MOR code $MORS3D$.

Finally, $MORS3D$ reads all the necessary information, calculates the remaining matrix coefficients and performs the time integration in the projected spaces. The RMS error was defined in Section 5 and the rank corresponds to the number of snapshots selected at the end of the integration for each shot. In Figure 7.1 we crossplot the corresponding time traces for Shot 2, the one with the worst accuracy in the whole calculation. We see that eyeball accuracy is quite good, and of course it is better for the more accurate shots. All the calculations were performed on an 8 $CPU's$, $2300MHz$ machine with 3.3 GB of fast memory, with a single thread especified.

Next we consider a 5×5 square array of sources. The specs for this larger experiment are:

$\varepsilon = 0.025$, $\delta\varepsilon = 0.8$, $ncut = 10$, $n_{sngl} = 500000$, $nsteps = 200$.

We perform full integrations at the corners but use only 20 time steps for every other of the interior ones. In Table 6 we show the shot pattern:

Shot	1	2	3	4	5	6	7	8	9
Rank	41	51	89	99	109	119	150	160	185
RMS	1.3(-5)	4.0(-3)	1.7(-5)	1.1(-3)	2.8(-3)	2.1(-3)	1.1(-4)	3.8(-3)	2.1(-3)

Table 4: Rank and accuracy for each shot in 3D model vel3l3D. $a.b(-c) = a.b \times 10^{-c}$.

Code	Time (sec)
HF3D	147096
HFS3D	52536
MORS3D	736
Ratio	2.76

Table 5: Comparative time performance for 3D model v3l33D. 9 shots.

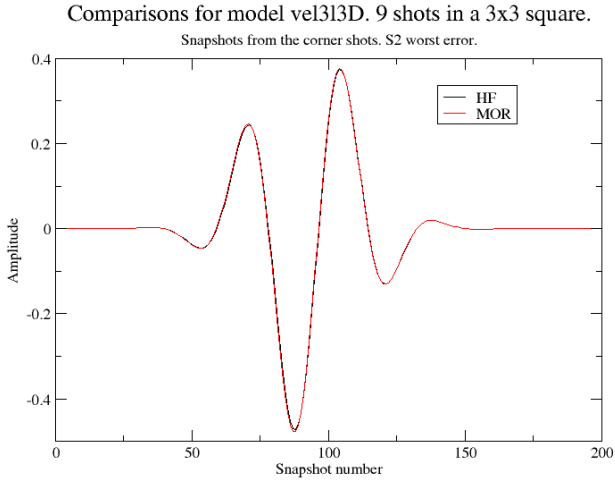

Figure 7.1: Crossplot of trace for worst error in model vel3l3D. Shot 2.

$$\begin{bmatrix} X & 0 & \triangle & 0 & X \\ 0 & 0 & 0 & 0 & 0 \\ \triangle & 0 & \triangle & 0 & \triangle \\ 0 & 0 & 0 & 0 & 0 \\ X & 0 & \triangle & 0 & X \end{bmatrix}$$

Table 6: Shot integration pattern: X full integration; 0 no integration; Δ $nsteps/ncut = 20$ steps.

Code	Time (sec)
HF3D	26617.46
HFS3D	7338
MORS3D	661
Ratio	3.33

Table 7: Time comparisons for problem vel3l3D using 25 shots.

We run this experiment in a 24 core machine with 100 GB of memory. We use the parallelized versions in OpenMP of the integration loops of HF3D and HFS3D, activating 12 cores. Although MORS3D is not paralellized, the relative cost is essentially negligible, so this does not distort the comparisons in any significant way. The rank is 220 and the maximum error of 0.015 occurs at Shot 10. There are small visible missmatches, but overal the accuracy is acceptable. The time performance is shown in Table 7.

8 Full survey simulation

A full survey simulation involving many thousands of shots will be partitioned in small subsets of the sort that we have tested above in order to use MOR. The easiest implementation is one in which the subsets are adjacent but do not overlap. In this case they can be calculated in parallel and no comunication will be necessary.

If we allow overlaps, considerable savings in computing time can be effected at the cost of programming complexity, increased storage and read/writes between fast memory and auxiliary storage. As usual these costs need to be balanced in order to attain a positive gain. We explain next a possible strategy that extends to a 2D array of sources what we did for a line of sources [27]. In the 2D modeling case we needed only to save the snapshots corresponding to the far end of the segment in order to reuse them for the next segment. Now we have two directions to attend to and things are more complex.

Algorithm

Collect the snapshots associated with each source in a single binary file with a header indicating the model name, shot number, length and number of snapshots. Assuming that we have a rectangle of subsets that form the whole shot array, the suggested algorithm proceeds as follows:

Start with the first row. After the first block is computed save the South (S) and East (E) snapshots.

For the next blocks in row one, re-use the E side of the previous block as the West (W) side of the new block, replace the old E side by the new one and save the new S side.

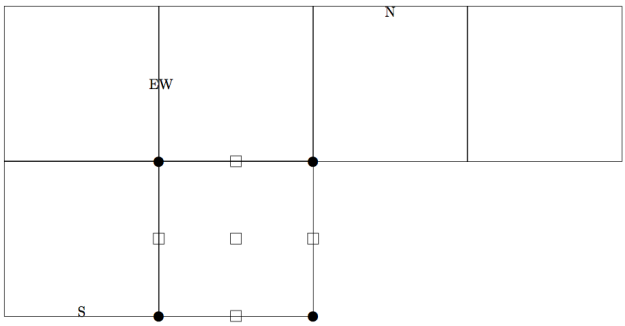

Figure 8.1: Schematic diagram of a full survey simulation

From block row 2 on, re-use S side of the block above and replace it by the new S side. Similarly with the E side (after the first block). Once a row is finished go to 3 until the whole survey has been simulated. For an interior 5×5 block as exemplified above, instead of 5 full integrations (assuming 1/10 of time steps for every other shot in the interior) we would have only 1.333 integrations for a $3.75\times$ save. For our example above that would imply a total speedup factor of approximately 12.5! Of course, for an especific problem one would also have to consider the possible disk access penalties. In the diagram of Figure 8.1 we present an schematic diagram of the geometry used.

8.1 Conclusions

We have described an important set of problems pertaining to oil exploration by seismic methods. It is now routine to use full wave equation simulations to try to image the earth interior. Many such related simulations are required and we have discussed the application of Model Order Reduction in an attempt to decrease the computational effort by a significant factor. We have already shown in 2D that these techniques are effective, but in 3D the problems are very large and tax the largest memories available, especially for MOR based on snapshots. Thus we have reviewed a number of techniques that potentially can help and we have shown numerical results for some of them that indicate that one can speed up these large calculations by an order of magnitude. The large 2D and 3D results have been obtained by combining slanted projections, deterministic reduction and compactification of the number of rows (spatial mesh positions) of the problem and adaptive QR to select a well conditioned basis of snapshots for the reduced problem. We did some preliminary studies of randomized linear algebra methods that are potential additional tools that can be used for this or similar problems, but

we have not integrated them within the solvers yet.

References

[1] K. Carlberg, C. Bou-Mosleh, and C. Farhat, *Efficient nonlinear model leduction via a least-squares Petrov-Galerkin projection and compressive tensor approximations.* Int. J. Numer. Meth. Engng. **86**:1-25 (2011).

[2] P. Drineas, R. Kannan and M. W. Mahoney, *Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication.* SIAM Journal on Computing **36**:132-157 (2006).

[3] N. Halko, P. G. Martinsson and J. A. Tropp, *Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. SIAM Review* **53**:217–288 (2011).

[4] P. C. Hansen, V. Pereyra and G. Scherer, Least Squares Data Fitting with Applications. Johns Hopkins University Press, Baltimore (2013).

[5] L. Ikelle, *Coding and decoding: Seismic data modeling, acquisition and processing.* SEG Extended Abstracts **77**:66-70 (2007).

[6] B. Kaelin and V. Pereyra, *Fast wave propagation by model order reduction.* ETNA **30**:406-419 (2008).

[7] H. B. Keller and V. Pereyra, *Symbolic generation of finite difference formulae.* Math. Comp. **32**:955-971 (1978).

[8] J. R. Krebs, J. E. Anderson, D. Hinkley, R. Neelamani, S. Lee, A. Baumstein and M-D. Lacasse, *Fast full-wavefield inversion using encoded sources.* Geophysics **74**:WCC177-WCC188 (2009).

[9] M. W. Mahoney, *Approximate computation and implicit regularization for very large-scale data analysis.* Proc. of the 2012 ACM Symposium on Principles of Database Systems, 143-154 (2012).

[10] M. W. Mahoney, *Randomized algorithms for matrices and data, Foundations and Trends in Machine Learning*, NOW Publishers **3:**123-224 (2011).

[11] P-G. Martinsson, V. Rokhlin, Y. Shkolnisky and M. Tygert, *ID: A software package for low-rank approximation of matrices via interpolative decompositions. Version 0.4.* Yale University, New Haven, Conn. (2014).

[12] Xiangrui Meng, Michael A. Saunders and Michael W. Mahoney, *LSRN: A parallel iterative solver for strongly over- or under-determined systems.* SIAM J. Sci. Comput., 36:C95–C118 (2014).

[13] R. Neelamani, C. E. Krhon, J. R. Krebs, M. Deffenbaugh, J. E. Anderson and J. K. Romberg, *Efficient seismic forward modeling using simultaneous random sources and sparsity.* SEG Expanded Abstracts **78**:2107-2111 (2008).

[14] Fabian Ojeda, Johan A.K. Suykens and Bart De Moor, *Low rank updated LS-SVM classifiers for fast variable selection.* Neural Networks **21**:437-449 (2008).

[15] V. Pereyra, *Wave equation simulation in two-dimensions using a compressed modeler.* American Journal of Computational Mathematics **3**:231-241 (2013).

[16] V. Pereyra and J. B. Rosen, *Computation of the pseudoinverse of a matrix of unknown rank.* Stanford University Computer Sciences Report SC18 (1964).

[17] V. Pereyra and G. Scherer, *Efficient computer manipulation of tensor products with applications in multidimensional approximation.* Math. Comp. **27**:595-605 (1973).

[18] V. Pereyra and G. Scherer, *Least squares scattered data fitting by truncated SVD's.* Applied Numerical Mathematics, **40**:73-86 (2002).

[19] V. Pereyra and G. Scherer, *Large scale least squares data fitting.* Applied Numerical Math. **44**:225-239 (2003).

[20] V. Pereyra and G. Scherer, *Least squares collocation solution of elliptic problems in general regions.* Mathematics and Computers in Simulation **73**:226-230 (2006).

[21] L. A. Romero, D. C. Ghiglia, C. C. Ober and S. A. Morton, *Phase encoding of shot records in prestack migration.* Geophysics **65**:426-436 (2000).

[22] J. B. Rosen, *Minimum and basic solutions to singular linear systems.* SIAM J. **12**:152-162 (1964).

[23] K. Schwarz, *Darts, dice, and coins: sampling from a discrete distribution.* Manuscript (2011).

[24] M. Seeger, *Low rank updates for the Cholesky decomposition.* University of California at Berkeley, Tech. Rep. (2007).

[25] Joel A. Tropp, Column subset selection, matrix factorization, and eigenvalue optimization. SODA '09 Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms. SIAM, Philadelphia, PA pp. 978-986 (2009).

[26] M. D. Vose, *A linear algorithm for generating random numbers with a given distribution.* IEEE Trans. Soft. Eng. **17**:972-974 (1991).

[27] Chunling Wu, Dimitri Bevc and V. Pereyra, *Model order reduction for efficient seismic modeling.* SEG Annual Meeting (2013).