Originally published as:

Venetis, I. E., Saltogianni, V., Stiros, S., Gallopoulos, E. (2020): Multivariable inversion using exhaustive grid search and high-performance GPU processing: a new perspective. - Geophysical Journal International, 221, 2, 905-927.

<https://doi.org/10.1093/gji/ggaa042>

Multivariable inversion using exhaustive grid search and high-performance GPU processing: a new perspective

Ioannis E. Venetis¹,^{1b} Vasso Saltogianni,^{2,3} Stathis Stiros³ and Efstratios Gallopoulos¹

¹University of Patras, 26504 Rio Achaia, Department of Computer Engineering & Informatics, Greece. E-mail: venetis@ceid.upatras.gr

²Helmholtz-Centre Potsdam, GFZ German Research Centre for Geosciences, 14473 Potsdam, Germany

³University of Patras, 26504 Rio Achaia, Department of Civil Engineering, Greece

Accepted 2020 January 22. Received 2020 January 22; in original form 2019 June 4

SUMMARY

Exhaustive searches in regular grids is a traditional and effective method for inversion, that is numerical solution of systems of non-linear equations which cannot be solved using formal algebraic techniques. However, this technique is effective for very few (3–4) variables and is slow. Recently, the first limitation was to a major degree overpassed with the new TOPological INVersion (TOPINV) algorithm which was used for inversion of systems with up to 18, or even more unknown variables. The novelty of this algorithm is that it is not based on the principle of the mean minimum misfit (cost function) between observations and model predictions, used by most inversion techniques. The new algorithm investigates for each gridpoint whether misfits of each observation are within specified uncertainty intervals, and stores clusters of ‘successful’ gridpoints in matrix form. These clusters (ensembles, sets) of gridpoints are tested whether they satisfy certain criteria and are then used to compute one or more optimal statistical solutions. The new algorithm is efficient for highly non-linear problems with high measurement uncertainties (low signal-to-noise ratio, SNR) and poor distribution of observations, that is problems leading to complicated 3-D mean misfit surfaces without dominant peaks, but it is slow when running in common computers. To overcome this limitation, we used GPUs which permit parallel processing in common computers, but faced another computational problem: GPU parallel processing supports only up to three dimensions.

To solve this problem, we used CUDA programming and optimized the distribution of the computational load to all GPU cores. This leads up to 100x speedup relative to common CPU processing, as is derived from comparative tests with synthetic data for two typical inversion geophysical problems with up to 18 unknown variables, Mogi magma source modeling and elastic dislocation modeling of seismic faults. This impressive speedup makes the GPU/CUDA implementation of TOPINV practical even for low-latency solution of certain geophysical problems.

This speedup in calculations also permitted to investigate the performance of the new algorithm in relation to the density of the adopted grids. We focused on a typical problem of elastic dislocation in unfavorable conditions (poor observations geometry, data with low SNR) and on synthetic observations with noise, so that the difference of each solution from the ‘true’/reference value was known (accuracy-based approach). Application of the algorithm revealed stable, accurate and precise solutions, with quality increasing with the grid density. Solution defects (bias), mainly produced by very coarse grids, can be identified through specific diagnostic criteria, dictating finer search grids.

Key words: Inverse theory; Numerical modelling; Numerical solutions; Statistical methods.

1 INTRODUCTION

Various processes or problems in Geophysics are described by equations f :

$$f(\mathbf{x}) = \mathbf{d} \quad (1)$$

relating a variable $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$ with another variable $\mathbf{d} = (d_1, d_2, \dots, d_m)^T$. An example is given in Fig. 1 which indicates the relationship between an inflating small magma source (variable $\mathbf{x}$) and of ground deformation (3-D dislocation $\mathbf{d}$ of a point P) described by equations f based on the elastic theory. If there are available a number of measurements of $\mathbf{d}$ (for example at different

Figure 1. Simulation of a physical process by a closed function f . (a) According to the Mogi model, a small inflating magma source M (described by a 4-D variable $\mathbf{x}$ (three components defining coordinates and one component defining an inflation parameter)) produces ground deformation and displacements $\mathbf{d} = (d_x, d_y, d_z)^T$ at a ground point P in the influence of the source M (i.e. at distances r smaller than a certain threshold different for each source M ; these displacements (deterministic, errorless values) can be computed (predicted) using closed functions f . This represents a forward problem. The inverse problem is to use measurements ℓ (estimates with uncertainties, errors) of displacements $\mathbf{d}$ (for example GPS measurements) and solve for $\mathbf{x}$. The solution (inversion) $\hat{\mathbf{x}}$ is characterized by uncertainties. Modified after (Saltogian & Stiros 2013a). (b), (c) Influence zone (grey circle, areas of expected ground deformation) of a magma source M (solid black dot), showing observations sites (red triangles), corresponding to P in (a) in horizontal projection. In (b) the configuration of the observations system is optimal (full coverage of the influence zone in two subvertical directions), leading to accurate modeling of the source M (open circle, nearly overlapping its 'true' position). In (c) the configuration is poor, leading to multiple and biased estimates of M (open circles).

points P in Fig. 1), a system of observation equations (SEO) is formed, and its solution is expected to define the unknown variable $\mathbf{x}$ which describes the 'parameters' of the physical process (magma source in this example).

However, even for this apparently simple case, the solution of the SEO is not simple. This is because such systems are redundant or underdetermined, equations are usually non-linear and no approximate solutions are known. In addition, measurements, although often regarded as errorless constants in observations eqs (Yang 2010; Gao *et al.* 2016), are contaminated by unknown measurement errors $\mathbf{v} = (v_1, v_2, \dots, v_m)^T$ and in fact represent random (stochastic) variables $\ell = (\ell_1, \ell_2, \dots, \ell_m)^T$ for which an estimate only is available through measurements (Mikhail 1976, p. 8).

For this reason, eq. (1) is written:

$$f(\mathbf{x}) = \ell + \mathbf{v} \quad (2)$$

and the solution $\hat{\mathbf{x}}$ of these systems of equations can only be derived from numerical search techniques, known as *inversion* or *optimization* (Sen & Stoffa 1995; Sambridge & Mosegaard 2002; Menke 2012).

The simplest and most traditional method of inversion is to define a $\mathbb{R}^n$ space which is expected to contain the solution $\hat{\mathbf{x}}$ of the SEO, discretize this search space by a regular n -dimensional grid (hypergrid if $n > 3$) and analyse each of the gridpoints (exhaustive search). For each of the gridpoints (nodes) $\mathbf{x}$, eq. (1) predicts a vector $\mathbf{d}$. The difference between measurements ℓ and model predictions $\mathbf{d}$, known as misfits are then computed. Mean misfits for each point (see below, Section 2.2) are then used to identify a gridpoint which better fits the observations and is recognized as the solution of the SOE (Panza 1981; Shibutani *et al.* 1996; Feng & Newman 2009; Menke 2012).

In a 2-D search space, the value of mean misfit (cost function) is regarded as a third dimension, with the relief of the mean misfit taking the form of a topographic surface approximated by contours (Fig. 2). Sometimes misfit contours indicate a prominent hill, that is a main ('deep') 'global' extremum, so that a 'global' solution is

inferred (Fig. 2c), but sometimes such a hill is not well-constrained (Fig. 2b). In highly non-linear functions, the misfit surface may be complicated, comprising numerous, non-dominant ('shallow'), local extrema (Fig. 2d, Tarantola & Valette 1982; Cervelli *et al.* 2001; Sambridge & Mosegaard 2002). Depending on the grid density, exhaustive searches can identify not only 'deep' but also 'shallow' extrema, the physical significance of which remains to be evaluated.

Exhaustive searches are regarded as efficient techniques for inversion (Mosegaard & Tarantola 1995), but are subject to two major limitations. First, they are confined to problems with few unknown variables, $n \leq 4$ (Feng & Newman 2009; Menke 2012), but usually to $n = 2$ or 3, so that they can be directly visualized in a 3-D space, either as contours, or as chromatic palettes. In some cases this process is adopted after *a priori* fixing values to one or more of the unknown variables. Another limitation of exhaustive searches is that they require excessive computation time because of forward calculations for each gridpoint. For these reasons, inversion is usually based on selective searches using various numerical, mostly stochastic search techniques, to one degree or another based on Monte Carlo simulations. A main advantage of these techniques is that they permit rapid convergence to a solution, but usually they focus on a single, main, 'global' extremum of the misfit surface. Hence 'shallow' extrema which may be of physical importance remain unnoticed, or the solution is trapped into local extrema (Fig. 2c). This is one of the reasons why the variable slip models of the M_w 7.4, 1999 Izmit (Turkey) earthquake are so different (see Beresnev (2003), Fig. 3), despite the overall simplicity of the fault and the magnitude of the earthquake (Barka *et al.* 2002).

Recently, a new exhaustive technique broke the limit of very few unknown variables which characterizes exhaustive searches and was used for inversion of at least 18 unknown variables (Saltogian & Stiros 2013a,b, 2015a). This new technique was inspired by the principle of beacon (lighthouse) positioning and of positioning using Wi-Fi or GSM beacons, in fact of compound geometric loci, and for this reason it was named TOPological INVersion (TOPINV) algorithm. The novelty of this algorithm is that it is not based on

Figure 2. Problems of estimation of global peaks. (a), (b) Coloured contours of the spatial variability of a moment tensor solution of two M_w 6 earthquakes which hit Cephalonia Island, Greece, in 2014. In (a) a well-defined ‘hill’ permits to define well the focal mechanism of the first shock. In (b) the ‘hill’ is not well constrained (only one contour closed). (c) A dominant extremum in 3-D and contour form indicating a clear solution. (d) A complicated misfit surface with numerous local and no dominant extremum. A relatively higher extremum is usually taken as global extremum and a solution, but things are more complicated, especially since each extremum is contaminated by errors and its statistical difference from other extrema is not clear. (a, b) After (Sokos *et al.* 2015). (c, d) After (Sambridge & Mosegaard 2002).

minima in mean misfit surfaces, but on identification of ‘successful’ gridpoints for which the difference between predictions and measurement for each observation equation are within parametric uncertainty intervals. Still, its application was hampered by the requirement of excessive computational time.

In this article we show that this main limitation of the new algorithm can be to a major degree overpassed exploiting the potential of graphics processing units (GPU) cards which are found or can be added at low cost in modern personal computers; that is using low-cost parallel computing. In fact, we show that GPU processing using a suitable programming language [in our case CUDA, a variation of C/C++ (NVidia 2019)] leads to reduction of computing time by a factor of up to ≈ 100 , a real revolution in inversion. This means that a processing time of 5 hr is suppressed to three minutes and this impressive speedup in calculations makes multivariable exhaustive search inversion practical and suitable for nearly real-time solutions required for many applications, for example for certain early warning systems (*cf.* Wright *et al.* 2012).

Since the grid density is a major factor controlling computational load, a second task of this article is to exploit the high speed offered by GPU processing in order to investigate the impact of the density of a regular grid in the accuracy (difference of the computed solution from the ‘true’ solution) and the precision (spread of various solutions around the computed solution) of computed solutions.

This is a problem of paramount importance to minimize the latency of results, for example to arrive at a reliable finite fault model of a seismic fault and hence evaluate its potential for tsunami generation in near coastal regions.

Both tasks were based on synthetic data, so that the output of the algorithm can be unambiguously compared with reference (objective, ‘true’) values, and on two common geophysical cases of inversion in spaces with $4 \leq n \leq 18$: modeling magma chambers of active volcanoes and of the finite seismic faults using GPS observations of ground deformation (*cf.* Cervelli *et al.* 2001; Atzori & Salvi 2014). For both tasks, results are very encouraging for the application of this algorithm in specific demanding cases of inversion.

2 INVERSION TECHNIQUES

Currently, a wide range of inversion techniques are used in Geophysics; for a review see Sen & Stoffa (1995), Sambridge & Mosegaard (2002), Oldenburg & Li (2005) and Menke (2012). Inversion, typically a search-based solution of a system of equations, has three requirements: a search space, a search criterion and a search (sampling) method (or algorithm).

Figure 3. Different variable slip models of the M_w 7.4 Izmit (Turkey) strike slip earthquake using seismological data. Although the magnitude of the earthquake permitted a favorable radiation pattern and the seismic fault was well constrained by strike slip surface faulting with multimetric displacements (Barka *et al.* 2002), each of the five solutions points to a strikingly different fault model. After Beresnev (2003). The ambiguity in the solution of this high dimensionality inversion problem (indicated by vectors in d) is partly due to non-uniqueness of the inversion.

2.1 Search space

The search space (or model space) corresponds to a n -dimensional space in which the solution is likely to exist. A search space is defined in advance, using certain additional, independent information (for example, in the case of the problem of Figs 2(a) and (b) the distribution of foci of micro-earthquakes), or even assuming

reasonable, large uncertainty margins for each components of the unknown variable x .

The dimension of the search space, defined by the number of unknown variables, may range from $n = 2$ ($n = 4$ in the example of the Mogi source of Fig. 1) to a few hundreds, or even more; for example, in the case of inversion of a fault of variable slip as that of Fig. 3.

In certain cases, the dimensions of the search space are not regarded constant but are adapted to the existing conditions [‘trans-dimensional inference’; (Green 1995; Sambridge *et al.* 2013)].

In the case of regular or exhaustive grid searches, all parts of the search space, represented by a vector $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$ are covered (sampled). This means that each of the $i = 1, 2, \dots, n$ dimensions of the search space is discretized, replaced (sampled) by $m_1, m_2, \dots, m_n$ discrete points (nodes), respectively, so that the overall search space is approximated by:

$$p = m_1 \cdot m_2 \cdot \dots \cdot m_n \quad (3)$$

discrete grid (hypergrid) points in total. The sampling density of the n -dimensional grid is critical, because in highly non-linear functions certain effects may be missed and hence solutions may be biased (Tarantola & Valette 1982). On the other hand, increasing the density of a grid may prove impractical from the computational point of view. For example, in the case of the modeling of seismic faulting consisting of two segments (18 unknown variables), assuming $m_1 = m_2 = \dots = m_{18} = 5$, according to eq. (3), $p = 5^{18} \approx 10^{12}$ gridpoints. The time required to calculate the solution of this system is of the order of one month for conventional computations with TOPINV. On the contrary, for selective, repeated searches in grid spaces (e.g. Sambridge & Gallagher 1993), the corresponding time is reasonable because sampling is limited. For example, (Shibutani *et al.* 1996) repeated 200 times a sampling of 50 points using a directed genetic algorithm (GA) so that 10 000 models were examined among the nodes of a regular grid with 10^{25} points.

2.2 Search criterion

The search or selection criterion, usually a cost function (or objective function), is a mathematical or statistical variable which permits to evaluate from a quantitative point of view how well a n -dimensional point from the search space fits the available observations. Usually, this criterion is the mean weighted misfit $u^T w u$. This last variable represents the sum of weighted squares of differences u_{ij} between each measurement ℓ_{ij} and the corresponding model prediction $f(x_i)$ for a tested point x_i , following eq. (4)

$$u_i = f(x_i) - \ell_i \quad (4)$$

$w_i = (w_{i1}, w_{i2}, \dots, w_{im})^T$ indicates the weights of observations and its components are equal to the inverse square of standard deviations σ_{ij} of the corresponding observations $\ell_{ij}, j = 1, 2, \dots, m$ (see Tiampo *et al.* 2000; Cervelli *et al.* 2001; Feng & Newman 2009; Menke 2012). Misfits u_i are obviously different from v_i in eq. (2) and are contaminated by errors, as implied by the law of error propagation (see Mikhail 1976); this is because they are functions of ℓ_i . If ℓ_i are assumed errorless (Yang 2010; Gao *et al.* 2016), the solution process is technically simplified, but at the expense of some bias or uncertainty in the solution.

In most cases, mean misfit is usually accepted as the selection criterion. However, in some cases other more elaborate techniques, such as probability density functions are adopted (Sambridge & Mosegaard 2002; Menke 2012).

2.3 Search method

The search method defines how, and which parts of the search space are to be sampled and what will be the output of the search, usually a single n -dimensional point representing a global peak in the misfit surface and a single optimal solution, or more rarely a set of

Figure 4. Symbolic representation of various inversion techniques, simplified after (Sambridge & Mosegaard 2002). Exploration indicates the tendency to cover as much as possible the search space, exploitation to identify the prominent parts of the search space in which the solution is located; arrows indicating their increase. Uniform search and gradient based techniques represent the two edge-point approaches, and in between are usually techniques based on Monte Carlo sampling. SA and GA indicate simulated annealing and genetic algorithms.

peak points (e.g. Lomax & Snieder 1995). There are two endpoint techniques to search (sample) the search space. On one hand, the search space is discretized and replaced (sampled) by a regular grid, and the search explores all parts of the search space (exhaustive search). As noticed above, this approach is limited to very few unknown variables. Secondly, search is directed to a promising part of the search space, using gradient-based or other techniques (Sambridge & Mosegaard 2002). As is schematically shown in Fig. 4, between these two endpoint approaches, there are included various, popular approaches for selective, random or quasi-random repeated sampling. Among them, of major importance for Geophysics are techniques based on genetic algorithms (GA) and simulated annealing (SA) which combine random and directed searches.

The basic procedure is schematically summarized in Fig. 5. At a first iteration the $\mathbb{R}^n$ search space is sampled and projected into a $\mathbb{R}^2$ space defined by two master coordinates (usually Easting and Northing). Using a cost function (usually mean misfit) as a third dimension, a promising $\mathbb{R}^2$ space (characterized by reduced misfits and hence expected to contain the solution) is identified. This promising part of the $\mathbb{R}^2$ space is then back-projected into the $\mathbb{R}^n$ space and the whole process is repeated until convergence to a solution is obtained.

Sampling of the $\mathbb{R}^n$ space is based on various versions of the Monte Carlo technique, using either random, or more frequently quasi-random or directed searches. In some cases, a combination of discretized search spaces and of random or quasi-random searches is adopted (Sambridge & Gallagher 1993). In all cases, however, care is taken so that in each round/iteration of the algorithm, a relatively limited number of $\mathbb{R}^n$ points are sampled so that calculations are rapid. Small size sampling has another advantage: it excludes the possibility two different points of the n -dimensional search space to share the same 2-D projection (Fig. 5; cf. below Fig. 7c) which is the basis of the selection of the solution.

Another class of algorithms searches not for isolated point solutions, but on ensembles of $\mathbb{R}^n$ points with misfits above a certain level (‘acceptable solutions’) which may also identify more than

Figure 5. Principle of identification of the minimum misfit solution through iterations. Left-hand panel: During Iteration 1, sampled points from a $\mathbb{R}^n$ search space are projected onto a $\mathbb{R}^2$ space defined by two selected master coordinates (usually Easting and Northing). In this plane, the contours of misfits are expected to define a ‘promising area’ for the search of the solution. From this $\mathbb{R}^2$ space is then the corresponding $\mathbb{R}^n$ space identified (subspace of the initial search space, around the promising n -dimensional points of the first iteration). Right-hand panel: In the second iteration, the promising $\mathbb{R}^n$ space of the first iteration is sampled and a new promising space is identified. The process is repeated until conversion to an acceptable solution.

one extrema/solutions (Lomax & Snieder 1995). Some of these approaches are based on Markov Chain Monte Carlo (MCMC) sampling (Gallagher *et al.* 2009; Sambridge *et al.* 2013).

2.4 Weaknesses of observations systems and possibility of multiple solutions

A major problem faced in inversion is that in some cases the misfit surface has the form of a pronounced hill (or inverse hill, Fig. 2c), while other times it is complicated, containing numerous small but no prominent extrema (Fig. 2d); a situation assigned to high non-linearities of the functions describing a geophysical problem (e.g. Curtis & Lomax 2001). In the first case, most inversion algorithms rapidly converge to an unambiguous solution, while in the second case the situation is complicated and the search of a global solution may lead to identification of the (apparently) highest peak. The physical significance of the latter is not clear, because the significance of multiple peaks is a problem not well understood (*cf.* Sambridge & Mosegaard 2002). Cases of residuals such as those of Fig. 2(d) are usually assigned to low signal-to-noise level in observations (Cervelli *et al.* 2001), which tend to amplify complications in misfit surfaces. Recently, it was found that in the case of very low SNR inversion algorithms tend to biased solutions (Hu *et al.* 2019).

There is evidence that the situation is more complicated. This evidence comes from two sources. First, from adjustment of geodetic networks, that is least squares solution of linearized observations with strict geometric constraints and a priori knowledge of approximate values of unknown variables (see Mikhail 1976). Second from experience with analysis of earthquake hypocentre location (Billings *et al.* 1994). Based on this evidence, the quality of a solution depends on several factors: (i) On the SNR, reflecting both a mean value of SNR of all observations, but also on specific, high-error or outlier-type observations which cannot be *a priori* identified. (ii) On the redundancy ($m - n$) of observations, that is the difference between the number m of observations equations and the number n of unknowns in the system of observation equations,

covering also underdetermined systems [$m - n < 0$; see Saltogian *et al.* (2014)]. (iii) On the configuration (a kind of geometry) of observation equations (Mikhail 1976; Billings *et al.* 1994).

This last point requires some additional concern. For example, in Fig. 1(b) observations stations control well the area of influence of the magma source in two subvertical directions and permit good control of all four parameters of the magma source; that is a geometry favorable for the accurate inversion. In Fig. 1(c), on the contrary, the same number of stations cover a part only of the influence zone and they are expected to lead to bias in the magma source coordinates, and possibly to more than one solution. A similar situation was demonstrated for the elastic dislocation of a seismic fault [fig. 9 in Saltogian *et al.* (2014)].

In summary, it is not justified to *a priori* regard a relatively higher misfit as a global extremum and as a solution of the SOE, especially without any statistical analysis justifying the statistical significance of its difference from smaller peaks. In addition, each of the ‘shallow’ extrema may also correspond to a ‘true’ spurious solution. This implies a need for identification and evaluation of *all* potential solutions and select the one or those with physical significance.

3 THE TOPOLOGICAL INVERSION ALGORITHM

3.1 Overview

Based on an exhaustive search, TOPINV identifies and records in matrix form the gridpoints for which *all* misfits are within uncertainty limits defined by a parameter k (‘successful’ gridpoints). The process is repeated for different values of k until the successful gridpoints define clusters of successful points which satisfy certain conditions and hence they lead to one or more statistical solutions of the SOE. For this reason, this algorithm is valuable in cases of inversion characterized by multiple shallow extrema and no dominant global extrema.

The new algorithm can be regarded as a generalization of approaches which do not seek a single minimum mean misfit solution, but they examine an ensemble of solutions above a certain threshold (Lomax & Snieder 1995; Gallagher *et al.* 2009).

3.2 Documentation

Let us assume that functions f describe well a physical process, that the vector of observations ℓ contain only small unknown errors $v = (v_1, v_2, \dots, v_m)^T$ with standard error $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_m)^T$, so that the absolute values of errors in each measurement and their variance are upper bounded variables (it will be shown below how outliers can be noticed and excluded), and that vector $\hat{x}$ is a solution of the system of eq. (2). Then,

$$|f(\hat{x}) - \ell| = |v| < \varepsilon, \quad (5)$$

where $\varepsilon = (\varepsilon_1, \varepsilon_2, \dots, \varepsilon_m)^T$, $\varepsilon_1 \approx \varepsilon_2 \approx \dots \approx \varepsilon_m = \varepsilon$ a small positive value. Eq. (5) is obviously a generalization of equation

$$|f(\hat{x}) - \ell| = 0 \quad (6)$$

which is valid only in the case of errorless measurements ℓ . Because $|v|$ are bounded, it is possible to introduce a variable k so that

$$|f(\hat{x}) - \ell| < k \cdot \sigma < \varepsilon \quad (7)$$

which corresponds to a typical uncertainty interval:

$$|f(\hat{x}) - \ell| < k \cdot \sigma \quad (8)$$

A first implication of eq. (8) is that if an *a priori* value for parameter k is adopted and a node of a regular grid (thereafter grid G) is tested, the corresponding eq. (8) (system of inequalities) contain no unknown elements, and the output can only be TRUE (if the node is at the vicinity of the solution) or FALSE (in all other cases); this is represented by a variable $b = 1$, $b = 0$. This ON/OFF, Boolean output corresponds to an answer to the question ‘does this specific gridpoint satisfy all the observations equations for the adopted uncertainty limit k ’, that is $b = 1$, or not ($b = 0$)? The process is repeated for all p gridpoints, and the results can be summarized in matrix form:

$$S(k) = \begin{bmatrix} x_{1j_1}, x_{2j_2}, x_{3j_3}, \dots, x_{n_{j_n}}, b \end{bmatrix} \\ j_1 = 1, 2, \dots, m_1; j_2 = 1, 2, \dots, m_2; j_n = 1, 2, \dots, m_n. \quad (9)$$

Matrix $S(k)$ covers all p nodes sampling the whole search space (see eq. 3). From $S(k)$ there can be produced a matrix $S_q(k)$ with q rows and n columns containing the q ‘successful’ gridpoints (i.e. those with $b = 1$) corresponding to the specific value of k . Depending on the value of k , q may contain no successful points at all ($q = 0$), a few points or too many points, or even nearly all or all gridpoints ($q \approx p$ or $q = p$) for large values of k (see Fig. 6). The different values of k define topologies around the ‘true’ solution(s).

In order that the q successful gridpoints described by matrix $S_q(k)$ correspond to one or more solutions, they must satisfy two criteria.

First, each cluster must contain a limited number of gridpoints, permitting a statistical solution. Experience with various inversion problems indicates that the number of points for a valid solution usually ranges between a few or a few tens to a few hundred (empirical statistical criterion).

Secondly, each n -dimensional cluster of points should correspond to a small, closed, compact space. Practically, this topological criterion means that the successful gridpoints should be distributed in a few adjacent nodes of the grid (see Fig. 7a), that any of their 2-D plots does not correspond to a shape with ‘inlets’ (cf. k_3 in

Fig. 6) and there are no voids which also tend to indicate multiple combined solutions (topological criterion).

In the example of Fig. 7(b) the value of k is too large, while Fig. 7(c) indicates two clusters which may indicate two different solutions, differing in one or a few dimensions only [slip in this case; see also fig. 4 in Saltogian *et al.* (2017)].

In order to represent a solution, a cluster of points should satisfy the above two criteria. The solution can then be computed using first and second statistical moments (mean values and variance-covariance matrices of the q gridpoints).

3.3 Implementation strategy

Inversion using TOPINV is made in five steps.

First, the search grid G and the system of parametric observation inequalities (eq. (8)) are defined.

Secondly, the set of values of k to be tested is defined; then for each k , an exhaustive search is made, and the output is stored in a matrix $S_q(k)$ of successful gridpoints.

Thirdly, each cluster of successful gridpoints is tested whether it satisfies the statistical and topological criterion.

Fourthly, for each cluster of gridpoints satisfying the two criteria, a statistical solution is computed. The critical points are the selection of values of k and the test for the topological criterion. These two points are explained below, while the overall process is summarized in Algorithm 1.

Fifthly, computed solution(s) are validated. This is made in three ways. (1) Through comparison of solutions with independent evidence (i.e. through comparison with ‘reference’ or ‘true’ values in the case of synthetic data, or using independent geophysical evidence); (2) Through examination of the solution in comparison with the distribution of points in the search space; for example a solution at the margins of the grid space may indicate a need to expand the limits of the search space (cf. Fig. 2b). (3) Through examination of the precision (standard deviation) of the estimates of each unknown variable and of the values of the mean weighted misfits as a function of the value of k . For example, null standard deviation combined with high mean misfit are indicative of a poor (or biased) solution constrained by a single gridpoint near the ‘true’ solution in a very coarse grid and hence a need for densification of the sampling of the critical dimension. Hence mean misfit is regarded not as a selection criterion, but as one of the validation criteria.

3.3.1 Selection of values of k

An initial guess of two values of k roughly corresponding to values k_2 and k_7 of Fig. 6 is made based on previous experience and trial-and-error. Then a value of $k_{\min}$ and $k_{\max}$ and a step for interpolation of values of k are adopted. This permits to define a set of interpolated values $k_1, k_2, k_3, \dots, k_{\min} < k_1 < k_2 < k_3 < \dots < k_{\max}$. An exhaustive search is made for each value of k and for each gridpoint, leading to a matrix $S_q(k)$ of successful gridpoints.

3.3.2 Topological criterion

In order to investigate whether the “successful” points of a cluster correspond to a compact, closed space, these points are projected in all possible combinations in a set of 2-D spaces, as in Figs 6 and 7(a) (cf. Cervelli *et al.* 2001). The investigation for closed, small spaces (i.e. with population of gridpoints for each cluster below a certain threshold) can be made visually (Fig. 7), using a

Figure 6. The concept of identification of solutions in the system of observations equations (SOE) through uncertainty intervals defined by parameter k . Left-hand panel: plot of q successful gridpoints (i.e. those satisfying all observations equations within specific uncertainty intervals) in one component (dimension) of the search space for various values of k . Right-hand panel: corresponding 2-D plots of successful gridpoints as a function of k [i.e. of a matrix of points S_q]. The geometries of plots guide the selection of k to identify clusters of gridpoints from which the solutions will be computed. A solution corresponding to a major peak (a deep extremum that can be identified by most inversion algorithms) as well as two additional solutions corresponding to two smaller peaks (shallow extrema) are indicated in this example. For simplicity, the graph indicates solutions, but in fact it indicates clusters of gridpoints which need to be tested whether they satisfy certain criteria to represent solutions.

common clustering algorithm (e.g. Tasoulis & Vrahatis 2006), or using formal pattern recognition techniques, classifying results in a few types of distributions, as those in Fig. 6.

3.4 Specific characteristics, advantages and limitations

3.4.1 Multiple, minimum variance and stable solutions

The function of the new exhaustive search algorithm is to identify all clusters (ensembles, sets) of gridpoints with misfits in all observation equations within certain parametric uncertainty levels (eq. 8), in analogy to Gaussian statistics. The optimal value of k is then identified, and the corresponding clusters of gridpoints solutions are used to compute solutions based on first and second statistical moments. These solutions correspond to minimum variance solutions, in analogy to least squares solutions and functions of k , and are insensitive to noise which characterizes misfits, leading to uncertainties in the recognition of differences in values between two mean misfits, as the Law of Error Propagation implies (cf. Mikhail 1976).

As will be explained in Section 7 below, these minimum variance solutions are different from the minimum mean misfit solutions, but they are in the vicinity of some of them, and the clusters of successful gridpoints of the new algorithm include points with mean minimum misfits [see also fig. 13 in Saltogianni & Stiros (2013b); fig. 4 in Saltogianni et al. (2017)].

As is shown in Fig. 6, for example k_3 defines a cluster from which a solution is inferred. A slight change of the value of k is expected to lead to a different solution. Still, these different solutions tend to a stable solution which represents a necessary condition for an optimal solution (cf. Tarantola & Valette 1982). This is because given the

constraints for k , all estimations of solutions differ slightly from each other (no statistical difference derived from their variances). This point is discussed below, in Section 7.

Solution are clearly influenced by the selection of standard deviations of observations, in analogy to typical least squares-based solutions (Mikhail 1976). Furthermore, eq. (8) implies that what plays a major role in the solution is not the values of standard deviations which may be somewhat uncertain, but their ratios. Ratios of standard deviations are indeed defined with much higher accuracy than standard deviations themselves, as for example in the case of GPS (GNSS) measurements: Standard deviations of GPS (GNSS)-derived coordinates and displacements depend on instrumentation, local conditions, etc., but with few exceptions, the uncertainty in vertical displacements is approximately twice as large as that in Easting and Northing. This highly reduces the deterministic character of weighting.

3.4.2 Identification of constraining observations/outliers

Since the algorithm tests all observations equations and all gridpoints, it can identify which observations are satisfied by low levels of k and which only by very high, and in fact constrain the solution. Such ‘constraining observations’ may in fact represent observations with increased noise, even outliers, in analogy to *a posteriori* estimates of large errors in *Least Squares* adjustments (Mikhail 1976), or alternatively, they may indicate that the adopted mathematical model (eq. 1) does not describe well the physical model.

This analysis is, however, not possible in speedup CPU versions of the algorithm. In these cases, if a misfit in an equation is found not to fall within the uncertainty limits defined by eq. (8), the algorithm

Figure 7. Testing of the statistical and topological characteristics of successful gridpoints described by matrix $S_q(k)$ through 2-D diagrams. (a) Corresponds to all possible combination of 2-D plots of nine variables defining a seismic fault. Black dots indicate covered nodes, not frequency of corresponding gridpoints, and for this reason their mean value (red dot) may not correspond to the geometric mean of the cluster. (b) Successful points in this plot cover most of the grid, and do not represent a solution; the algorithm should be repeated with smaller values of parameter k . (c) Successful points in this plot define two distinct clusters/closed spaces, hence they define two solutions within the search space. These two solutions may appear only in a single diagram, hence the two solutions may differ only in one variable.

ignores all other equations, assigns $b = 0$ to this gridpoint (eq. 9), and moves to the next one.

3.5 Diagnostic criteria

In Section 3.2 it was explained that a solution should satisfy two specific, statistical and topological criteria. Three diagnostic criteria,

partly derived from experience with similar problems, should be satisfied for high quality solutions in real-world applications.

(i) That the standard deviation s_i of each variable satisfies the condition $0 < s_i < s_{max}$, or better $s_0 < s_i < s_{max}$, $s_0 \rightarrow 0$, where s_{max} and s_0 represent empirical maximum and minimum values, adopted to avoid error artefacts or that a solution is trapped by a single gridpoint (see Supplement S1).

Algorithm 1 Synoptic form of the TOPINV Algorithm.

```

1: START:
2: Input:  $n$ -dimensional search space & sampling frequency per
   dimension
3: Input:  $m$  displacement measurements and uncertainties
4: Input: Preliminary values of  $k_{min}$  and  $k_{max}$ 
5:
6: Form search grid  $G$  with  $p$  grid-points (nodes)
7: Form  $m$  observations equations
8:
9: Call INVERSION( $k_{min}$ )
10: Call INVERSION( $k_{max}$ )
11: if  $k_{min}$  or  $k_{max}$  not acceptable then
12:   Go to START
13: end if
14:
15: Define  $k_1, k_2, k_3, \dots$ 
16: for  $i \leftarrow 1, 2, 3, \dots$  do
17:   Call INVERSION( $k_i$ )
18:   Call PLOT-TEST()
19: end for
20:
21: procedure INVERSION( $k$ )
22:   Compute model predictions using forward calculations,
   for each of the  $p$  grid-points and each of the  $m$  observations
   equations; Boolean test whether model prediction is
   within uncertainty interval (point defining a potential
   solution); For parallel computation see Algorithm~2
23:   Form matrix of  $q$  successful grid-points satisfying all  $m$ 
   uncertainty intervals
24: end procedure
25:
26: procedure PLOT-TEST()
27:   Plot  $q$  points in 2-D plots
28:   Separate clusters
29:   if clusters satisfy conditions then
30:     Call SOLVE()
31:   end if
32: end procedure
33:
34: procedure SOLVE()
35:   Compute solution, uncertainty, mean misfit
36: end procedure

```

(ii) That the mean misfit is within acceptable limits, empirically derived.

(iii) That the value of optimal k is within specific limits, empirically derived.

Criterion (1) in particular excludes the possibility of a solution trapped into a single node ($s_i \approx 0$) of a very coarse grid ($s_i > s_{max}$). Possible remedies are higher values of k (see Table S2), denser grids, perhaps only for the variables with the problem so that the additional computational cost is limited (*cf.* Fig. 11) and cleaning for outliers (see above, Paragraph 3.4). Additional remedies include adaptation of alternative physical models (*cf.* Feng & Newman 2009) or even modified search spaces.

3.6 Rapid, heuristic solutions

A formal solution as that described above is especially suitable for an in-depth analysis of geophysical effects/processes, while a rapid solution, not necessarily optimal, may be useful for cases in which a rapid result is necessary. Such rapid ('heuristic') solutions can be obtained simply imposing empirical constraints to k , usually $k < 5 - 7$, and to p , usually $10^1 < p < 10^2$ (see Supplement S2). Clearly, such heuristic solutions cannot identify multiple solutions and can only be regarded as preliminary solutions.

3.7 Applications and validation of the algorithm

3.7.1 Applications

The algorithm has been used so far for modelling of magma sources (problems in $\mathbb{R}^4$ and $\mathbb{R}^{14}$ spaces (for example Saltogianini *et al.* 2014), for modeling of finite seismic faults using geodetic data (problems in $\mathbb{R}^9$ and $\mathbb{R}^{18}$ spaces (for example Saltogianini & Stiros 2015a; Saltogianini *et al.* 2017) and for the solution of certain geodetic problems (Harvey 2013). Such applications have led to some experience concerning the ranges of values for the parameters discussed above.

3.7.2 Validation

TOPINV has been validated so far in three ways. First, through analysis of synthetic data, so that the 'true' value of the unknown variables is statistically not different from the output of the algorithm (Saltogianini & Stiros 2013a,b). Secondly, through inversion of the same problem using TOPINV and a conventional geodetic network adjustment algorithm (Saltogianini & Stiros 2013b). Third, through assessment of the results of the geodetic inversion using seismological evidence (e.g. Saltogianini *et al.* 2017).

4 METHODOLOGY

As noticed in the Introduction, the aim/task of this article is twofold. To document the speedup offered by GPU parallel processing and to investigate the influence of grid density (i.e. sampling frequency) in the accuracy of the estimated solutions. Both these tasks are based on analysis of realistic, synthetic data permitting full-control of the inversion output (Cervelli *et al.* 2001; Saltogianini & Stiros 2013a).

To fulfill the aims of this paper, two programs have been developed. The first is a program written in FORTRAN and executed on a conventional CPU. The second program is written in CUDA and is executed on a GPU. Both programs perform the same task, that is, they compute model predictions and determine the gridpoints that satisfy eq. (8).

4.1 Speedup using GPU processing

For the first task we compared the time required for inversions using the GPU/CUDA implementation against the time required for the same inversions using the CPU/FORTRAN implementation. To make things representative of a wider class of cases of inversion, we examined two common problems in Geophysics, inversion of GPS data to model inflating magmatic Mogi sources [Mogi model (Mogi 1958)], and finite fault models reactivated during an earthquake [elastic dislocation analysis (Okada 1985), see Battaglia *et al.* (2013)]. Both problems were examined using simpler and more

complicated sources of deformation (single or double), described by $n = 4$ to $n = 18$ unknown variables.

We adopted realistic sources of deformation (magma sources or seismic fault models) and using forward modelling, we computed the predicted exact, error-free displacements for a set of ground points, representative of GPS observations stations. This process is schematically shown in Fig. 1 in B. To make data realistic, we added some noise to the predicted displacements in order to produce synthetic observations of ground displacement.

For each of these physical problems, we selected grids with different density to formulate different inversion scenarios, sensitive to two main parameters influencing time required for inversion, the number of unknown variables and the number of gridpoints.

Each inversion scenario was solved using two different processing schemes (CPU/FORTRAN and GPU/CUDA) and there was recorded the time required for each scheme as well as the ratio reflecting speedup. This analysis is summarized as ‘Task1’ in flowchart form in Fig. 8. This task obviously requires a transformation of the existing FORTRAN code into code suitable for GPU processing, exploiting its capacities for the solution of geophysical problems.

4.2 Sampling effect

For the second task, we investigated the influence of the density of the grid in the accuracy of estimated solutions using a common elastic dislocation problem. A representative deformation source (finite model of an Okada-type seismic fault) was selected, and the dislocations predicted at a set of GPS stations around it were computed using forward calculations. The inverse problem was to use these predicted (synthetic) observations to compute the source of deformation for different grids. This overall approach is a generalization of the approach of (Cervelli *et al.* 2001) and (Saltogian & Stiros 2013a) and permits to compare estimated variables and their variances with the true (reference) values.

5 OPTIMIZED GPU PROCESSING

5.1 Basic characteristics of GPU processing

A possible path to overcome the long computational time of TOPINV is the exploitation of parallel computing. Supercomputers consist of large numbers of processors, connected through specialized, high-speed networks. Using appropriate parallel programming models, programs that distribute the computational load among processors can be developed and the overall execution time can be drastically reduced. Such systems, however, are not readily accessible to all researchers, due to their high cost, access policies and large numbers of applications that request simultaneously resources for their execution.

Recently, graphics processing units (GPUs) have evolved into computing engines for general purpose programming. Similar to modern CPUs, they consist of computational cores, which are the units that perform calculations at the hardware level. They include, however, a much larger number of cores (up to a few thousands currently), which provide extreme computational power at an affordable cost and with very low energy consumption, keeping running costs low. For these reasons, GPU-based personal computers are currently exploited for performing computations in many scientific fields (Manavski 2007; Manavski & Valle 2008; Tölke & Krafczyk

2008; Nobile *et al.* 2016), providing significant speedups in execution time, whereas GPUs have been included as computational engines even in supercomputers (TOP 500 2019).

In the context of this paper, the CUDA programming model (NVidia 2019) has been used to develop TOPINV for execution on a GPU. CUDA is an extension of the C/C++ programming languages and enables an application to exploit the large number of available cores of a GPU by executing a sequence of instructions on each core. The entities of the application that represent the sequences of instructions that each core has to execute are called ‘threads’ and are defined by the programmer during development of the application. Programmers are in charge of defining the calculations that each thread has to perform, whereas CUDA is then responsible to decide on which core each thread will execute. In order to keep the flow of the discussion on the Geophysical aspects of this paper, we focus on the details about the CUDA programming model that are relevant to the implementation of TOPINV in A1.

5.2 Optimized GPU implementation for inversion

As is explained above and in Algorithm 1, the basic function of TOPINV is to select an uncertainty level k and then test whether each of the p gridpoints satisfies all m observation equations, described by eq. (8). This set of m computations represents a *thread*, and for each value of k there must be executed p threads. In the case of a simple CPU this may be an extremely time-consuming process if calculations are sequential and p is large ($p > 10^9$), but the execution time is drastically reduced if calculations are made in parallel.

The GPU/CUDA implementation permits to drastically minimize the execution time through distribution of the p threads into ‘groups’ of calculations (*blocks of threads* and *grid of blocks*, see Fig. A1) which are performed at the same time (in parallel) so that each of the available cores of the GPU receives the same load and no cores are overloaded or remain idle, to the extend this is possible.

The time-consuming part in this procedure is the computation of model predictions for each of the p gridpoints and for each observation station (Algorithm 1, Line 22). During each successive iteration, the GPU/CUDA implementation distributes calculations to threads so that calculations are made in parallel and the execution time is drastically reduced (Section 6.3).

For problems with $n = 1$ to 3 dimensions this is a straightforward task because CUDA is designed for problems with 1 to 3 dimensions, and threads are readily assigned to the GPU cores using the default CUDA grid geometry (Fig. A1). Geophysical problems on the contrary, have n dimensions, and in addition the nodes in each dimension of Grid G are different (see eq. 4), that is they are not compatible with the CUDA default implementation architecture. Hence the programming problem we had to solve was how to distribute the groups of threads into groups of calculations (or how to project/translate the n -dimensional Grid G into the CUDA computation scheme), so that the execution time is minimized.

The solution of this problem is based on an algorithm [Algorithm 2 in Section A2 of A] which searches the distribution of gridpoints of Grid G in the n dimensions (variables $m_1, m_2, \dots, m_n$ in eq. (4)) in order to group them in CUDA blocks and a CUDA grid, so that the computational load is nearly equally distributed to the p threads, respecting the limitations of the CUDA grid geometry (for example up to up 1024 threads per block). This is in fact another optimization problem.

Figure 8. Flowchart of the two tasks of this study. Task 1: Evaluation of the speedup of the algorithm using GPU. Task 2: Exploit this potential for a sensitivity analysis confirming the reliability of the TOPINV-based exhaustive searches.

The details of the solution of this last optimization problem are explained in Section A2, and in addition, in Section A3, an example of how this problem is solved in one of our case studies is presented.

6 INVESTIGATION OF SPEEDUP IN INVERSION USING GPU PROCESSING

6.1 Physical models and synthetic data sets

As noted above, for the analysis we selected two physical models, the first describing ground deformation due to small magmatic sources described by the Mogi model (Mogi 1958). This model relates the location of a magma source (described by three Cartesian coordinates, x_M, y_M, z_M) and its volume (V), four variables in total, with the 3-D elastic dislocations observed at a certain control point s_i , usually a GPS station with known coordinates $x_{s_i}, y_{s_i}, z_{s_i}$. Eq. (2) can hence be written in the following form for displacement along the x -axis:

$$f(x_M, y_M, z_M, V, x_{s_i}, y_{s_i}, z_{s_i}) = \ell_x + v \quad (10)$$

Similar equations can be written for the y - and the z -axis and for all other observations points, selected with a realistic geographical distribution (Saltogian & Stiros 2013a). In the case of two magmatic sources with parameters $(x_{M1}, y_{M1}, z_{M1}, V_1), (x_{M2}, y_{M2}, z_{M2}, V_2)$, elastic dislocation from the two sources is accumulating and eq. (10) takes the form:

$$f(x_{M1}, y_{M1}, z_{M1}, V_1, x_{M2}, y_{M2}, z_{M2}, V_2, x_{s_i}, y_{s_i}, z_{s_i}) = \ell_x + v. \quad (11)$$

In our analysis, the variables characterizing the magma centres and the coordinates of the observations points were *a priori* known. Hence using eqs (10) and (11) and forward calculations displacements at measuring points were predicted. Random noise was added to the predicted displacements to simulate realistic GPS observations of displacement during volcanic unrest. A realistic search space and grids with different densities to permit different inversions were also adopted. The overall approach is explained in detail in Saltogian & Stiros (2013a).

A similar approach was adopted to obtain synthetic observations for Okada-type elastic dislocations, assuming a single fault, defined by nine variables (*Easting* and *Northing* of the center of the fault, *Depth* of its upper tip, *Length* and *Width*, *Strike*, *Dip*, *Slip* and *Rake*) or two faults defined by 18 variables. Faults and distribution of GPS

stations were inspired from realistic cases of earthquakes (Cervelli et al. 2001). Random noise was also added to model-predicted observations in order to obtain more realistic synthetic data. A realistic search space and grids with different densities to permit different inversions were also adopted.

This process led to 24 inversion cases in total corresponding to four sets of synthetic data: of single and double magma (Mogi) sources and of single and double seismic (Okada) faults, corresponding to grids with a different number of gridpoints. This is a number of cases which permits significant results.

6.2 Inversion

22 of the above 24 cases were inverted using the same algorithm and identical parameters, using two different processing schemes: A FORTRAN program running on a CPU and a CUDA program running on a GPU. Following certain preliminary investigations, a value of k was selected leading to selection of about 1000 gridpoints satisfying the SOE for the selected k for each case, a cluster of points not necessarily corresponding to an optimal solution. The reason was the need to test the compatibility of the two results using a relatively large sample of successful gridpoints.

For each of the 22 case studies, the set of approximately 1000 successful gridpoints for the two processing schemes were compared, and were found identical; in only a few cases up to 5% of the number of their gridpoints were different. These differences can safely be attributed to round-off errors, due to the use of single precision floating point numbers in the implementations. Such trivial differences had no effect in the pattern of distribution of gridpoints.

The last two out of the 24 cases were solved only using GPU, because the predicted execution time on CPU was extremely high (~ 42 d). The reason for these two last cases was to test the capability of the GPU platform for inversions using extra-large grids ($> 10^{12}$ gridpoints) in acceptable time frames.

Inversions using a CUDA program running on an NVidia GPU were computed on a system equipped with an Intel Core i7-3770K @ 3.50GHz, 16GB of memory and Ubuntu Linux 14.04.3 LTS as its operating system. The GPU used to perform the calculations is an NVidia Tesla K40c with 2880 cores and the CUDA version is 7.5.

6.3 Differences in execution time

Our first comparison of performance is between the aforementioned CUDA program and of an optimized FORTRAN program. Inversions for the FORTRAN program were performed on a conventional CPU-based computer. The system used is equipped with an Intel Core i5-3210M @ 2.50 GHz, 4 GB of memory and its operating system is Windows 10 (64-bit). The program has been compiled with the Intel Fortran Compiler 11.1.060. We selected this system due to the fact that inversions using TOPINV have been typically run on this system. Hence, the reported speedup of the GPU implementation reflects the speedup that we (as users of TOPINV) have really experienced.

For each of the 22 cases of groups of solutions derived from the two processing schemes (FORTRAN/CPU, CUDA/GPU), we recorded the execution time for the computation, as well as the ratio of the execution times in the two scenarios. Results are summarized in Fig. 9. This figure indicates an impressive decrease in the execution time using the GPU/CUDA version relative to the CPU/FORTRAN, of the order of 100. For very small grids speedups up to 2500 have been recorded (shaded, left part of Fig. 9), but these results are not significant, because the very short execution time in GPU, was probably within timing uncertainty.

For inversions with 18 unknown variables and a search space represented by up to 10^{12} gridpoints, GPU processing lasted between a few and 24 hr. This indicates that inversion of huge grids is feasible, but no nearly real-time results are expected with the hardware used.

6.4 Comparison of the GPU with an OpenMP implementation

Modern CPUs are composed of multiple cores, which can be exploited to speedup execution of applications. Our second comparison of performance is between the CUDA program and a new implementation of TOPINV that has been parallelized using OpenMP (OpenMP 2019). The OpenMP implementation has been run on the system that hosts the GPU used in this study. The CPU of this system has 4 cores, each of which supports HyperThreading, so that up to 8 threads can execute concurrently.

To perform the comparison we selected a grid that contains a large number of gridpoints, so that measurements of execution times are stable both on the CPU and the GPU, and that is favourable for the OpenMP implementation (grid G_{10} , see Section 7.1; Table B1). More specifically, one of the dimensions of this grid has 16 nodes, which is exactly divisible by all numbers of threads used in our comparison. This avoids uneven distribution of iterations among threads. Since this loop is the last in the structure of nested loops in our implementations, special care has also been taken to avoid repeated creation and finalization of parallel regions, which would negatively impact execution time.

Fig. 10 summarizes the speedup achieved when the OpenMP implementation is run with 1, 2, 4 and 8 threads. Despite the fact that the application is embarrassingly parallel, the grid used is favourable for the OpenMP implementation and special care has been taken to reduce overheads due to parallelization, speedup is not optimal. The reason is that the code of the physical model includes conditions that might exclude computations for gridpoints or requires special handling of gridpoints due to physical constraints. Hence, despite our effort to balance the number of iterations assigned to each thread, imbalance in computations is still present. Performance further declines when 8 threads are used. This is due to the fact that the CPU used has 4 cores, each of which supports HyperThreading.

In this case, 2 threads running on the same core share execution units of that core and delays are introduced.

Fig. 10(b) shows the speedup of the GPU implementation against the OpenMP implementation for different numbers of threads used in the second one. Even for the case of 8 threads, the GPU implementation is over 32 times faster than the OpenMP implementation.

7 IMPACT OF DISCRETE SAMPLING ON INVERSION RESULTS

7.1 Physical models and synthetic data generation

As noticed in Section 4.2, we used the elastic dislocation analysis, a common geophysical problem, to investigate the influence of grid sampling in the accuracy of the solutions (*cf.* Tarantola & Valette 1982) offered by the topological inversion using synthetic data. This is an approach somewhat similar with that of (Cervelli *et al.* 2001) and (Saltogian & Stiros 2013a). We adopted a known source of deformation (a typical Okada-type fault) producing small displacements (Table B2) with low SNR (of the order of 8), an array of 12 GPS observation points in an unfavourable observations geometry (see Fig. 1b), a broad search space (Table B1) and different grids covering the same search space. Random noise was added to forward-computed displacements (Table B2) in order to produce realistic synthetic observations of ground dislocation and lead to more realistic results.

In total 12 different grids (G_{01} – G_{12}) were examined. These grids become gradually denser (finer) with the addition of one point in each of the nine dimensions of the previous grid. The first three grids were intentionally selected too coarse. For example, in G_{01} the fault depth has only three nodes (so that the algorithm can assign the depth to three nodes only, 0, 10, 20 km), and in G_{12} it has 14 nodes with spacing $20/13 \approx 1.54$ km. Because of the selected method of grid generation, these grids share few common nodes only. The only variable is the grid density, practically described by parameter p , the number of total gridpoints discretizing the search space (see eq. 4), with p ranging between 5×10^6 and 10^{11} . In addition, in order to further control the algorithm performance, grids were formed using integers as reference ('true') values and grid boundaries, so that solutions can be identified with selected nodes (see Supplement S1).

7.2 Inversion and results

Because of the high computational load and the statistical basis of this study, we selected to use not formal, but rapid, 'heuristic' solutions, which are not optimal but satisfactory for the aim of this study, and in fact describe a lower-bound of the quality (accuracy and precision) of solutions.

Using an automatic search, we first identified the values of k leading to $q = 1$ and $q = 1000$ successful gridpoints in each of the 12 grids and then made the inversion for linearly interpolated values of k so that 10 solutions for each grid were available. For each solution and for each grid we computed the covariance matrix and mean misfits, and selected the solution corresponding to the minimum value of k for non-zero variances. Only G_{01} did not satisfy this last condition. Due to the coarseness of the grid, solutions were attracted by isolated grid nodes. Results are summarized in Fig. 11 and Table B3 and S2.

In addition, Fig. 12 was compiled using data from Table B3. This figure summarizes the variations of the mean accuracy δ , mean

Figure 9. Execution time (upper row) and speedup (lower row) of the TOPINV algorithm for CPU/FORTRAN-based and GPU/CUDA-based computations, for the cases examined (Mogi source and Okada fault inversions). Execution time increases exponentially with the number p of gridpoints of the grid G which represents a discrete form of the search area. An average speedup of the order of 100 is obtained with GPU/CUDA. Shaded is the area of statistically non-significant results.

Figure 10. (a) Speedup obtained from the OpenMP implementation as a function of the number of threads used. A non optimal speedup is observed due to imbalance in the computations performed in each thread. (b) Speedup of the GPU against the OpenMP implementation as a function of the number of threads used in the latter. The GPU implementation maintains a speedup of over 32 times.

precision s and mean misfits χ_v^2 of each solution as a function of the density of the grids. Mean accuracy δ and mean precision s were computed using values of each of the nine variables defining each solution normalized to the corresponding mean value of all the solutions and norm ℓ^2 .

7.3 Quality and significance of results

Fig. 11 summarizes the difference of solutions (dots) from their 'true' values (i.e. a measure of their accuracy) and their 1σ standard deviations (yellow shading, a measure of the precision of the solutions). The solutions of the very coarse grids G_{01} , G_{02} , G_{03} are

Figure 11. Accuracy (difference from the ‘true’ (reference) values) and precision (spread of data around the best value, yellow shade) of the nine variables defining a seismic fault for grids becoming gradually denser. Heuristic, rapid solutions. Accuracy and precision tend to increase with the grid density, while computed solutions are statistically not different from the ‘true’, reference values, even in an unfavorable case of inversion (small and poor distribution of observations). Results for very coarse grids G_{01} , G_{02} , G_{03} are shaded because of the inevitably low quality of solutions, satisfying specific diagnostic criteria. The pattern of attenuating stationary waves, clearer in ‘East’, is not an artefact, but evidence of good performance of the algorithm in discrete spaces (see Supplement S1).

of lower quality (misfits about one order of magnitude higher than in denser grids; Table S2), and for this reason they are shown with shading.

The higher quality solutions (grids G_{04} – G_{12}) are characterized by a consistent pattern, as can be derived from Fig. 11 with the following characteristics.

First, the plots of solutions as a function of the grid density resemble to stationary attenuating waves. This is not an artefact of computations, nor expresses instabilities characterizing various inversion schemes (e.g. Zahradnik & Gallovc 2010), but as is explained in details in Supplement S1, it indicates the efficiency and reliability of the algorithm. Concentration of the clusters of gridpoints defining a solution into a single node is to some degree expected for discrete grids, especially in coarse grids (see Table S1). This effect is intentionally amplified in the diagrams of Fig. 11, especially for variable ‘East’ (see Supplement S1) in order to explain the situation and indicate the efficacy of the algorithm.

Secondly, Fig. 11 indicates an increase in accuracy and precision with the density of the grid.

Thirdly, estimated values of variables in all grids G_{04} – G_{12} are within 1σ uncertainty levels from the ‘true’ values (i.e. statistically not different from their ‘true’ values even at the 1σ level) and differences (δ) from ‘true’ values are within a few degrees/km/cm only; this result is remarkable for such a poorly constrained problem.

Fourthly, Fig. 12 indicates that the mean accuracy δ , mean precision s and mean misfits χ_v^2 of each solution as a function of the density of the grids share a common trend (especially δ and χ_v^2), and above a certain resolution threshold they tend to become stabilized. Especially mean accuracy and mean misfits are characterized by a quasi-exponential trend, and above a certain threshold (in this case grid with resolution above that of G_{04}) their values tend to be stabilized.

8 DISCUSSION

The main advantage of TOPINV is that it has overpassed the main limitation of exhaustive searches, solution of problems with up to

Figure 12. Mean accuracy, mean precision and misfits of solutions of Fig. 11 as a function of the grid density, increasing towards G_{12} . Low-density grids G_{01} – G_{03} are shown shaded. Mean accuracy and misfits are characterized by an exponential trend and tend to be stabilized after a certain threshold.

3–4 unknown variables and can solve problems with at least 18 variables. The importance of this result is that exhaustive searches, the ideal tool for inversion (Mosegaard & Tarantola 1995) can be used for certain demanding cases of inversion such as modelling of seismic faults, even of seismic sequences associated with reactivation of two distinct faults (Saltogian & Stiros 2015a; Saltogian et al. 2018). Hence, the first main limitation of inversion based on exhaustive grid searches is at least partially overpassed. This study showed that GPU processing can overpass the second limitations of these approaches, the calculation speed: results summarized in Section 6 indicate that parallel computing in common desktop computers with low-cost GPU processors permit low latency inversions.

These two results permit to push the limits of exhaustive grid searches and make them suitable in certain demanding cases of inversion corresponding to complicated misfit surfaces characterized by several local (shallow) extrema but no a major global (deep) extremum. The new algorithm permits an approach which can be characterized as identification of all these extrema (Fig. 6) and not of the relatively higher which is the focus of most other algorithms.

The significance of this geometry of misfit surfaces is far from clear (Sambridge & Mosegaard 2002), and it was usually assigned to low SNR (Cervelli et al. 2001). Based on geodetic evidence (cf. Mikhail 1976) it was shown in this study that the complicated geometry of misfit surfaces is the combined effect of a mean SNR, of a high SNR of specific observations, which may indicate outliers or unsuitable physical model, and defects in the geometry of the observations system (Fig. 2d). An advantage of TOPINV is that it can

identify observations not fitting well the adopted physical model ('constraining observations', see Section 3.4). Another aspect of this problem is that all extrema represent solutions from the computational point of view, but not necessarily solutions of the physical problem; especially the relatively larger extremum, the difference of which from the smaller one is not usually documented through statistical techniques. This is one of the reasons for problems such as that of Fig. 3.

Taking advantage of the speedup of calculations, it was also investigated what the impact of grid density on the quality of results is. It was found that grids with density above a certain threshold lead to reliable stable results (Fig. 12).

Analysis in this study was based on carefully selecting synthetic data because they permit an accuracy-based approach (comparison of the solution and of its statistical characteristics with the 'true' solution, and focusing on unfavourable scenarios: on one hand low SNR data and poor observations geometry, and on the other hand selected grids to amplify the typical limitations of regular grids (Supplement S1). Just to notice that the results of Fig. 11 derive from synthetic measurements with a mean amplitude of a few cm and a SNR of an order as low as 8, while the fault of Fig. 3 was marked with displacements up to several metres; (see Barka et al. 2002). Hence results reflect a lower bound of the performance of the algorithm.

The new results pose some problems discussed below.

8.1 Maximum dimensions of inversion problems

In the past exhaustive searches were avoided because the minimum misfit criterion led to confusing or even biased results for problems with dimension $n > 3$. This problem is overpassed with the uncertainty criterion on which the new algorithm is based and results with $n = 18$ were obtained. The question arising is whether this value represents the limit of the algorithm. Clearly, no test to confirm this has been made, but the evidence from case studies with $n = 18$, in relatively low density grids (nested grids with 10^{11} points), low SNR and unfavourable configuration revealed a clear pattern of distribution of gridpoints [Figs S5, S6 in Saltogian et al. (2018)]. Hence, there is no reason to believe that using GPU processing this limit cannot be overpassed and problems for example with $n = 27$ (to cover modeling of three seismic faults) or even higher are not a realistic future option.

Of course, it is unlikely that inversion problems with hundreds of variables can be solved with TOPINV, and there is no reason to believe that a certain algorithm can/should cover all problems. However, for a class of problems with many (perhaps several hundred unknown variables), a different strategy can be adopted, a combination of two different algorithms, where a global search is first performed and then a local search to refine results (cf. Cervelli et al. 2001).

A common geophysical problem that can be solved using a combination of two algorithms is the calculation of variable slip models in seismic faults, in analogy to models shown in Fig. 3, but using observations of ground deformation (displacement) instead of seismological data. In this case, at a first step, the *non-linear* parameters of seismic faulting are defined using a *redundant* (and hence more reliable) system of observations with few unknown variables (a uniform slip fault model with 9 unknown variables for a single fault). TOPINV is ideal for this role. At a second step, the inversion is made for an *underdetermined*, *linear* variable slip model (or perhaps using another inversion algorithm), in which the unknown

variables are the two components of slip in numerous patches on the fault surface; the latter is defined by the results of the previous, uniform slip model and by certain additional constraints (for example Wang *et al.* 2013). This approach leads to more realistic models and avoids problems such as those highlighted by (Beresnev 2003).

Another possibility of the new algorithm is that it can incorporate as unknown variables not only data (eqs 1 and 2), but also some model parameters; the latter can be treated as typical unknowns, simply increasing the dimensionality of the problem.

Finally, if there is some evidence that the possibility of solutions is higher in specific parts of a search space, it is possible to use not a regular grid (with constant step) but with variable step, so that the frequency of sampling in some parts and certain axes is higher reducing the number of nodes and the computational cost.

8.2 From synthetic data to real world problems

The particularity of synthetic data is that they permit to recognize the accuracy, that is the deviation of a solution from the ‘true’, reference value, but this is not possible in real-world data. As has been explained in Section 3, certain, usually empirical diagnostic criteria can be used to speed up the solution process, mostly concerning the selection of the optimal grid [occasionally nested grids, see for example Saltogian *et al.* (2018)] and the selection of the range and the optimal value of parameter k . These criteria are to some degree based on the mean misfits which are useful for identification of a solution, but do not represent the solution criterion. This is because the TOPINV solutions, both formal and heuristic, for a certain range of k lead to statistically non-different results, and these results correspond to the area of relatively low mean misfits but not of minimum misfits. Hence, mean misfits, although they do not correspond to the selection criterion, they provide information which facilitates/speeds up the selection of the optimal solution, even if no empirical criteria for the selection of the solution are available.

9 CONCLUSIONS AND PERSPECTIVES

The overall conclusion of this study is that using a new inversion concept, the identification of clusters (sets, ensembles) of gridpoints with all misfits within parametric uncertainty levels, and GPU processing it became possible to push the limits of exhaustive searches so that some critical geophysical problems can be solved. The new algorithm is very much suitable for inversions leading to complicated mean misfit surfaces and has the potential to identify not only the major extremum but practically all smaller extrema. The significance of the latter is somewhat underestimated, since the relatively higher extremum is usually regarded as the physical solution of a problem. In fact, in observational systems characterized by defects (high measurement noise, low SNR and poor configuration) any of the local and minor peaks may represent physical solutions, and for these reasons all of them should be identified and evaluated. TOPINV seems to be in position to respond to this challenge, which was hampered by excessive computational cost.

Using the GPU/CUDA platform, which can be easily obtained by most investigators, a speedup in calculations of the order of 100 was obtained. This permits problems with 10^{12} gridpoints that require several days of processing on conventional desktop computers to complete in less than 2 hr, and problems with 10^{10} gridpoints require only 10 min, representing fully acceptable time frames (low latency, if not nearly real-time solutions).

This speedup can of course be amplified if the GPU/CUDA option is implemented on a computing platform in which multiple GPUs are available, or even on supercomputers. This will allow even faster analysis of problems with much larger regular search grids, either *a posteriori*, or even in nearly real-time. Obviously, the application of the TOPINV algorithm is not limited to the geophysical problems analysed, and the speed-up offered by the GPU/CUDA implementation may prove useful in various other fields.

ACKNOWLEDGEMENTS

The per was highly improved by constructive comments of Editor Jean Virieux, Evangelos Roussos and Jose Moreira.

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research. Research partly supported by the TSMEDE fund of the University of Patras.

REFERENCES

- Atzori, S. & Salvi, S., 2014. SAR data analysis in solid earth geophysics: from science to risk management, in *Land Applications of Radar Remote Sensing*, eds Holecz, F., Pasquali, P., Milisavljevic, N. & Closson, D., IntechOpen.
- Barka, A. *et al.*, 2002. The surface rupture and slip distribution of the 17 August 1999 Izmit earthquake (M 7.4), North Anatolian Fault, *Bull. seism. Soc. Am.*, **92**(1), 43–60.
- Battaglia, M., Cervelli, P. & Murray, J., 2013. Modeling crustal deformation near active faults and volcanic centers: a catalog of deformation models, in *Techniques and Methods, Book 13, Chap. B1*, p. 96, U.S. Geological Survey.
- Beresnev, I.A., 2003. Uncertainties in finite-fault slip inversions: to what extent to believe? (A critical review), *Bull. seism. Soc. Am.*, **93**(6), 2445–2458.
- Billings, S.D., Kennett, B.L.N. & Sambridge, M.S., 1994. Hypocentre location: genetic algorithms incorporating problem-specific information, *Geophys. J. Int.*, **118**(3), 693–706.
- Cervelli, P., Murray, M., Segall, P., Aoki, Y. & Kato, T., 2001. Estimating source parameters from deformation data, with an application to the March 1997 earthquake swarm off the Izu Peninsula, Japan, *J. geophys. Res.*, **106**(B6), 11 217–11 237.
- Curtis, A. & Lomax, A., 2001. Prior information, sampling distributions, and the curse of dimensionality, *Geophysics*, **66**(2), 372–378.
- Feng, L. & Newman, A.V., 2009. Constraints on continued episodic inflation at Long Valley Caldera, based on seismic and geodetic observations, *J. geophys. Res.*, **114**(B6), doi:10.1029/2008JB006240.
- Gallagher, K., Charvin, K., Nielsen, S., Sambridge, M. & Stephenson, J., 2009. Markov chain Monte Carlo (MCMC) sampling methods to determine optimal models, model resolution and model choice for Earth Science problems, *Mar. Petrol. Geol.*, **26**(4), 525–535.
- Gao, Z., Pan, Z. & Gao, J., 2016. Multimutation differential evolution algorithm and its application to seismic Inversion, *IEEE Trans. Geosci. Remote Sens.*, **54**(6), 3626–3636.
- Green, P.J., 1995. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination, *Biometrika*, **82**(4), 711–732.
- Harvey, B.R., 2013. Adjustment of survey networks by topological grid search, *J. appl. Geod.*, **7**(3), doi:10.1515/jag-2013-0006.
- Hu, Y., Han, L., Wu, R. & Xu, Y., 2019. Multi-scale time-frequency domain full waveform inversion with a weighted local correlation-phase misfit function, *J. geophys. Eng.*, **16**(6), 1017–1031.
- Lomax, A. & Snieder, R., 1995. Identifying sets of acceptable solutions to non-linear, geophysical inverse problems which have complicated misfit functions, *Nonlin. Process. Geophys.*, **2**(3/4), 222–227.

- Manavski, S.A., 2007. CUDA compatible GPU as an efficient hardware accelerator for AES cryptography, in *Proceedings of the 2007 IEEE International Conference on Signal Processing and Communications (ICSPC 2007)*, pp. 65–68, Dubai, United Arab Emirates.
- Manavski, S.A. & Valle, G., 2008. CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment, *BMC Bioinform.*, **9**(Suppl 2), doi:10.1186/1471-2105-9-S2-S10.
- Menke, W., 2012. *Geophysical Data Analysis: Discrete Inverse Theory (MATLAB Edition)*, 3rd edn, Elsevier.
- Mikhail, E., 1976. *Observations and Least Squares*, 1st edn, IEP - A Dun-Donnelley Publisher.
- Mogi, K., 1958. Relations between the eruptions of various volcanoes and the deformations of ground surfaces around them, *Bull. Earthq. Res. Inst.*, **36**(2), 99–134.
- Mosegaard, K. & Tarantola, A., 1995. Monte Carlo sampling of solutions to inverse problems, *J. geophys. Res.*, **100**(B7), 12431–12447.
- Nobile, M.S., Cazzaniga, P., Tangherloni, A. & Besozzi, D., 2016. Graphics processing units in bioinformatics, computational biology and systems biology, *Brief. Bioinform.*, **18**(5), 870–885.
- Nvidia, 2019. CUDA Web Site, <https://developer.nvidia.com/cuda-zone> [Online; accessed 23-April-2019].
- Okada, Y., 1985. Surface deformation due to shear and tensile faults in a half space, *Bull. seism. Soc. Am.*, **75**(4), 1135–1154.
- Oldenburg, D.W. & Li, Y., 2005. *Inversion for Applied Geophysics: A Tutorial*, pp. 89–150, Society of Exploration Geophysicists.
- OpenMP, 2019. *OpenMP Application Programming Interface*, OpenMP Architecture Review Board.
- Panza, G., 1981. The resolving power of seismic surface waves with respect to crust and upper mantle structural models, in *The Solution of the Inverse Problem in Geophysical Interpretation*, Ettore Majorana International Science Series, Vol. **11**, ed. Cassinis, R., pp. 39–77, Springer US.
- Saltogian, V. & Stiros, S., 2013a. A new algorithm for modelling simple and double Mogi magma sources in active volcanoes: accuracy, sensitivity, limitations and implications, *Bull. Volcanol.*, **75**(10), doi:10.1007/s00445-013-0754-x.
- Saltogian, V. & Stiros, S., 2013b. Topological inversion in geodesy-based, non-linear problems in geophysics, *Comput. Geosci.*, **52**, 379–388.
- Saltogian, V. & Stiros, S., 2015a. A two-fault model of the 2003 Leucas (Aegean Arc) earthquake based on topological inversion of GPS data, *Bull. seism. Soc. Am.*, **105**(5), 2510–2520.
- Saltogian, V., Stiros, S., Newman, A.V., Flanagan, K. & Moschas, F., 2014. Time-space modeling of the dynamics of Santorini volcano (Greece) during the 2011–2012 unrest, *J. geophys. Res.*, **119**(11), 8517–8537.
- Saltogian, V., Taymaz, T., Yolsal-Çevikbilen, S., Eken, T., Moschas, F. & Stiros, S., 2017. Fault model for the 2015 Leucas (Aegean Arc) earthquake: analysis based on seismological and geodetic observations, *Bull. seism. Soc. Am.*, **107**(1), 433–444.
- Saltogian, V., Moschas, F. & Stiros, S., 2018. The 2014 Cephalonia earthquakes: finite fault modeling, fault segmentation, shear and thrusting at the NW Aegean Arc (Greece), *Pure appl. Geophys.*, **175**(12), 4145–4164.
- Sambridge, M. & Gallagher, K., 1993. Earthquake hypocenter location using genetic algorithms, *Bull. seism. Soc. Am.*, **83**(5), 1467–1491.
- Sambridge, M. & Mosegaard, K., 2002. Monte Carlo methods in geophysical inverse problems, *Rev. Geophys.*, **40**(3), 1009.
- Sambridge, M., Bodin, T., Gallagher, K. & Tkalčić, H., 2013. Transdimensional inference in the geosciences, *Phil. Trans. R. Soc., A*, **371**(184), doi:10.1098/rsta.2011.0547.
- Sen, M.K. & Stoffa, P.L., 1995. *Global Optimization Methods in Geophysical Inversion*, 1st edn, Elsevier.
- Shibutani, T., Sambridge, M. & Kennett, B., 1996. Genetic algorithm inversion for receiver functions with application to crust and uppermost mantle structure beneath eastern Australia, *Geophys. Res. Lett.*, **23**(14), 1829–1832.
- Sokos, E. et al., 2015. Rupture process of the 2014 Cephalonia, Greece, earthquake doublet (M_w 6) as inferred from regional and local seismic data, *Tectonophysics*, **656**, 131–141.
- Tarantola, A. & Valette, B., 1982. Generalized nonlinear inverse problems solved using the least squares criterion, *Rev. Geophys. Space Phys.*, **20**(2), 219–232.
- Tasoulis, D.K. & Vrahatis, M.N., 2006. Unsupervised clustering using fractal dimension, *Int. J. Bifurcat. Chaos*, **16**(07), 2073–2079.
- Tiampo, K., Rundle, J., Fernandez, J. & Langbein, J., 2000. Spherical and ellipsoidal volcanic sources at Long Valley caldera, California, using a genetic algorithm inversion technique, *J. Volc. Geotherm. Res.*, **102**(3–4), 189–206.
- Tölke, J. & Krafczyk, M., 2008. TeraFLOP computing on a desktop PC with GPUs for 3D CFD, *Int. J. Comput. Fluid Dyn.*, **22**(7), 443–456.
- TOP 500, 2019. Top 500 list of supercomputers, <https://www.top500.org> [Online; accessed 23-April-2019].
- Wang, R., Parolai, S., Ge, M., Jin, M., Walter, T.R. & Zschau, J., 2013. The 2011 Mw 9.0 Tohoku Earthquake: comparison of GPS and strong-motion data, *Bull. seism. Soc. Am.*, **103**(2B), 1336–1347.
- Wright, T.J., Houlié, N., Hildyard, M. & Iwabuchi, T., 2012. Real-time, reliable magnitudes for large earthquakes from 1 Hz GPS precise point positioning: the 2011 Tohoku-Oki (Japan) earthquake, *Geophys. Res. Lett.*, **39**(12), doi:10.1029/2012GL051894.
- Yang, X.S., 2010. A new metaheuristic bat-inspired algorithm, in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, Vol. 284 of *Studies in Computational Intelligence*, pp. 65–74, eds González, J.R., Pelta, D.A., Cruz, C., Terrazas, G. & Krasnogor, N., Springer.
- Zahradnik, J. & Gallovc, F., 2010. Toward understanding slip inversion uncertainty and artifacts, *J. geophys. Res.*, **115**(B9), doi:10.1029/2010JB007414.

SUPPORTING INFORMATION

Supplementary data are available at [GJI](https://doi.org/10.1002/gj.14755) online.

Table S1. Distribution of nodes of the grids summarized in Fig. 12 for variable ‘East’; data based on Table B1. Because of intentionally selected lower and upper limits and the grid densification process, the ‘true’ value of the variable coincides with a node of grids G02, G04, G06, etc. (shown by red circle), but is found at the middle of two nodes of grids G01, G03, G05, etc. (position marked by an ellipse). This explains the pattern of Fig. 12 (pattern of a stationary, attenuating wave). This pattern characterizes all variables of Fig. 12, but it is very clear in ‘East’ and a measure of the good performance of the algorithm.

Table S2. This composite table summarizes data used for the selection of the optimal heuristic solution (in red, shown in Table B3) for each of the 12 grids G_{01} – G_{12} and from 12 sets of successful gridpoints with $q = 1$ to $q = 1000$. Variables k , the number q of the corresponding successful gridpoints and their misfits x_v^2 , as well as variables of zero variance for grids G_{01} – G_{12} are shown. A characteristic systematic pattern in the variables is evident.

Please note: Oxford University Press is not responsible for the content or functionality of any supporting materials supplied by the authors. Any queries (other than missing material) should be directed to the corresponding author for the paper.

APPENDIX

A1 CUDA architecture for parallel computing

As noted in Section 5.1, *threads* are the computational entities or series of calculations that have to be performed on each core on the GPU. In order to facilitate easier mapping of applications to the programming model, CUDA bundles threads into *blocks*, which

arrange threads in 1, 2 or 3 dimensions. Blocks are further organized into a *CUDA grid*, which can also have up to 3 dimensions (Fig. A1). The number of dimensions does not need to be the same for the CUDA grid and the blocks, but both are subject to certain limitations. For example, each block may contain up to 1024 threads (512 in certain older CUDA versions). The number of dimensions for blocks and the CUDA grid, the number of threads on each dimension within each block, as well as the number of blocks on each dimension within the CUDA grid are defined by certain variables (*threadIdx.x*, *threadIdx.y*, etc.) explained in Table A1 and Fig. A1. A more specific example is presented in Section A3.

This organization allows to easily map 1-D, 2-D and 3-D problems on CUDA. As an example, assume that elements of a three dimensional array of size $9 \times 12 \times 3$ have to be processed. Fig. A1 shows an example of a CUDA grid that could be used for this purpose, with blocks being arranged in two dimensions and threads within a block being arranged in three dimensions. CUDA provides built-in variables that represent the size of the CUDA grid and of each block, as well as the coordinates of each block within the CUDA grid and of each thread within a block. Using this information each thread can decide on the element of the array it has to process. For example, the thread of the CUDA grid with the different colour can determine, using these built-in variables, that it has to process element (3, 8, 1) of the array. It is also worth noting that the CUDA grid used in this example is not the only one that could have been used. Programmers are responsible to decide the most appropriate CUDA grid to be used for a given problem and CUDA will automatically adapt the information contained in the built-in variables representing the CUDA grid.

A2 CUDA implementation for n -dimensional problem

Typically the most interesting and computationally demanding problems where TOPINV can be used, and hence a GPU implementation is of interest, have at least four dimensions. In this paper, we examine problems with up to $n = 18$ dimensions. These numbers exceed the available number of dimensions of the CUDA grid. Hence, an important issue to be handled is how to map the gridpoints of the n -dimensional TOPINV grid onto the threads within the CUDA grid.

For the purpose of demonstrating possible approaches to tackle the issue and be able to depict these approaches, we will assume a 3-D physical problem. Furthermore, we will assume that the CUDA grid can have only up to two dimensions, so that indeed the dimensions of the problem are larger than the dimensions of the CUDA grid.

Fig. A2 presents a possible approach how each thread might decide which gridpoints from the discretized search space it will process when a 2-D CUDA grid with 2-D blocks of threads is used. Similarly, Fig. A3 presents a possible approach when a 1-D CUDA grid with 1-D blocks of threads is used.

The two previous examples assume that the number of TOPINV gridpoints per dimension exceed the size of the corresponding block of threads on the same dimension. Typically, however, this is not the case. Even for quite dense discretizations, the number of gridpoints per dimension does not reach 20 in our experiments (see Section 7). On the other hand, the implementation of CUDA on contemporary GPUs imposes a limit of 512 or 1024 threads per block, with the exact limit being dependent on the specific GPU being used. For efficient exploitation of the computational resources of a GPU, we adopt an approach of using a number of threads within a block as

Algorithm 2 TOPINV for the GPU

```

1: Input:  $k$                                 ▷ Parameter  $k$  in eq.(8)
2: Input:  $n$                                 ▷ Number of dimensions
3: Input:  $m$                                 ▷ Number of observations
                                           (displacement measurements  $\ell$ )
4: Input:  $G[n][3]$                           ▷ Grid: min, max and spacing values
                                           per dimension
5: Input:  $D[m]$                             ▷ Observations (displacement measurements  $\ell$ )
6: Input:  $I = \{m_1, m_2, \dots, m_n\}$       ▷ Number of grid-points
                                           per dimension (eq. (4))
7:
8:  $maxTPB \leftarrow$  maximum allowed number of Threads Per Block
9:  $maxBPG \leftarrow$  maximum allowed number of Blocks Per Grid
10:
11: Select  $T = \{m_{t_1}, m_{t_2}, \dots, m_{t_{k_1}}\} \subseteq I, k_1 \leq n$  such that  $TPB = \prod_{i=1}^{k_1} m_{t_i}$  is the largest number that is  $\leq maxTPB$ 
12: Select  $B = \{m_{b_1}, m_{b_2}, \dots, m_{b_{k_2}}\} \subseteq I - T, k_2 \leq n - k_1$  such that  $BPG = \prod_{i=1}^{k_2} m_{b_i}$  is the largest number that is  $\leq maxBPG$ 
13:
14: Solve  $\lll BPG, TPB \ggg (G, n, D, m, I, T, B, BPG, TPB)$ 

```

near as possible to this limit. As a result, a single dimension of the TOPINV grid is not sufficient to fill a block of threads. Furthermore, there is no gain in using two or three dimensions for the CUDA grid for the problems under consideration, compared to using only one dimension. In fact, using a single dimension proves to be more straightforward in translating coordinates of threads to coordinates in the TOPINV grid. Therefore, the approach adopted in our work is to create a 1-D CUDA grid containing 1-D blocks, with multiple dimensions of the TOPINV grid being used to fill each block. Hence, each thread must determine the gridpoints it has to process within the TOPINV grid by translating the one dimensional coordinates of each thread into n -dimensional coordinates of gridpoints in the TOPINV grid. For this purpose the built-in variables provided by CUDA are used (*blockIdx*, *threadIdx*, *gridDim*, *blockDim*). Fig. A4 presents an application of our approach.

Taking into account the above, our algorithm, presented in Algorithm 2, starts by selecting a subset T of k_1 dimensions of the n -dimensional TOPINV grid (Line 11), so that the number of gridpoints in that subset is as near as possible to the limit of threads within a block. To determine the number of gridpoints it is necessary to know the number of gridpoints per dimension (Line 6). The purpose of this step is for each block to handle a set of gridpoints of the TOPINV grid for which the combination of the k_1 selected coordinates can be fully distributed among threads, as presented in the example of Fig. A4. This means that each thread within a block has a single, constant value for each of these dimensions, but different threads have a different combination of constant values.

Similarly, there is a limit on the number of blocks that a grid can contain. Our algorithm continues by selecting a subset B of k_2 dimensions from the remaining dimensions of the TOPINV grid (Line 12), so that the number of gridpoints in that subset is as near as possible to the limit of blocks within the grid. Using the aforementioned built-in variables of CUDA, every block in the grid selects a unique combination of k_2 coordinates. The combined result of this and the previous step is that every thread in the CUDA grid will be assigned a unique combination of $k_1 + k_2$ coordinates from the TOPINV grid. If more dimensions remain, then each thread will handle multiple gridpoints of the TOPINV grid by iterating through the iteration space of all of the remaining dimensions, but will always keep the $k_1 + k_2$ coordinates constant.

Figure A1. A 3×4 CUDA grid. Each block consists of 27 threads arranged in 3 dimensions ($3 \times 3 \times 3$). Coordinates for each dimension start at zero. The size of the CUDA grid on each dimension is provided by CUDA built-in variables *gridDim.x* and *gridDim.y*, which have values 4 and 3, respectively, as 4 blocks of threads are available on the *x*-axis and 3 on the *y*-axis. Information about the size of each block of threads is provided by built-in variables *blockDim.x*, *blockDim.y* and *blockDim.z*. All of them have a value of 3, since 3 threads on each dimension comprise a block. To identify the thread with the different color, the coordinates of the containing block within the grid have to be determined first. Built-in variables *blockIdx.x* and *blockIdx.y* provide these coordinates and have values 2 and 1, respectively. Coordinates of the thread within the block are provided by built-in variables *threadIdx.x*, *threadIdx.y*, *threadIdx.z*, which have values 2, 0 and 1, respectively (details only for *threadIdx.x* are depicted).

Table A1. Built-in variables of CUDA, read by threads to determine their coordinates within the CUDA grid.

Variable	Description
<i>gridDim.x</i> <i>gridDim.y</i> <i>gridDim.z</i>	Number of blocks on the corresponding dimension of the CUDA grid
<i>blockDim.x</i> <i>blockDim.y</i> <i>blockDim.z</i>	Number of threads on the corresponding dimension of a block
<i>blockIdx.x</i> <i>blockIdx.y</i> <i>blockIdx.z</i>	Coordinate of a block within the CUDA grid on the corresponding dimension (0 to number of blocks on dimension - 1)
<i>threadIdx.x</i> <i>threadIdx.y</i> <i>threadIdx.z</i>	Coordinate of a thread within a block on the corresponding dimension (0 to number of threads on dimension - 1)

Having decided how to distribute work, the *computational kernel* that solves the problem under consideration, that is a function that encapsulates the code that has to be executed on each thread, can be called on the GPU (Line 14). The use of the one dimensional grid can be seen in Line 14, where variables *BPG* and *TPB* determine the number of blocks in the grid and the number of threads within each block. It is worth noting that Algorithm 2 only provides a method to distribute work. Therefore, it can be used for any problem where an exhaustive search is required, as it does not depend on the specifics of each problem.

On the other hand, routines that find the acceptable solutions must take into account the specific equations that describe the problem. Hence, the computational kernel *SOLVE()* must be defined for each problem independently. Using the information produced by Algorithm 2 with respect to work distribution, the routine must examine each dimension of the TOPINV grid. If a dimension belongs to the subset *T*, each thread within the same block selects a single grid-point from the TOPINV grid. The same procedure must be repeated if the dimension belongs to the subset *B*. Otherwise, the whole iteration space of the dimension has to be traversed by the thread. For

Figure A2. Mapping of a discretized search space on a 2-D CUDA grid. The CUDA grid has been defined so that the plane (x, y) in (b) can be exactly mapped to the CUDA grid. Gridpoints of (b) will be processed by same-coloured threads of (a). Each thread in (a) selects a pair of coordinates on the (x, y) plane. These coordinates remain constant for each thread, but all other dimensions are swept [z -axis in (b)]. For example, the light-blue coloured thread will process all gridpoints in the light-blue ellipse.

Figure A3. Mapping of a discretized search space on a 1-D CUDA grid. The CUDA grid has been defined so that the x dimension in (b) can be exactly mapped to the CUDA grid. Gridpoints of (b) will be processed by same-coloured threads of (a). Each thread in (a) selects a single coordinate of the x dimension. This coordinate remains constant for each thread, but all other dimensions are swept [y - and z -axis in (b)]. For example, the light-blue colored thread will process all gridpoints contained in the light-blue ‘slice’, whereas the light-orange coloured thread will process all gridpoints contained in the light-orange ‘slice’.

every gridpoint a thread has to examine, it tests whether eq. (8) is satisfied. If so, the gridpoint is a solution of the system under the TOPINV algorithm and it is added to the set of solutions found.

A3 Example of implementation of the GPU/CUDA grid in a TOPINV problem

We examine the case of an inversion problem discussed in Section 7.1, corresponding to Grid G_{04} in Table B1. The parameters of this 9-D grid are summarized in Table A2. This grid consists of $p = 9 \cdot 9 \cdot 6 \cdot 9 \cdot 9 \cdot 8 \cdot 8 \cdot 11 \cdot 10 = 277\,136\,640$ gridpoints that have to be divided among threads. This distribution of variables is not compatible with the default architecture of the CUDA grid (1–3 dimensions, Fig. A1) and for this reason an architecture involving a 1-D grid is selected (Fig. A4). This grid is divided into blocks and each block to threads.

Step 1. Algorithm 2 searches which combination of products of $m_1, m_2, \dots, m_n$ gives a value as close as possible (but lower) than 1024, which is the limit of threads within a block. The answer in our example is $m_1 \cdot m_8 \cdot m_9 = 990$. Hence these three variables (x_1, x_8, x_9) are used to form an optimal block (contains 990 threads), characterized by a unique combination of the nodes of the variables x_1, x_8, x_9 . This block represents the basic unit for distribution of the p threads into blocks and grid.

Step 2. The remaining 6 dimensions ($x_2, x_3, \dots, x_7$), corresponding to $9 \cdot 6 \cdot 9 \cdot 9 \cdot 8 \cdot 8 = 279\,936$ combinations, have to be allocated to blocks. The straightforward way is to allocate them to 279 936 blocks, each of which will contain 990 threads, or combinations of variables $x_1, x_2, \dots, x_9$.

Step 3. All $p = 990 \cdot 279\,936 = 277\,136\,640$ gridpoints have been allocated to 279 936 blocks, each representing a unique combination of the values of the each variable, that is of all p gridpoints. This

Figure A4. The 1-D CUDA grid has been defined so that the combined x and y dimensions in (b) can be exactly mapped to the CUDA grid. Special attention is paid so that the block size is less, but as near as possible, to the limit imposed by CUDA. Each thread in (a) selects a unique combination of coordinates of the x and y dimensions. In this example, the selection is made per row. These coordinates remain constant for each thread, but all other dimensions are swept [z -axis in (b)]. For example, the light-blue colored threads will process all gridpoints contained in the corresponding light-blue sections of the TOPINV grid.

Table A2. An example of implementation of CUDA in the case of one of our case studies (grid G_{04} in Table B1). Values with bold indicate those selected at ‘Step 1’ to build the CUDA grid.

x_1 (East)	x_2 (North)	x_3 (Depth)	x_4 (Strike)	x_5 (Dip)	x_6 (Length)	x_7 (Width)	x_8 (Rake)	x_9 (Slip)
m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
9	9	6	9	9	8	8	11	10

Table A3. Values that the built-in variables of CUDA receive for the specific example presented in this section.

Variable	Value
$gridDim.x$	279 936 (Number of blocks on dimension x)
$gridDim.y$	Unused (CUDA grid has only 1 dimension)
$gridDim.z$	Unused (CUDA grid has only 1 dimension)
$blockDim.x$	990 (Number of threads on dimension x)
$blockDim.y$	Unused (Blocks have only 1 dimension)
$blockDim.z$	Unused (Blocks have only 1 dimension)
$blockIdx.x$	Ranging from 0 to 279 935 (Unique for each block of the CUDA grid)
$blockIdx.y$	Unused
$blockIdx.z$	Unused
$threadIdx.x$	Ranging from 0 to 989 (Unique for each thread of a block)
$threadIdx.y$	Unused
$threadIdx.z$	Unused

signifies an hierarchy, with all possible combinations of the points of the three selected variables x_1, x_8, x_9 with all possible combinations of the remaining variables $x_2, x_3, \dots, x_7$. These 279 936 blocks correspond to a 1-D CUDA grid.

In some cases (for example because of limitations in the number of threads per block in older CUDA versions), the above approach is not possible. In this case, an easy alternative is to regard as a thread not the calculations for a *single* gridpoint, but for a combination of points, corresponding to the different possible values of a certain dimension. For example, a thread may correspond to calculations for a unique combination of all dimensions excluding one, for example

x_8 , and of all m_8 combinations of this dimension. Practically this means that a thread does not correspond to a single gridpoint, but to a specific set of gridpoints (covering one dimension).

Table A3 presents the values that the built-in variables of CUDA (Table A1) receive for the above example. These numbers are used by each thread to determine the specific gridpoint it has to process.

APPENDIX B:

Summary of input data for the sensitivity analysis (‘Task 2’ in Fig. 8).

Table B1. Summary of the characteristics of the search space and of the most coarse- (G_{01}) and of the most fine-grained grid (G_{12}) used in the sensitivity analysis ('Task 2'). Intermediate grids (G_{02} , G_{03} , etc.) are formed adding one additional node in the data of each dimension.

Dimension	Ref. value	Search space		Gridpoints per dimension	
		Min. value	Max. value	G_{01}	G_{12}
East (km)	50	25	75	6	17
North (km)	50	30	80	6	17
Depth (km)	5	0	20	3	14
Strike ($^{\circ}$)	25	5	55	6	17
Dip ($^{\circ}$)	55	30	80	6	17
Length (km)	45	20	60	5	16
Width (km)	20	5	45	5	16
Rake ($^{\circ}$)	120	100	170	8	19
Slip (cm)	60	30	90	7	18
Total points				5.44×10^6	1.02×10^{11}

Table B2. Coordinates of the 12 stations used in the inversions for the sensitivity analysis and synthetic observations produced adding some random noise ($\sigma = 3$ mm for Easting, Northing, $\sigma = 5$ mm for vertical) to model-predicted displacements.

Coordinates of observation points		Model-predicted displacements			Added noise			Synthetic observations of displacement		
x (km)	y (km)	d_{East} (cm)	d_{North} (cm)	d_{Up} (cm)	East (cm)	North (cm)	Up (cm)	d_{East} (cm)	d_{North} (cm)	d_{Up} (cm)
35.0	75.0	4.2	-1.8	-2.1	0.3	-0.1	0.4	4.5	-1.9	-1.7
67.5	80.0	-1.4	0.1	-0.4	-0.1	0.1	0.2	-1.5	0.2	-0.2
52.5	70.0	-2.1	3.2	8.6	-0.3	-0.4	-0.4	-2.4	2.8	8.2
22.5	60.0	6.4	-1.4	-1.8	0.2	0.1	0.4	6.6	-1.3	-1.4
42.5	56.0	0.0	0.6	1.3	-0.1	-0.3	0.0	-0.1	0.3	1.3
67.5	62.5	-2.0	-0.9	2.9	-0.1	0.4	-0.4	-2.1	-0.5	2.5
25.5	39.0	6.8	0.0	-2.6	-0.2	-0.1	-0.2	6.6	-0.1	-2.8
51.0	45.0	0.6	-9.4	17.8	-0.4	0.3	0.3	0.2	-9.1	18.1
72.5	47.5	-2.7	-1.3	0.9	-0.2	-0.2	-0.5	-2.9	-1.5	0.4
35.5	26.0	-2.9	-7.5	6.0	0.1	-0.3	0.2	-2.8	-7.8	6.2
65.0	39.0	-1.0	-3.3	2.7	-0.1	-0.1	0.0	-1.1	-3.4	2.7
57.5	22.5	0.8	-4.6	2.7	0.4	0.5	-0.2	1.2	-4.1	2.5

Table B3. Parameters of the solutions. Values rounded to the second decimal. Mean misfit corresponds to the solution derived from a set of gridpoints. Mean misfit corresponds to χ_v^2 , a dimensionless variable, corresponding to the mean difference between observations and predictions divided by the corresponding uncertainty of observations.

	p (Total gridpoints)	k (Uncertainty level)	q (Gridpoints defining the solution)	Mean misfit
G_{01}	5.44×10^6	8.02	11	16.93
G_{02}	2.49×10^7	8.65	290	6.12
G_{03}	9.03×10^7	6.83	67	4.12
G_{04}	2.77×10^8	6.41	266	1.96
G_{05}	7.48×10^8	4.87	13	1.81
G_{06}	1.83×10^9	5.07	244	0.87
G_{07}	4.11×10^9	4.40	54	0.82
G_{08}	8.64×10^9	4.74	598	0.85
G_{09}	1.71×10^{10}	3.95	115	0.84
G_{10}	3.24×10^{10}	4.02	542	0.49
G_{11}	5.87×10^{10}	3.59	191	0.46
G_{12}	1.02×10^{11}	3.56	539	0.36