Parallel Finite Element Computations for Soil–Foundation—Structure Interaction Problems

Report UCD–CompGeoMech–02–07

by:

Boris Jeremić and Guanzhou Jie

Computational Geomechanics Group

Department of Civil and Environmental Engineering

University of California, Davis

Report version: 1. May, 2008, 14:59

The work presented in this short report was supported in part by the following grant sources: Civil and Mechanical System program, Directorate of Engineering of the National Science Foundation, award NSF–CMS–0201231 (cognizant program director Dr. Richard Fragaszy); award NSF–CMS–0337811 (cognizant program director Dr. Steve McCabe); Center for Information Technology Research in the Interest of Society (CITRIS); and by the Department of Civil and Environmental Engineering at the University of California at Davis.

ABSTRACT: Described here is the Plastic Domain Decomposition (PDD) Method for parallel elastic-plastic finite element computations related to Soil–Foundation–Structure Interaction (SFSI) problems. The PDD provides for parallel elastic-plastic finite element computations steered by an adaptable, run-time repartitioning of the finite element domain. The adaptable repartitioning aims at balancing computational load among processing nodes (CPUs), while minimizing inter–processor communications and data redistribution during elasto-plastic computations. The PDD method is applied to large scale SFSI problem. Presented examples show scalability and performance of the PDD computations. In addition to that, an interesting examples of computed SFSI behavior is shown.

The main goal of this short report is to show, besides describing PDD in some details and showing set of interesting results, that detailed, high fidelity modeling and simulations of geotechnical and structural systems is currently available and can be done very efficiently. By using high fidelity models in simulations, modeling uncertainty is reduced significantly which increases our confidence in produced results.

Detailed report describing the Plastic Domain Decomposition method and related design and implementation details is available as well (Jeremić and Jie (17))

For sources or installation help of the parallel simulation system, please contact the first Author. Parallel simulation system is currently installed at our local parallel computer GeoWulf, as well as on SDSC, TACC and CU Boulder parallel machines.

INTRODUCTION

Parallel finite element computations have been developed for a number of years mostly for elastic solids and structures. The static domain decomposition (DD) methodology is currently used almost exclusively for decomposing such elastic finite element domains in subdomains. This subdivision has two main purposes, namely (a) to distribute element computations to CPUs in an even manner and (b) to distribute system of equations evenly to CPUs for maximum efficiency in solution process.

However, in the case of inelastic (elastic–plastic) computations. the static DD is not the most optimal method, as some subdomains become computationally slow as the elastic–plastic zone propagates through the domain. This propagation of the elastic plastic zone (extent of which is not know a–priori) will in turn slow down element level computations (constitutive level iterations) significantly, depending on the complexity of the material model used. This plastification will finally result in some subdomains becoming very computationally heavy (and thus slow for computations) while some others (that are still mostly elastic) will still be very computationally efficient. This discrepancy in computational efficiency between different subdomains will result in inefficient parallel performance. In other words, subdomains (and their respective CPUs) with mostly elastic elements will be finishing their local iterations much faster (and idle afterwords) than subdomains (and their respective CPUs) that have elastic–plastic elements.

This computational imbalance motivated development of the so called Plastic Domain Decomposition (PDD) method described in some details in this short report. Developed PDD is applied to a large scale seismic soil–foundation–structure (SFS) interaction problem for bridge systems. It is important to note that the detailed analysis of seismic SFSI became possible only with the development of PDD as the modeling requirements (finite element mesh size, size of the prototype bridge...) were such that sequential simulations were out of questions at this time.

Motivation

Main motivation for the development of PDD is the need for detailed analysis of realistic, large scale SFSI models. Proper modeling of seismic wave propagation in elastic–plastic soils dictates finite element sizes that result in large models. For example, recent work on full dynamic simulations of a soil–foundation–bridge system resulted in a number of finite element models with different degrees of sophistication. Table 1 gives an overview of model size (number of elements and element size) as function of cutoff frequencies represented in the model, material (soil) stiffness (given in terms of G/G_{max}, however full incremental elasto–plasticity was used for soil modeling) and amount of shear deformation for given stiffness reduction. It is important to

Table 1: Variation in model size (number of elements and element size) as function of frequency, stiffness and shear deformation.

model size (# of elements)	element size	f_{cutoff}	min. $G/Gmax$	shear def. γ
12K	1.0 m	10 Hz	1.0	$<$0.5 %
15K	0.9 m	$>$3 Hz	0.08	1.0 %
150K	0.3 m	10 Hz	0.08	1.0 %
500K	0.15 m	10 Hz	0.02	5.0 %

not the the largest FEM model had over 0.5 million elements and over 1.6 million DOFs. One of the FEM model meshes is shown in Figure 1.

Detailed analysis of this finite element model of the prototype bridge system required that loading be separated into a number of load stages. The first load stage was soil self weight. This was done on a soil only model. The space for piles was then excavated and piles placed into the open holes. It is important to note that the piles are modeled as nonlinear beam–column elements. The connection of the beam–column nodes, that are in the middle of the void occupied by the pile volume, with soil at the edge of of that void, is done using stiff beams. This setup allows for very realistic modeling of interaction of nonlinear piles and surrounding soils. The second stage of loading is self weight of piles. The thirds load stage comprises building (placement) of structure on top of piles and application of its own self weight. Only after these loading stages is the model ready to be dynamically excited by the earthquake motions. Two types of soil were chosen for this study, namely the stiff, sandy soil and and the soft, bay mud. Soil types were

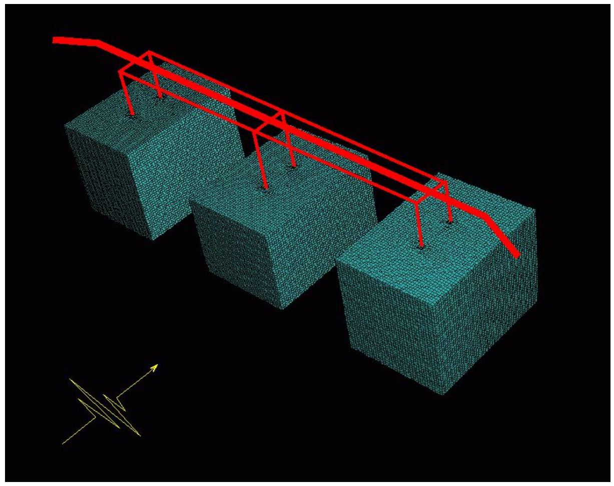

Figure 1: FEM model for seismic response of a three bend, four span bridge SFS system.

alternated beneath bridge bents in order to investigate effects of variation of site conditions on dynamics of a bridge system.

The elastic plastic model for stiff soil (sand) was represented by a combination of the Drucker–Prager yield surface, similar plastic potential surface and the Armstrong–Frederick rotational kinematic hardening (Jeremić (16)). The elastic–plastic material model for soft clay was composed of perfectly plastic von Mises yield surface and similar (volume preserving) plastic potential function. Both models were developed by blending of components through the use of Template 3D EP concept developed by (Jeremić and Yang (21)). Ground motions for Northridge and Kocaeli earthquakes (measured at the free field surface) were used and propagated to certain depth (beneath our model) using 1D wave propagation solutions as implemented in SHAKE91 program (Idriss and Sun (15)). The vertical propagation of free field motions were then used to develop loads needed by the DRM method (Bielak et al (4)). The structural bridge model (nonlinear bridge piers and the elastic box girder) was developed in collaboration with the University of California at Berkeley (Fenves et al.) and University of Washington (Eberhard et al.).

Literature Overview

The idea of domain decomposition method can be found in an original paper from 1870 by H.A. Schwarz (Rixena and Magoulès (32). Current state of the art in distributed computing in computational mechanics can be followed to early works on parallel simulation technology. For example, early endeavors using inelastic finite elements focused on structural problems within tightly coupled, shared memory parallel architectures. We mention work by Noor et al. (30), Utku et al. (38) and Storaasil and Bergan (36) in which they used substructuring to achieve distributed parallelism. Fulton and Su (12) developed techniques to account for different types of elements used in the same computer model but used substructures made of same element types which resulted in non–efficient use of compute resources. Hajjar and Abel (13) developed techniques for dynamic analysis of framed structures with the objective of minimizing communications for a high speed, local grid of computer resource. Klaas et al. (24) developed parallel computational techniques for elastic–plastic problems but tied the algorithm to the specific multiprocessor computers used (and specific network connectivity architecture) thus rendering it less general and non-useful for other types of grid arrangements. Farhat (8) developed the so–called Greedy domain partitioning algorithm. None of the above frameworks is capable to perform highly efficient simulations on a loosely coupled, heterogeneous computational grid. More recently Farhat et al. (11; 9; 10) proposed FETI (Finite Element Tearing and Interconnecting) method for domain decomposition analysis. In FETI method, Lagrange multipliers are introduced to enforce compatibility at the interface nodes.

Although many works have been presented on domain decomposition methods, the most popular methods such as FETI-type and BDDC all stem from the root of subdomain interface constraints handling. The merging of iterative solving with domain decomposition-type preconditioning is promising as shown by many researchers (Pavarino (31) and Li and Widlund (25)). Schwartz-type preconditioners for parallel domain decomposition system solving have also shared part of the spotlight (Hwang and Cai (14) and Sarkis and Szyld (33)). From the implementation point of view, for mesh-based scientific computations, domain decomposition corresponds to the problem of mapping a mesh onto a set of processors, which is well defined as a graph partitioning problem (Schlegel et al. (34)).

A number of papers in recent years have investigated the influence of the SSI on behavior of bridges (22; 27; 26; 37; 28; 5; 7). McCallen and Romstadt (28) performed a remarkable full scale analysis of the soil–foundation–bridge system. The soil material (cohesionless soil, sand) was modeled using equivalent elastic approach (using Ramberg–Osgood material model through standard modulus reduction and damping curves developed by Seed et al. (35)). The two studies by Chen and Penzien (5) and by Dendrou et al. (7) analyzed the bridge system including the soil but the developed models used a very coarse finite element meshes. Jeremić et al. (18) attempted a detailed, complete bridge system analysis. However, due to computational limitations, the large scale pile group had to be modeled separately and its stiffness used with the bridge structural model. In present work, with the development of PDD (described in some details), such computational limitations are removed and high fidelity, detailed models of Earthquake–Soil–Structure systems can be performed.

Simulation Platform

Developed parallel simulation program uses a number of numerical libraries. Namely. Graph partitioning is achieved using ParMETIS libraries (Karypis et al. (23)). Parts of the OpenSees framework (29) were used to connect the finite element domain. In particular, Finite Element Model Classes from OpenSees (namely, class abstractions Node, Element, Constraint, Load and Domain) where used to describe the finite element model and to store the results of the analysis performed on the model. In addition to that, an existing Analysis Classes were used as basis for development of parallel PDD framework which is then used to drive the global level finite element analysis, i.e., to form and solve the global system of equations in parallel. On a lower level, a set of Template3Dep numerical libraries (Jeremić and Yang (21)) were used for constitutive level integrations, nDarray numerical libraries (Jeremić and Sture (20)) were used to handle vector, matrix and tensor manipulations, while FEMtools element libraries from the UCD CompGeoMech toolset (Jeremić (16)) were used to supply other necessary libraries and components. Parallel solution of the system of equations has been provided by PETSc set of numerical libraries (Balay et al. (2; 1; 3)).

Most of the simulations were carried out on our local parallel computer GeoWulf. Only the largest models (too big to fit in GeoWulf system) were simulated on TeraGrid machine (at SDSC and TACC). It should be noted that the software part of the simulation platform (executable program) is available through Author's web site.

Plastic Domain Decomposition Method

Domain Decomposition (DD) approach is the most popular and effective method to implement parallel finite element method. The underlying idea is to divide the problem domain into subdomains so that finite element calculations will be performed on each individual subdomain in parallel. The DD can be overlapping or non-overlapping. The overlapping domain decomposition method divides the problem domain into several slightly overlapping subdomains. Non-overlapping domain decomposition is extensively used in continuum finite element modeling due to the relative ease to program and organize computations and is the one that will be examined in this work. In general, a good non-overlapping decomposition algorithm should be able to (a) handle irregular mesh of arbitrarily shaped domain, and (b) minimize the interface problem size by delivering minimum boundary conductivities, which will help reducing the communication overheads.

The Elastic–Plastic Parallel Finite Element Computational Problem

The distinct feature of elastic-plastic finite element computations is the presence of two iteration levels. In a standard displacement based finite element implementation, constitutive driver at each integration (Gauss) point iterates in stress and internal variable space, computes the updated stress state, constitutive stiffness tensor and delivers them to the finite element functions. Finite element functions then use the updated stresses and stiffness tensors to integrate new (internal) nodal forces and element stiffness matrix. Then, on global level, nonlinear equations are iterated on until equilibrium between internal and external forces is satisfied within some tolerance.

Elastic Computations. In the case of elastic computations constitutive driver has a simple task of computing increment in stresses ($\Delta\sigma_{ij}$) for a given deformation increment ($\Delta\epsilon_{kl}$), through a closed form equation ($\Delta\sigma_{ij} = E_{ijkl}\Delta\epsilon_{kl}$) It is important to note that in this case the amount of work per Gauss point is known in advance and is the same for every integration point. If we assume the same number of integration points per element, it follows that the amount of computational work is the same for each element and it is known in advance.

Elastic-Plastic Computations. On the other hand, for elastic-plastic problems, for a given incremental deformation the constitutive driver is iterating in stress and internal variable space until consistency condition is satisfied ($F \approx 0$). The number of iterations is not known in advance. Initially, all Gauss points are in elastic range, but as the incremental loads are applied, the elastic–plastic zones develops. For integration points still in elastic range, there are no iterations, the constitutive driver just computes incremental stresses from closed form solution. Computational load will increase significantly for integration of constitutive equations in elastic–plastic range. In particular, constitutive level integration algorithms for soils, concrete, rocks, foams and other granular materials are very computationally demanding. More than 70% of wall clock time during an elastic-plastic finite element analysis can be spent in constitutive level iterations. This is in sharp contrast with elastic computations where the dominant part is solving the system of equations which consumes about 90% of run time. The extent of additional, constitutive level iterations is not known before the actual computations are over. In other words, the extent of elastic-plastic domain is not known ahead of time.

The traditional pre–processing type of Domain Decomposition method (also known as topological DD) splits domain based on the initial geometry and assigns roughly the same number of elements to every computational node and minimizes the size of subdomain boundaries. This approach might result in serious computational load imbalance for elastic–plastic problems. For example one subdomain might be assigned all of the elastic–plastic elements and spend large amount of time in constitutive level iterations. The other subdomain will have elements in elastic state and thus spend far less computational time in computing stress increments. This results in program having to wait for the slowest subdomain (the one with large number of elastic-plastic finite elements) to complete constitutive level iterations and only proceed with global system iterations after that.

The two main challenges with computational load balancing for elastic–plastic dynamic (and static as well) are that they need to be:

- Adaptive, dynamically load balancing computations, as the extent of elastic and elastic-plastic domains changes dynamically and unpredictably during the course of the computation.

- Multiphase computation as elastic-plastic computations follow up the elastic computations and there is a synchronization phase between those two. The existence of the synchronization step between the two phases of the computation requires that each phase be individually load balanced.

Plastic Domain Decomposition Algorithm

The Plastic Domain Decomposition algorithm (PDD) provides (computational load) balanced subdomains, minimizes subdomain boundaries and minimizes the cost of data redistribution during The PDD optimization algorithm is based on dynamically monitoring both data redistribution and analysis model regeneration costs during program execution in addition to collecting information about the cost of constitutive level iterations within each finite element. A static domain decomposition is used to create initial partitioning, which is used for the first load increment. Since, after that first load step, parts of the computational domain will (might) plastify, a computational load imbalance might occur. Computational load (re–)balancing will (might) be triggered if the performance gain offsets the extra cost associated with the repartitioning. The decision on whether to trigger repartitioning or not is based on an algorithm described in some details below.

We define the global overhead associated with load balancing operation as two parts, data communication cost T_{comm} and finite element model regeneration cost T_{regen},

$$T_{overhead} := T_{comm} + T_{regen} \quad (1)$$

Performance counters have been setup within the program to study both. Data communication patterns characterizing the network configuration can be readily measured (T_{comm}) as the program runs the initial partitioning. Initial (static) domain decomposition is performed for the first load step. The cost of networking is inherently changing as the network condition might vary as simulation progresses, so whenever data redistribution happens, the metric is automatically updated to reflect the most current network conditions. Model regeneration cost (T_{regen}) is a result of a need to regenerate the analysis model whenever elements (and nodes) are moved between computational nodes (CPUs). It is important to note that model (re–) generation also happens initially when the fist data distribution is done (from the static DD). Such initial DD phase provides excellent initial estimate of the model regeneration cost on any specific hardware configurations.

For the load balancing operations to be effective, the $T_{overhead}$ has to be offset by the performance gain T_{gain}. The computational load imposed by each finite element (FE) is represented by the associated vertex weight $vwgt[i]$ (as the finite element mesh is represented by a graph, with each FE representing a vertex). If the SUM operation is applied on every single processing node, the exact computational distribution among processors can be obtained as total wall clock time for each CPU

$$T_j := \sum_{i=1}^{n} vwgt[i],\ j = 1, 2, \ldots, np \tag{2}$$

in which n is the number of elements on each processing domain and np is the number of CPUs. The wall clock time is controlled by T_{max}, defined as

$$T_{sum} := sum(T_j),\ T_{max} := max(T_j),\ \text{and}\ T_{min} := min(T_j),\ j = 1, 2, \ldots, np \tag{3}$$

and minimizing T_{max} becomes the main objective. Computational load balancing operations comprises delivering evenly distributed computational loads among processors. Theoretically, the best execution time is,

$$T_{best} := T_{sum}/np,\ \text{and}\ T_j \equiv T_{best},\ j = 1, 2, \ldots, np \tag{4}$$

if a perfect load balance is (to be) achieved.

Based on definitions above, the best performance gain T_{gain} that can be obtained from computational load balancing operation can be calculated as,

$$T_{gain} := T_{max} - T_{best} \tag{5}$$

Finally, the load balancing operation will be beneficial iff

$$T_{gain} \geq T_{overhead} = T_{comm} + T_{regen} \tag{6}$$

Previous equation is used in deciding if the re–partitioning is triggered in current incremental step. It is important to note that PDD will always outperform static DD as static DD represents the first decomposition of the computational domain. If such decomposition becomes computationally unbalanced and efficiency can be gained by repartitioning, PDD be triggered and the T_{max} will be minimized.

Solution of the System of Equations

Only brief details of the solution of the system of equations in presented here. To this end, a number of different algorithms and implementations exists for solving unsymmetric[1] systems of equations in parallel. In presented work, use was made of both iterative and direct solvers as available through the PETSc interface (Balay et al. (2; 1; 3)). Direct solvers, including MUMPS, SPOOLES, SuperLU, PLAPACK have been tested and used for performance assessment. In addition to that, iterative solvers, including GMRES, as well as preconditioning techniques (Jacobi, inconsistency LU decomposition and approximate inverse preconditioners) for Krylov methods have been also used and their performance assessed.

Some of the conclusions drawn are that, for our specific finite element models (non–symmetric, using penalty method for some connections, possible softening behavior), direct solvers outperform the iterative solver significantly. As expected direct solver where not as scalable as iterative solvers, however, specifics of our finite element models resulted in poor initial performance of iterative solvers, that, even with excellent performance scaling, could not catch up with the efficiency of direct solvers. Final conclusion of this brief overview of the system of equation solvers use is that parallel direct solvers such as MUMPS and SPOOLES showed the best performance and would be recommended for use with finite element models that, as ours did, feature non–symmetry, are poorly conditioned (they are ill–posed due to use of penalty method) and can be negative definite (for softening materials).

PDD Scalability Study

The above developed methodology was tested on a number of static and dynamic examples. Presented here are scalability (speed–up) results for a series of soil–foundation–structure model runs. A hierarchy of models, described in Introduction section was used for scaling study for dynamic runs. Finite element models were subjected to 1997 Northridge earthquake. Total wall clock time has been recorded and used to analyze the parallel scalability of PDD, which are presented in Figure 2.

There are some interesting observation about the performance scaling results shown in Figure

[1] Non-associated elasto–plasticity results in a non-symmetric stiffness tensors, which result in non–symmetric system of finite element equations. Non–symmetry can also result from the use of consistent stiffness operators as described by Jeremić and Sture (19).

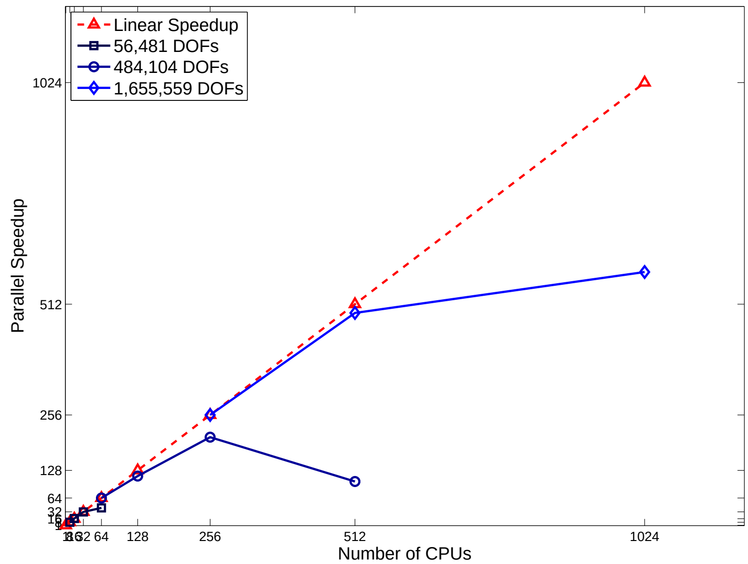

Figure 2: Scalability Study on 3 Bent SFSI Models, DRM Earthquake Loading, Transient Analysis, ITR=1e-3, Imbal Tol 5%, Performance Downgrade Due to Increasing Network Overhead

2:

- the scalability is quite linear for small number of DOFs (elements) and it such linearity extends to larger number of DOFs as well,
- there is a certain relation between number of DOFs (elements) and number of CPUs which governs the parallel efficiency. In other words, there exists certain ratio of the number of DOFs to number of CPUs after which the communication overhead starts to be significant. For example for a models with 484,104 DOFs in Figure 2, the computations with 256 CPUs are more efficient that those with 512 CPUs. This means that for the same number of DOFs (elements) doubling the number of CPUs does not help, rather it is detrimental as there is far more communication between CPUs which destroys the efficiency of the parallel computation. Similar trend is observable for the large model with 1,655,559 DOFs, where 1024 CPUs will still help (barely) increase the parallel efficiency.
- Another interesting observation has to do with the relative computational balance of local, element level computations (local equilibrium iterations) and the system of equations solver. PDD scales very nicely as its main efficiency objective is to have equal computational load for element level computations. However, the efficiency of the system of equations solver

becomes more prominent when element level computations are less prominent (if they have been significantly optimized with a large efficiency gain). For example, for the model with 56,481 DOFs it is observed that for sequential case (1 CPU), elemental computation amount for approx. 70 % of wall clock time. For the same model, in parallel with 8 CPUs, element level computation accounts for approx. 40% of wall clock time, while for 32 CPUs the element level computation account for only 10% of total wall clock time. In other words, as the number of CPUs increase, the element level computations are becoming very efficient and the main efficiency gain can then be made with the system of equations solver. However, it is important to note that parallel direct solver are not scalable up to large number of CPUs (Demmel et al. (6)) while parallel iterative solver are much more scalable but difficult to guarantee convergence.

ILLUSTRATIVE RESULTS

Bridge model described above was used to analyze a number of cases of different foundation soils and earthquake excitations. Presented here is one specific set of results. Our purpose here is not to make an in depth analysis of bridge system behavior during an earthquake (which we have done), but rather to show how large scale, high fidelity detailed modeling and simulations can help make new discoveries explain behavior of such complex soil–structure systems.

Two sets of ground motions were used for the same bridge structure. Variation of foundation soil was made in order to investigate if free field motions can be directly used for structural model input (as is almost exclusively done nowadays), that is, to investigate how significant are the SFSI effects.

Ground motions for Northridge and Kocaeli earthquakes (measured at the free field) were used and propagated (de–convoluted) to certain depth (beneath our model) using 1D wave propagation solutions as implemented in SHAKE91 program (Idriss and Sun (15)). The vertical propagation of free field motions were then used to develop loads needed by the DRM method. The structural bridge model (nonlinear bridge piers (beam–columns) and the elastic box girder) was developed in collaboration with the University of California at Berkeley (Fenves et al.) and University of Washington (Eberhard et al.).

Two types of input motions were considered in this exercise. Since the main aim of the exercise was to investigate SFS system a set of short period motions were chosen among Northridge motions records, while long period motions from Kocaeli earthquakes were used for long period example. Record of accelerations and displacements for both motions are given in Figure 3.

A very important aspect of SFSI is the difference between free field motions and the motions that are changed (affected) but the presence of the structure. Figure 4 shows comparison of free field short period motions (obtained by vertical propagation of earthquake motions through the model without the presence of bridge structure and piles) and the ones recorded at the base of column of the left bent in stiff and soft soils. It is immediately obvious that the free field motions

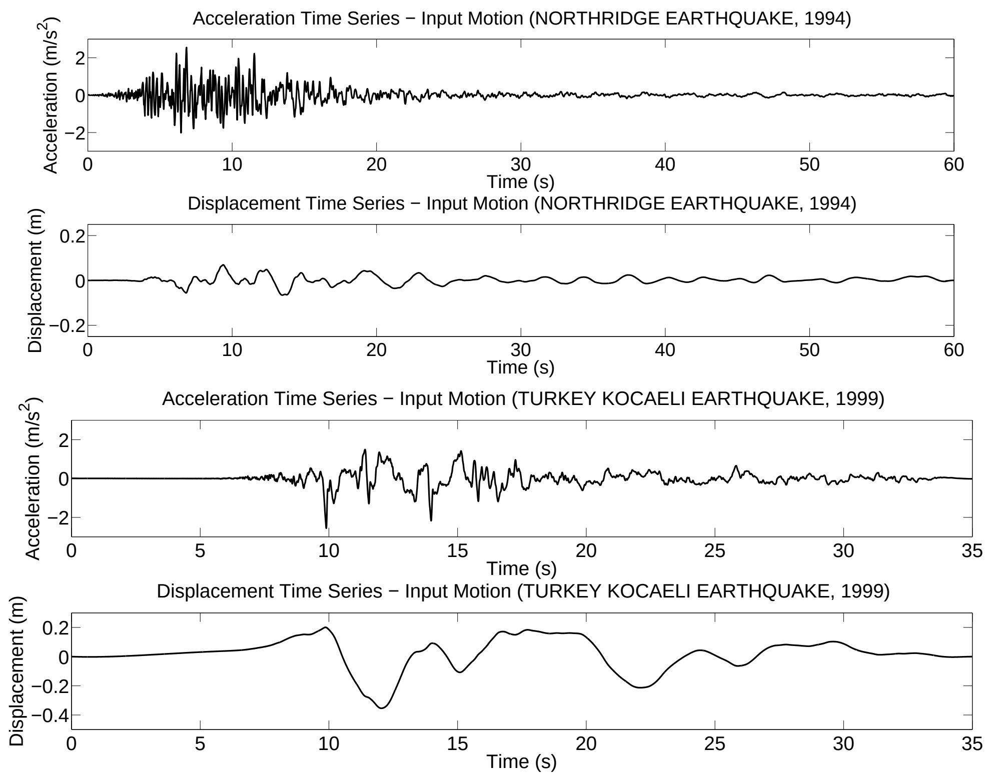

Figure 3: Input motions: short period (Northridge) and long period (Kocaeli).

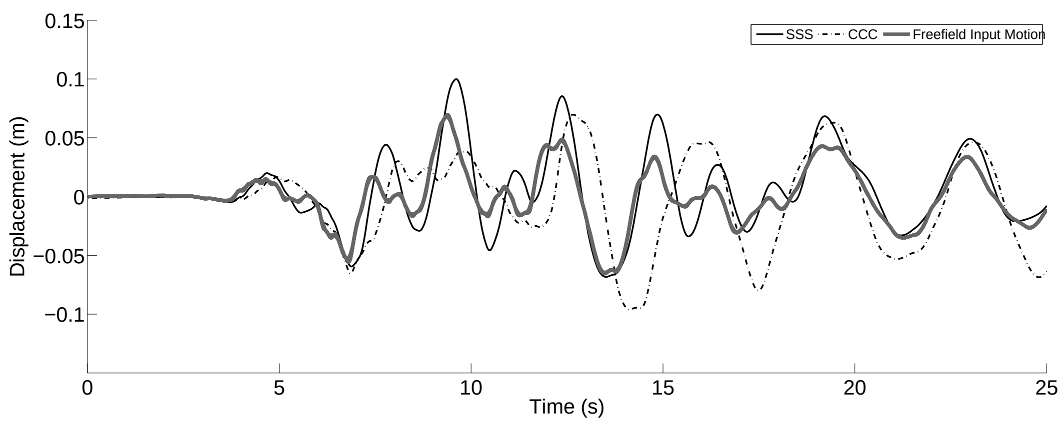

Figure 4: Comparison of free field versus measured (prototype model) motions at the base of left bent for the short period motions for all clay (CCC) and all sand (SSS) soils.

in this case do not correspond (not even close) to motions observed in bridge SFS system with stiff or soft soils. In fact, both the amplitude and period are significantly altered for both soft and stiff soil and the bridge structure. This quite different behavior can be explained by taking into account the fact that the short period motions excite natural periods of stiff soil and so produce (significant) amplifications. In addition to that, for soft soils, significant elongation of period is observed.

On the other hand, as shown in Figure 5 the same SFS system (same structure with stiff or soft soil beneath) responds quite a bit different to long period motions. The difference between

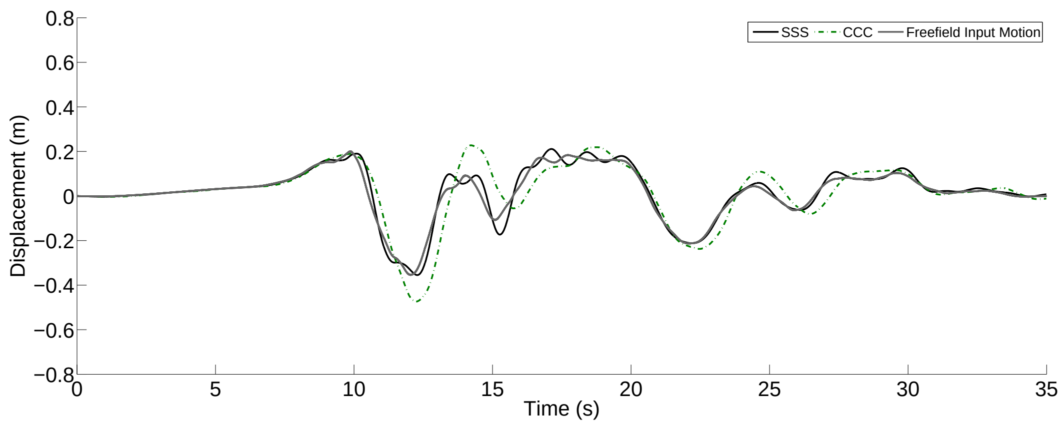

Figure 5: Comparison of free field versus measured (prototype model) motions at the base of left bent for the long period motions for all clay (CCC) and all sand (SSS) soils.

free field motions and the motions measured (simulated) in stiff soils is very small. This is understandable as the stiff soil virtually gets carried away as (almost) rigid block on such long period motions. For the soft soil one of the predominant periods of the SFS system is excited briefly (at around 12-17 seconds) but other than that excursion, both stiff and soft soil show fair matching with free field motions. In this case the SFS effects are not that pronounced, except during the above mentioned period between 12 and 17 seconds.

SUMMARY

In this short report an algorithm, called the Plastic Domain Decomposition (PDD), for parallel elastic–plastic finite element computations was presented, including a parallel scalability study. Presented also was an application of PDD to soil–foundation–structure interaction problem for a bridge system and Earthquake–Soil–Structure (ESS) interaction effects were emphasized. The importance of the (miss–) matching of the ESS triad to the dynamic behavior of the bridge Soil–Structure was shown on an example using same structure, two different earthquakes (one with short and one with long predominant periods) and two different subgrade soils (stiff sand and soft clay)

The main goal of this paper is to show that high fidelity, detailed modeling and simulations of geotechanical and structural systems are available and quite effective. Results from such high fidelity modeling and simulation shed light on new types of behavior that cannot be discovered using simplified models. These high fidelity models reduce uncertainty to (currently) lowest possible level so that there is high confidence that newly discovered behavior is very realistic.

Bibliography

[1] Satish Balay, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, 2004.

[2] Satish Balay, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page, 2001. `http://www.mcs.anl.gov/petsc`.

[3] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[4] Jacobo Bielak, Kostas Loukakis, Yoshaiki Hisada, and Chaiki Yoshimura. Domain reduction method for three–dimensional earthquake modeling in localized regions. part I: Theory. *Bulletin of the Seismological Society of America*, 93(2):817–824, 2003.

[5] M. A. Chen and J. Penzien. Nonlinear soil–structure interaction of skew highway bridge. Report UCB/EERC-77/24, Earthquake Engineering Research Center, 1977.

[6] James W. Demmel, Stanley C. Eisenstat, John R. Gilbert, Xiaoye S. Li, and Joseph W. H. Liu. A supernodal approach to sparse partial pivoting. *SIAM J. Matrix Analysis and Applications*, 20(3):720–755, 1999.

[7] B. Dendrou, S. D. Werner, and T. Toridis. Three dimensional response of a concrete bridge system to traveling seismic waves. *Computers and Structures*, 20:593–603, 1985.

[8] Charbel Farhat. *Muliprocessors in Computational Mechanics*. PhD thesis, University of California, Berkeley, 1987.

[9] Charbel Farhat. Saddle-point principle domain decomposition method for the solution of solid mechanics problems. In *Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, Norfolk, VA, USA, May 6-8 1991.

[10] Charbel Farhat and M. Geradin. Using a reduced number of lagrange multipliers for assembling parallel incomplete field finite element approximations. *Computer Methods in Applied Mechanics and Engineering*, 97(3):333–354, June 1992.

[11] Charbel Farhat and Francois-Xavier Roux. Method of finite element tearing and interconnecting and its parallel solution algorithm. *International Journal for Numerical Methods in Engineering*, 32(6):1205–1227, October 1991.

[12] R. E. Fulton and P. S. Su. Parallel substructure approach for massively parallel computers. *Computers in Engineering*, 2:75–82, 1992.

[13] J. F. Hajjar and J. F. Abel. Parallel processing for transient nonlinear structural dynamics of three–dimensional framed structures using domain decomposition. *Computers & Structures*, 30(6):1237–1254, 1988.

[14] Feng-Nan Hwang and Xiao-Chuan Cai. A class of parallel two-level nonlinear Schwarz preconditioned inexact Newton algorithms . *Computer Methods in Applied Mechanics and Engineering*, 196(8):1603–1611, January 2007.

[15] I. M. Idriss and J. I. Sun. *Excerpts from USER'S Manual for SHAKE91: A Computer Program for Conducting Equivalent Linear Seismic Response Analyses of Horizontally Layered Soil Deposits*. Center for Geotechnical Modeling Department of Civil & Environmental Engineering University of California Davis, California, 1992.

[16] Boris Jeremić. Lecture notes on computational geomechanics: Inelastic finite elements for pressure sensitive materials. Technical Report UCD-CompGeoMech–01–2004, University of California, Davis, 2004. available online: `http://sokocalo.engr.ucdavis.edu/~jeremic/CG/LN.pdf`.

[17] Boris Jeremić and Guanzhou Jie. Plastic domain decomposition method for parallel elastic–plastic finite element computations in gomechanics. Technical Report UCD–CompGeoMech–03–07, University of California, Davis, 2007. available online: `http://sokocalo.engr.ucdavis.edu/~jeremic/wwwpublications/CV-R24.pdf`.

[18] Boris Jeremić, Sashi Kunnath, and Feng Xiong. Influence of soil–structure interaction on seismic response of bridges. International Journal for Engineering Structures, 26(3):391–402, February 2003.

[19] Boris Jeremić and Stein Sture. Implicit integrations in elasto–plastic geotechnics. *International Journal of Mechanics of Cohesive–Frictional Materials*, 2:165–183, 1997.

[20] Boris Jeremić and Stein Sture. Tensor data objects in finite element programming. *International Journal for Numerical Methods in Engineering*, 41:113–126, 1998.

[21] Boris Jeremić and Zhaohui Yang. Template elastic–plastic computations in geomechanics. *International Journal for Numerical and Analytical Methods in Geomechanics*, 26(14):1407–1427, 2002.

[22] A. J. Kappos, G. D. Manolis, and I. F. Moschonas. Sismic assessment and design of R/C bridges with irregular configuration, including SSI effects. *International Journal of Engineering Structures*, 24:1337–1348, 2002.

[23] George Karypis, Kirk Schloegel, and Vipin Kumar. *ParMETIS: Parallel Graph Partitioning and Sparse Matrix Ordering Library*. University of Minnesota.

[24] O. Klaas, M. Kreienmeyer, and E. Stein. Elastoplastic finite element analysis on a MIMD parallel–computer. *Engineering Computations*, 11:347–355, 1994.

[25] Jing Li and Olof B. Widlund. On the use of inexact subdomain solvers for BDDC algorithms . *Computer Methods in Applied Mechanics and Engineering*, 196(8):1415–1428, January 2007.

[26] N. Makris, D. Badoni, E. Delis, and G. Gazetas. Prediction of observed bridge response with soil–pile–structure interaction. *ASCE Journal of Structural Engineering*, 120(10):2992–3011, October 1994.

[27] N. Makris, D. Badoni, E. Delis, and G. Gazetas. Prediction of observed bridge response with soil-pile-structure interaction. *ASCE J. of Structural Engineering*, 120(10):2992–3011, 1995.

[28] D. B. McCallen and K. M. Romstadt. Analysis of a skewed short span, box girder overpass. *Earthquake Spectra*, 10(4):729–755, 1994.

[29] Francis Thomas McKenna. *Object-Oriented Finite Element Programming: Frameworks for Analysis, Algorithms and Parallel Computing*. PhD thesis, University of California, Berkeley, 1997.

[30] A. K. Noor, A. Kamel, and R. E. Fulton. Substructuring techniques – status and projections. *Computers & Structures*, 8:628–632, 1978.

[31] Luca F. Pavarino. BDDC and FETI-DP preconditioners for spectral element discretizations. *Computer Methods in Applied Mechanics and Engineering*, 196(8):1380–1388, January 2007.

[32] Daniel Rixena and Frédéric Magoulès. Domain decomposition methods: Recent advances and new challenges in engineering. *Computer Methods in Applied Mechanics and Engineering*, 196(8):1345–1346, January 2007.

[33] Marcus Sarkis and Daniel B. Szyld. Optimal left and right additive Schwarz preconditioning for minimal residual methods with Euclidean and energy norms. *Computer Methods in Applied Mechanics and Engineering*, 196(8):1612–1621, January 2007.

[34] Kirk Schloegel, George Kaypis, and Vipin Kumar. Graph partitioning for high performance scientific simulations. Technical report, Army HPC Research Center, Department of Computer Science and Engineering, University of Minnesota, 1999.

[35] H. B. Seed, R. T. Wong, I. M. Idriss, and K. Tokimatsu. Moduli and damping factros for dynamic analysis of cohesionless soils. Earthquake Engineering Research Center Report UCB/EERC 84/14, University of California Berkeley, 1984.

[36] O. O. Storaasli and P. Bergan. Nonlinear substructuring method for concurrent processing computers. *AIAA Journal*, 25(6):871–876, 1987.

[37] J. Sweet. A technique for nonlinear soil–structure interaction. Technical Report CAI-093-100, Caltrans, Sacramento, California, 1993.

[38] S. Utku, R. Melosh, M. Islam, and M. Salama. On nonlinear finite element analysis in single– multi– and parallel processors. *Computers & Structures*, 15(1):39–47, 1982.