České vysoké učení technické v Praze
Fakulta stavební

Czech Tecnical University in Prague
Faculty of Civil Engineering

Doc. Dr. Ing. Bořek Patzák

Parallel Computations in Structural Mechanics

Paralelní výpočty v mechanice konstrukcí

Summary

The recent developments in many scientific and engineering disciplines bring in new challenges for computational science. The Finite Element Method has become a widely used tool for solving problems described by partial differential equations. Its application often leads to nonlinear models of very complex geometries and many degrees of freedom. Such complex analyses initiate demands for large-scale computing, which must be feasible from the view of both time and available resources. This naturally leads to utilization of parallel processing, that allows not only obtaining results in acceptable time by significantly speeding up the analysis, but also performing large and complex analyses, which often do not fit into a single, even well equipped, machine with one processor unit (regardless of achieved speedup).

The design of parallel algorithms requires the partitioning of the problem into a set of tasks, the number of which is greater than or equal to the number of available processors. The partitioning of the problem can be fixed at runtime (static load balancing) or can change during the solution (dynamic load balancing). The latter option is often necessary in order to achieve good load balancing of work among processors and thus optimal scalability. The load balance recovery is achieved by repartitioning of the problem domain and transferring the work (represented typically by finite elements) from one processor to another. The repartitioning is an optimization problem with multiple constraints, optimal algorithm should balance the work while minimizing the work transfer and keeping the sub-domain interfaces as small as possible.

To illustrate the potential of the parallel adaptive finite element analysis, the present summary offers two particular examples of nonlinear fracture analysis: 2D simulation of Brazilian splitting test and 3D analysis of anchor pull out.

Souhrn

Během posledního desetiletí došlo k výraznému pokroku v mnoha vědeckých a inženýrských oborech, které nastolily nové výzvy v numerickém modelování. Metoda konečných prvků se stala nejpoužívanější numerickou metodou pro řešení problémů popsaných systémem parciálních diferenciálních rovnic. Běžně se používá pro řešení nelineárních problémů s komplexní geometrií a mnoha stupni volnosti. Modelování komplexních problémů vede přirozeně k rozsáhlým adaptivním výpočtům, které jsou limitovány dostupným výpočetním časem a dostupnými výpočetními prostředky. To vede k paralelním výpočtům, které nejen umožní získat požadované výsledky v přijatelném čase zrychlením výpočtu, ale často také v případě rozsáhlých problémů výpočet vůbec provést, neboť i špičkově vybavený standardní počítač nemá dostatek prostředků pro jeho provedení.

Návrh paralelního algoritmu vyžaduje rozdělení problému na jednotlivé dílčí úlohy, jejichž počet je větší nebo roven počtu dostupných výpočetních jednotek. Dělení úlohy může být během řešení konstantní, nebo se může během řešení měnit. Dynamická adaptace dělení je často nezbytná pro dosažení dobrého vyvážení výpočetní zátěže mezi výpočetními jednotkami a tedy dobré škálovatelnosti algoritmu. Obnova rovnoměrné výpočetní zátěže vyžaduje nové přerozdělení a přenos výpočetní zátěže z vytížených výpočetních jednotek na méně zatížené. Rozdělení či přerozdělení je obecně vícekriteriální optimalizační problém. Optimální algoritmus by měl rovnoměrně rozložit výpočetní zátěž a minimalizovat vzájemnou komunikaci.

Pro ilustraci potenciálu paralelních adaptivních výpočtů metodou konečných prvků uvádí tato práce dva příklady adaptivních nelineárních výpočtů tahového porušení betonu na paralelních počítačích: dvourozměrnou analýzu brazilského testu v příčném tahu a trojrozměrnou analýzu vytahování kotvy.

Keywords: adaptive analysis, parallel processing, dynamic load balancing, finite element method, nonlinear fracture mechanics

Klíčová slova: adaptivní analýza, paralelní výpočty, dynamické vyvažování výpočetní zátěže, metoda konečných prvků, nelineární lomová mechanika

Contents

1 Introduction

1.1 Motivation and approaches in parallel adaptive FEM simulations

The finite element method has become the most powerful tool for structural analysis. During the last decades, the method has matured to such a state that it is massively used in practical engineering for the solution of broad range of problems starting from linear elasticity up to highly nonlinear transient simulations of the behavior of real materials and complex structures. However, the quality of the obtained solution is dependent on many aspects (including the adopted spatial and time discretization, material model, equation solver and its parameters, etc.). Thus it is very important to keep the solution error under control. This can be conveniently (and usually also most economically) accomplished by the application of the adaptive analysis.

A very natural goal of the adaptive finite element analysis is to calculate solution of the governing partial differential equation(s) with uniformly distributed error not exceeding a prescribed threshold in the most economical manner. This is achieved by improving the discretization in areas where the finite element solution is not adequate. It is therefore essential to have an assessment of the quality of the approximate solution and a capability of discretization enrichment. For linear problems, error analysis of the finite element solution can be developed in a mathematically rigorous way [2, 37]. In the nonlinear range, however, rigorous error estimates can be constructed only for a restricted class of problems. A general theory is not available since there are various sources and forms of nonlinearities. In the linear elastic case the error arises essentially from the discretization of the domain (so-called spatial error). In the nonlinear case, the error depends on the time discretization for history-dependent solids, and a part of the error is always induced by the incremental-iterative technique. The path dependency renders the problem more complex and, consequently, a reliable error estimation becomes more difficult, especially for non-conventional theories of enriched continua. Nevertheless, considerable progress has been made in recent years. For instance, Rodriguez-Ferran and Huerta [29] proposed a sophisticated error estimator for nonlocal damage models. Ladeveze and coworkers [21, 16] developed a posteriori estimators based on the error in the constitutive relation, and Comi and Perego [9] adapted this technique to their nonlocal damage theory [8]. However, the implementation of these complicated estimators requires

a considerable effort. A simple and convenient alternative to rigorous error estimators is provided by heuristic error indicators (see [25], for example). They are often based on physical intuition and insight into the problem at hand.

There are three main directions of the adaptive discretization enrichment. The first one, a natural way for most engineers, is the h-version [5, 10, 12], which refines the computational finite element mesh while preserving the approximation order of the elements. The p-version [34] keeps the mesh fixed but increases hierarchically the order of the approximation being used. The hp-version [10, 22, 38] is a proper combination of h- and p-versions and exhibits an exponential convergence rate independently of the smoothness of the solution. However, its implementation is not trivial. Similarly, the treatment of higher order elements in the p-version is rather complicated, especially when nonlinear analysis is considered.

The application of adaptivity paradigm can result in computationally very demanding analysis in terms of both computational time and computer resources (memory, disk space, etc.). These demands can be alleviated by performing the analysis in a parallel computing environment. Typical parallel application decreases the demands on memory and other resources by spreading the task over several mutually interconnected computers and speeds up the response of the application by distributing the computation to individual processors.

The design of parallel algorithms requires the partitioning of the problem into a set of tasks, the number of which is greater than or equal to the number of processors. The partitioning of the problem can be either fixed (static load balancing) or can change during the solution (dynamic load balancing). The latter option is often necessary in order to achieve good load balancing of individual processors and consequently also reasonable scalability.

As the spatial distribution of error prior the remeshing is usually not uniform, the number of elements can vary significantly between partitions after a few remeshing steps. This is particularly true for problems with strong localization, where error is localized into narrow bands, which leads to enormous refinement in these regions. Thus, the load rebalancing on the fly is necessary to recover the load balance and to obtain scalable implementation. The load balance recovery is achieved by repartitioning the problem domain and by transferring the work (represented typically by finite elements) from one subdomain to another.

1.2 Introduction to parallel computing

Traditional computer codes run on a computer with a single processing unit. The algorithm is translated into a sequence of instructions, which are executed by a processing unit one by one. On the other hand, parallel computing is based on premise, that large problems can be divided into smaller tasks, which are solved concurrently by the simultaneous use of multiple computing resources. Parallelization of the problem reduces the computational time, and, for some cases, it allows large analyzes to be at least performed.

The existing computer architectures can be classified using Flynn's Taxonomy [14] into following four classes:

- Single Instruction, Single Data (SISD) architecture, representing a computer with single processing unit, capable to perform a single instruction on a single data in one clock cycle.
- Single Instruction, Multiple Data (SIMD). This represents a type of parallel computer, where all processing units perform the same instruction, but on different data. Typical example is a vector processor implementing an instruction set containing instructions that operate on one-dimensional arrays of data. Vector processing techniques are also quite often used in modern computer graphical cards.
- Multiple Instruction, Single Data (MISD) concept, where every processing unit may execute a different instruction, but all units operate on the same data.
- Multiple Instruction, Multiple Data (MIMD) architecture, where every processing unit may execute a different instruction sequence operating on a different data set. This is the most common type of parallel computer.

In terms of memory distribution, the parallel computers can be divided into shared, distributed, and hybrid memory systems [4], see Fig. 1. In the shared memory system, the main memory with a single address space is shared between all processing units which can directly address and access the same logical memory. Any modification in a memory location caused by one processor is visible to all other processors. The globally accessible memory is the main advantage, as can facilitate the programming to the great extent. On the other hand, the link between memory and processing units is not scalable for increasing number of processing units. Distributed memory refers to the fact

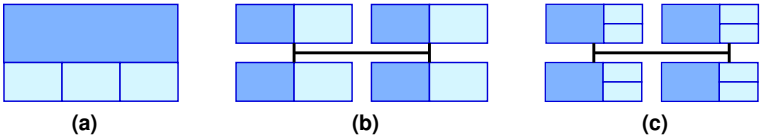

Figure 1: Parallel Computer Memory Architectures: (a) shared memory, (b) distributed memory, and (c) hybrid architecture. Dark rectangle represents memory, light rectangle processing unit, and line communication network.

that the memory is physically distributed. Processors have their own local memory, which does not map to another processor, so there is no concept of global address space across all processors. When a processor needs to access data in another processor, it is usually the task of the programmer to explicitly define how and when data is communicated. The cost of communication compared to local memory access can be very high. On the other hand, the advantage is that the overall memory is scalable with processors. A Typical hybrid system consists of several shared memory units forming large-scale distributed memory system. This type of system seems to be the most promising architecture for future systems.

Despite the variety of available platforms, there exists a common programming model based on message passing paradigm, which is available on most hardware platforms. The message passing is a form of communication based on sending and receiving messages. Nowadays, the Message Passing Interface (MPI) [15, 33] has become de facto standard. MPI libraries are highly portable and are available on virtually all parallel computing platforms, from shared memory systems to distributed workstation clusters. These factors, together with a relatively easy to learn interface are the main factors of MPI success; many scientific libraries like PETSc [3], ScaLAPACK [6] employ MPI for the message-passing communication.

The design of parallel algorithms requires the partitioning of the problem into a set of tasks which are assigned to individual processors. In general, one can distinguish two basic approaches: functional and domain decomposition. The former uses decomposition based on the work that must be performed, so that each task represents a portion of the overall work. In the latter, the focus is on data set which is decomposed. In this approach, each processor works on a different portion of the overall data. Domain decomposition is often used in finite

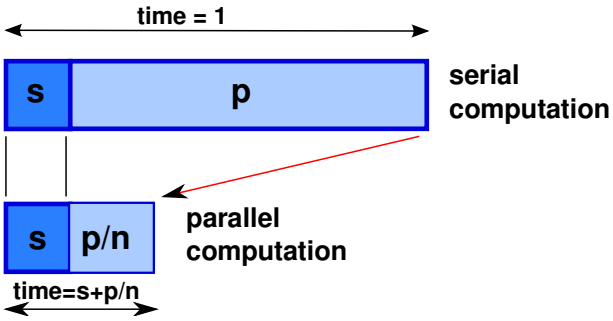

Figure 2: Amdahl's law.

element simulations, where the computational mesh is decomposed into a set of non-overlapping partitions, assigned to individual processors.

The increased performance in terms of time is one of the most important and expected goals of parallel computing. The efficiency of a parallel algorithm is often measured by a speedup which is defined as fraction of the solution time required on a single (serial) processor and the solution time of a parallel task. Optimally, one would expect linear speedup, where doubling the processing units should halve the computational time. However, the optimal speedup is hard to achieve, due to many reasons: the parallelization usually induces costly communication and synchronization, potentially some parts of the algorithm could not be partitioned, etc. One possibility to asses the potential speedup is the Amdahl's law, originally formulated by Gene Amdahl [1] (see also Fig. 2).

$$S(n) = \frac{T(1)}{T(n)} = \frac{s+p}{s+p/n} = \frac{1}{s+p/n} \tag{1}$$

where S is the speed-up of the program, n is a number of processors, p is a parallel fraction of the program, and $s = 1 - p$ is a serial (non paralellizable) fraction of the program. The time spent by the parallel fraction decreases with number of processors, while the time needed by serial fraction remains constant. For increasing number of processors the serial part will limit the achieved speedup and computation cannot scale to match availability of computing power as the machine size increases. For example, if the serial fraction of an algorithm is 5% ($s = 0.05$), one can't achieve better speedup than 20, regardless the number of available processors. The Amdahl's law predictions are not very

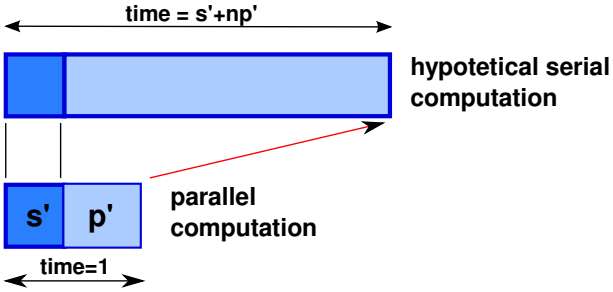

Figure 3: Gustafson's law.

optimistic and have caused many frustrations. The key assumptions behind Amdahl's law are fixed problem size and size of the sequential section independent of the number of processors. The possible remedy has been proposed by Gustafson's law [17] which does not make these assumptions. Today, it is clear that for many problems the Amdahl's assumption of fixed problem size does not apply. In many problems, the model resolution and thus the amount of work required can scale with the number of available processors. From a particular parallel simulation, one can construct a hypothetical serial computation (see Fig. 3) and compute speedup as

$$S(n) = \frac{T(1)}{T(n)} = \frac{s' + np'}{s' + p'} = s' + np' \tag{2}$$

where n is the number of processors, p' is the parallel fraction of the model, and s' is the serial fraction. One can see, that with increasing problem size, the speedup obtained through parallelisation increases, because the parallel work scales with problem size. In some sense, the Gustafson's law has given a new optimism to the community, as the Amdahl's predictions were quite pessimistic. The Amdahl's and Gustafson's laws are compared in Fig. 4.

2 Parallelization Concept

The adopted parallelization strategy is based on domain decomposition concept, in which the mesh is decomposed into a set of non-overlapping sub-domains, which are assigned to individual processors. In general, one can distinguish between two dual partitioning approaches. With

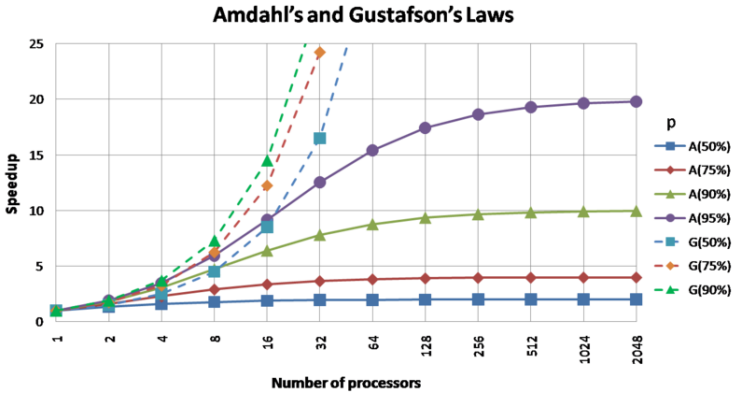

Figure 4: Comparison of Amdahl's and Gustafson's laws.

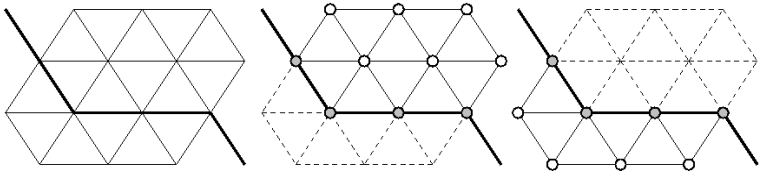

Figure 5: Node cut partitioning.

respect to the character of a cut, dividing the problem mesh into partitions, one can distinguish between node-cut and element-cut strategies [20]. In the node-cut strategy (see Figure 5), the elements are uniquely assigned to individual partitions by leading the cut along element sides. Nodes on these element sides are shared by two or more adjacent partitions and are called shared nodes. In the element-cut strategy (see Figure 6), the nodes are uniquely assigned to individual partitions by running the cut across elements.

It is clear that the partitioning of the problem requires modification of standard sequential code in terms of additional data exchange. For example, in the node cut approach, contributions from individual partitions to shared nodes must be assembled. This data exchange can be done quite efficiently because for a given mesh and its partitioning the individual communication maps, setuped initially on each partition, do not change, unless the problem is subjected to repartitioning

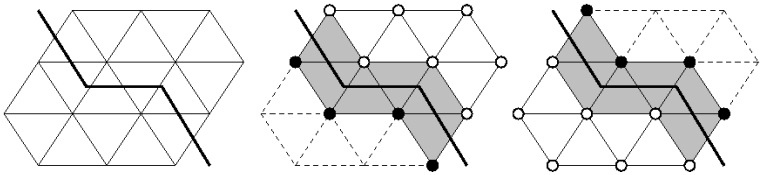

Figure 6: Element cut partitioning.

to recover the load balance and/or mesh refinement. In such a case, the communication maps have to be appropriately updated. From the computational point of view, the node-cut strategy is usually considered more efficient due to the smaller amount of interprocessor data transfer and because the duplicated processing of shared elements is avoided. Therefore, only the node-cut strategy is considered thereafter.

The partitioning is an optimization problem with multiple constraints. To be optimally load-balanced, the partitioning should take into account a number of factors. Firstly, the computational work on each subdomain should be balanced, so that no processor will be waiting for others to complete. The computational work is typically related to elements, and can be, for example, computed as a sum of computational weights of individual elements, which expresses the amount of numerical work (number of operations) associated with each element compared to the selected reference element. Secondly, in order to reduce the communication among partitions, the interface between the partitions should be minimal. Other constraints can reflect the differences in the processing power of individual processors or may be induced by the topology of the network. See Fig. 7 illustrating the domain decomposition of real engineering structure.

The graph partitioning problem is generally NP-complete. However, in recent years a lot of attention has been focused on developing suitable heuristics and many powerful mesh (re)partitioning algorithms which can facilitate this task have been developed. Algorithms based on spectral methods [27, 32] have been shown to be reasonably efficient for partitioning of unstructured problems in many applications, but they have a relatively high computational complexity. Geometric partition methods [18, 13] are quite fast but they typically provide worse partitioning than other (more expensive) methods. Recently, a number of researchers have investigated a class of multilevel graph partitioning algorithms that have a moderate computational complexity

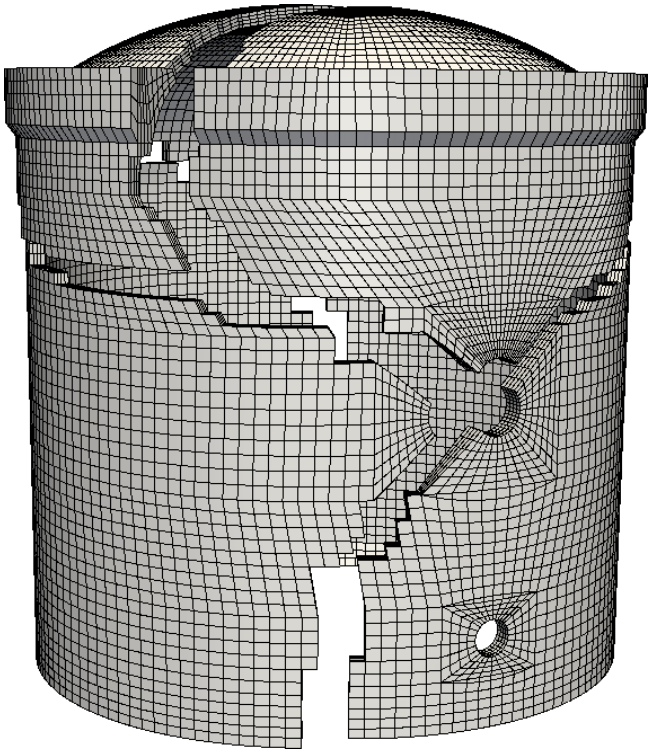

Figure 7: Domain decomposition of nucler power plant containment. Partitions artificially shifted.

and provide excellent (even better than spectral) partitioning [19, 36].

3 Adaptivity

3.1 General concept

The traditional nonlinear finite element analysis is typically based on a fixed mesh and interpolation order, where both the mesh density as well as interpolation order are usually controlled by user experience. The adaptive analysis, on the other hand, allows to control solution error by automatically refining or coarsening the mesh (h-adaptivity), adjusting the interpolation order (p-adaptivity), or both (hp-adaptivity).

The solution error is usually evaluated using suitable a posteriori error estimation.

The adaptive nonlinear analysis consists of several steps. The computation starts with initial discretization with usual incremental solution procedure. After reaching the equilibrium state, an a posteriori error estimation is performed, in order to evaluate the error distribution. If the obtained error level is still acceptable, the analysis continues with the next load increment on the existing mesh. If the evaluated error exceeds a limit defined by the user, the required mesh density is determined from the error distribution and a new spatial discretization is generated. After that the primary unknowns (displacements) and state variables are transferred from the old to the new mesh. The type of internal variables transferred depends on the material model used. For instance, for the isotropic damage model, adopted in the presented study, it is sufficient to transfer the damage variable (or, alternatively, the maximum equivalent strain). After the mapping, the internal variables together with the strains computed from the mapped displacement field are used to update the internal state of each integration point (to ensure local consistency). Once the transfer is finished, the old mesh is deleted and the mapped configuration is brought into global equilibrium by iteration at the last achieved value of the loading parameter. Afterward, the solution continues with the next load increment.

3.2 Parallel adaptivity

The parallel implementation of the adaptive strategy brings in an additional level of complexity. In order to obtain scalable implementation, it is necessary to parallelize all steps involved. Particularly, the parallel mesh generation is quite chalenging. The new mesh should be generated in parallel and partitioned as close as possible to the old mesh partitioning in order to make the transfer of variables local as much as possible within a given partition to avoid costly communication. In general, the newly generated mesh and its partitioning will not exactly respect interpartition boundaries of the old mesh, which implies the need for the nonlocal (remote) data access. This fact complicates the transfer because it requires the adaptation of existing local operators and induces additional interpartition communication.

The spatially optimal overlap of the new and old partitions is important as many transfer operators are based on local projection so that the remote data access can be completely avoided when the new mesh

partitioning coincides exactly with the old one. This fact is the main motivation for the implementation of locality preserving subdivision based remeshing, that allows to profit from local transfer on each partition. However, keeping the old and new partitions coinciding can result in dramatic load imbalance between partitions after a few remeshing steps. This is especially true for problems with strong localization, where error is localized into narrow bands that are in turn subjected to the most progressive refinement. To make the implementation scalable, a load balance recovery is necessary. The error evaluation and possible mesh adaptation can be performed before or after the load balance recovery. The first approach requires more communication (balancing is done on refined grid, so more data need to be moved), but can profit from the fact that load balance is recovered on the adapted grid which will be used in subsequent solution steps. The latter approach is less demanding in terms of communication (provided that the mesh will be refined), but the load balance recovery is based on information from the old mesh, so it may be less optimal, compared to the first approach. In the examples presented in this paper, both approaches are compared.

The above mentioned remeshing approach is based on a subdivision approach using the longest edge bisection algorithm [28] designed for triangular meshes. This approach does not generally require an interaction with an external meshing package and can be performed directly within the analysis tool, which eliminates overhead related to spawning the external meshing. On the other hand, within the analysis tool, only limited (if even any) interaction with the original geometrical model is available, which may complicate handling of curved boundaries and specification of boundary conditions on the refined mesh.

The remeshing itself is performed in multiple loops, the number of which depends on the required refinement level (rate) evaluated from the local error estimate. The elements designated for the refinement are symbolically bisected by introducing a new node on the longest edge. The symbolic bisection propagates to neighbor element sharing the longest edges. During the traversal, on each processed element, the longest edge is again bisected. The propagation stops, when the longest edge is the longest also for the neighbor element (see Fig. 8). This bisection propagation is proved to be finite and usually reasonably localized. After the symbolic bisection is completed the actual element subdivision takes place. The longest edge bisection will ensure that the quality of the refined mesh will not significantly deteriorate even if the bisection is performed repeatedly.

The remeshing strategy as described above is valid for a serial com-

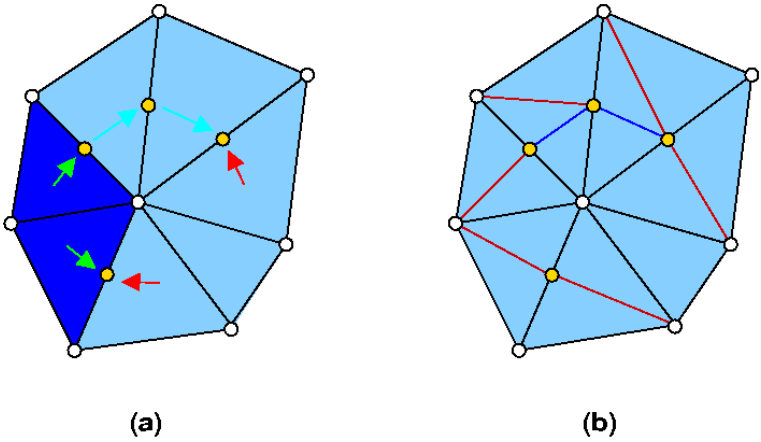

Figure 8: Mesh refinement using symbolic bisection: (a) initial mesh (darker triangles are designated for refinement) with midside nodes (full circles) introduced during symbolic bisection, (b) refined mesh after subdivision of initial mesh.

putation. When considering distributed parallel implementation, one has to take into account that the symbolic bisection may propagate across shared boundaries of individual subdomains. It is therefore necessary for each newly created midside node to identify whether it is located on the shared boundary of local subdomain. If it is this case, this information has to be passed to neighboring partition to further propagate the bisection. The complete parallel algorithm suitable for implementation, as well as its extension into 3D has been described in [26].

When the subdivision is completed, the final mesh may be optionally smoothed using Laplacian smoothing. The smoothing can be generally performed in parallel. However, if also shared interpartition nodes are to be smoothed, some additional communication between the processors is required. In the current implementation, only the non-shared nodes may be subjected to the smoothing, which implies that the smoothing is performed in parallel but only locally on each partition without any communication. As a consequence, the overall shape of individual partitions remain unchanged during the remeshing. This is an important prerequisite for efficient remapping of the state from the old mesh to the new one. Since these meshes are mutually and en-

tirely covering each other, the remapping is performed also in parallel but only locally on each partition without exchange of any data.

4 Dynamic Load Balancing

The load balancing is a process of recovering the load balance in terms of work redistribution among participating processors. There are in general two basic reasons causing a load imbalance between individual subdomains: (i) problem and application related reasons, such as switching from linear to nonlinear response followed by increased computational demands in certain regions or local adaptive refinement, and (ii) external factors, caused by resource reallocation, typical in nondedicated cluster environments, where individual processors are shared by different applications and users, leading to a variation in the allocated processing power.

Once the imbalance has been detected, the amount of work (typically attributed to elements) that should be migrated between individual partitions to recover the load balance has to be determined. This task itself is relatively complicated because it is an optimization problem constrained by many factors. These include requirements for minimal work transfer, minimal interface size that has direct influence on the amount of interpartition data exchange, for example. After the elements to be migrated on each partition have been identified, the work transfer is performed, followed by the update of the internal data structure to reflect the changed topology.

The work transfer involves not only the migrating elements themselves, but also data needed to maintain the overall consistency. The load balancing typically consists of following tasks (described in next subsections): (i) load monitoring, keeping track of the solution process and detecting load imbalance, (ii) load rebalancing, responsible for determining the work redistribution to recover load balance, and (iii) load transfer itself, implementing physical work migration. Several advanced techniques have been developed, which monitor the memory, network, and CPU utilization and availability statistics [11, 35] from which imbalance can be detected and individual processor weights for repartitioning algorithm can be derived.

4.1 Load Monitoring

The load monitoring is responsible for detecting a potential imbalance between participating processors. During execution, the processor

loads are continuously monitored, the eventual imbalance is detected and the decision whether to initiate redistribution of work is made. This decision is derived from the relative and absolute imbalance of individual processors and from an estimate (prediction) whether the cost of load balancing does not exceed the cost of continuing the computation on existing partitioning with load imbalance.

In the present study, a simple implementation of a load monitor, based on run-time measurements of the wall-clock time consumed by computational tasks on individual processors, is adopted. The differences in wall-clock times2 recorded on individual CPUs are considered as imbalance and the updated processing powers (weights) are made proportional to the number of processed equivalent elements3 per unit time. An alternative technique is based on measuring the "wait" times of the processors involved in the computation. These "wait" times measure how long each CPU remains idle while all other processors work on the same task.

4.2 Load Balancing

Work transfer calculation determines the amount of work that should be ideally transferred from one computer to another. The work transfer matrix is not unique, because generally an infinite number of work transfer matrices can satisfy the load balance. Additional criteria are necessary to calculate the work transfer matrix. For example, the minimum of the total transferred work can be required, preferably between neighboring processors, in order to preserve locality.

The traditional partitioning is limited in the sense that it assumes only a single quantity to be load balanced (see [31] for a survey of parallel static and dynamic single-constraint partitioning algorithms). However, many important types of multi-phase simulations require that multiple quantities are load-balanced simultaneously. This is because synchronization steps exist between the different phases of the computations, and so each phase should be individually load balanced.

In order to minimize the work transfer between the old and new partitions, it is desirable to reuse the existing partitioning as much as possible, which is often referred to as an adaptive repartitioning. Since the discretizations used in large-scale simulations are often too large to fit in the memory of a single processor, the repartitioning should be made in parallel. A parallel (re)partitioner can take advantage of the increased memory capacity of parallel machines and improves the overall performance. Several parallel load (re)balancing libraries have

been developed in recent years, notably ParMETIS (adopted in present work), by Karypis et all [30], JOSTLE, by Walshaw et all [36], and Zoltan [7], developed at Sandia National Labs, that provides also data management services.

4.3 Load Transfer

The load transfer phase is responsible for data migration. First, the data to be sent to individual remote partitions have to be identified, followed by their serialization into a communication buffer and initialization of the send operation. This is followed by receiving migrated data from remote partitions and by the final update of the internal data structure.

The individual FEM components on each partition have typically local numbering, which allows using a fast direct array-based access to each component during the solution phase. For similar reasons, a local numbering of equations exists on each partition. However, the data migration requires a unique identification of individual components and unknowns. Therefore, the global numbering of individual components and unknowns is introduced. During packing and unpacking operations the mapping from local to global numbering (and vice versa) is needed. A typical example is the element migration to a remote partition, where the information about its geometry (node numbers) has to be packed as well, so the local node numbers have to be mapped to corresponding global ones before sending.

5 Examples

To illustrate the capabilities and performance of the parallel adaptive load-balancing framework, the results of two analyzes of nonlinear fracture mechanics problems are presented. In both examples, h-adaptive analysis has been used together with heuristic error indicator based on attained damage level. The PETSc [3] (solution of the linearized system) and ParMETIS [19] ((re)partitioning) libraries have been used. Both examples have been computed using the OOFEM solver [24, 23], developed by the author.

In order to assess the behavior and performance of the proposed methodology, the case study analyzes were run without the dynamic load balancing (static partitioning was employed, marked as “nolb”) and with dynamic load balancing performed before (“prelb”) or after (“postlb”) the error assessment.

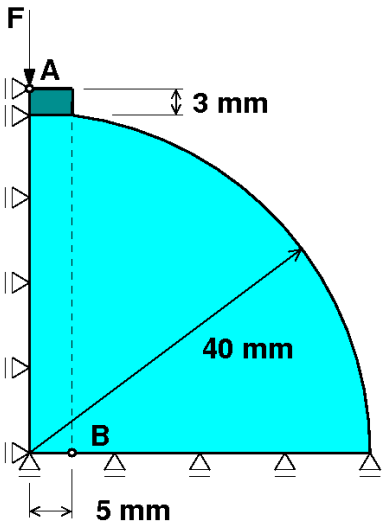

Figure 9: Brazilian test.

5.1 Brazilian splitting test

This test is a standard technique for determination of the tensile strength of concrete. A cylindrical specimen is loaded along its vertical diametral plane. The compressive load, transferred to the specimen via steel bearing plates at the top and bottom sides, induces tension stress in the horizontal direction leading finally to the rupture of the specimen along the loading plane. Due to the double symmetry, the analysis itself is performed only on the quarter of the specimen under plain strain conditions, see Fig. 9. The concrete behavior is described by the nonlocal scalar damage model, while the steel bearing plates are assumed to be linearly elastic. The nonlinear problem was solved incrementally in 40 time steps. During the solution the initial coarse mesh consisting of 220 elements was gradually refined up to 8763 elements.

The problem has been solved on workstation cluster composed of office-based PCs (Dell Optiplex) with single core CPUs at 3.4 GHz, interconnected by gigabit ethernet and on SGI Altix 16-node machine with dual core CPUs running at 1.3 and 1.5 GHz, interconnected by the NUMAflex architecture (NUMAlink 3 with 3.2 GB/s and NUMAlink 4 with 6.4 GB/s bidirectionally). Parallel nonlinear adaptive simulations have been performed using 2, 4, 6, and 8 processors. In each run with

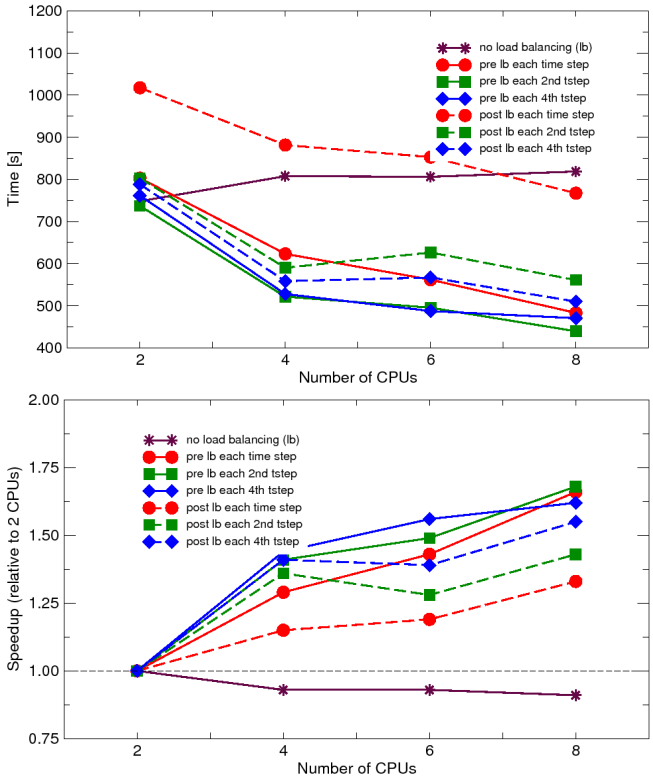

Figure 10: Brazilian test: solution times and speedups on workstation cluster.

the dynamic load balancing, the balancing was enforced either at each time step or at each second or fourth time step. The obtained solution times (averaged over two or three analysis runs) and corresponding speedups (relative to 2 CPUs) are summarized in Figures 10 and 11.

The results on cluster reveal quite poor scalability. On the other hand, the effect of the dynamic load balancing is quite apparent. When no load balancing is applied the solution times are slightly increasing with the number of CPUs, which is the direct consequence of heavy imbalance due to the localized refinement (five levels of subdivision in the fracture process zone were necessary) resulting in dramatic increase of number of elements in one or a few subdomains. With the dynamic load balancing, this effect is alleviated, which results in some speedup.

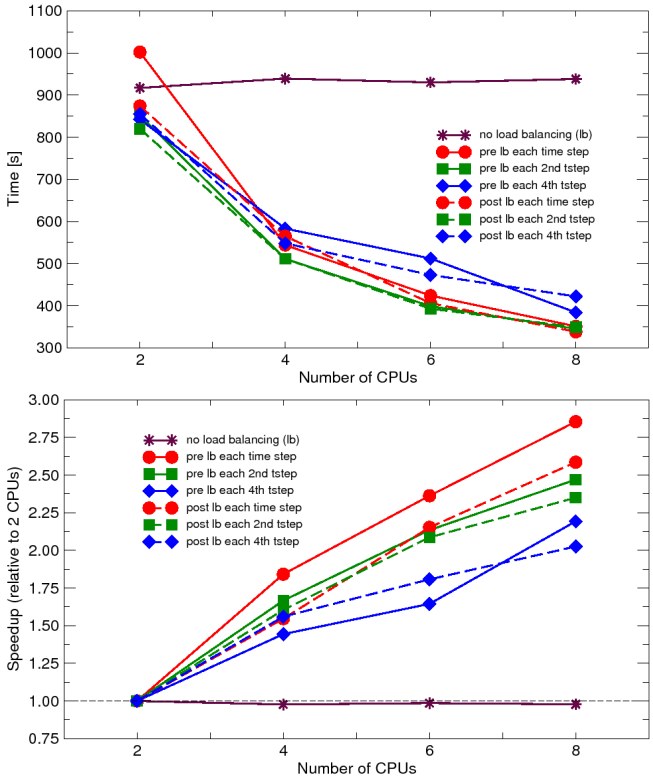

Figure 11: Brazilian test: solution times and speedups on SGI Altix.

The dynamic load balancing applied before the adaptive remeshing performs generally better than that performed after the remeshing. This indicates that the cost of the rebalancing on the adaptively refined mesh is relatively large compared to the gain that the next time step is solved on the balanced refined mesh. Note, that for such a small-scale analysis the communication cost (further increased by nonlocality of the used material model) is clearly the dominating factor. The same analysis on SGI Altix reveals much better scalability and attained speedups, caused by much faster communication on this platform, compared to the workstation cluster.

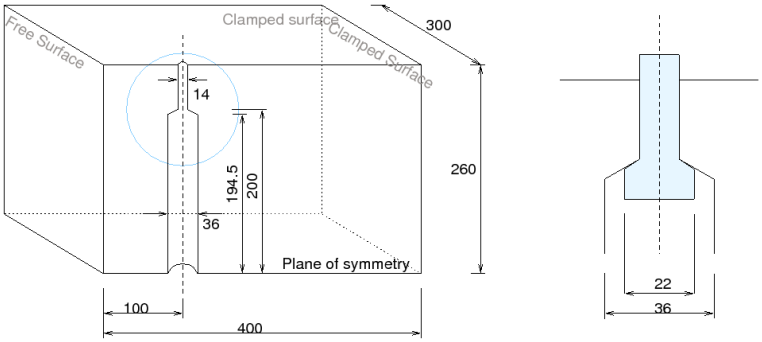

Figure 12: Anchor pullout test: geometry

5.2 Anchor pullout

The geometry and setup of the test are shown in Fig. 12. The anchor is located close to the boundary, requiring full 3D analysis with only one plane of symmetry. As the steel anchor is pulled out of concrete, the crack surface is initiated at the anchor head and starts to propagate towards the boundary as the loading increases. To model concrete fracture, an anisotropic, non-local damage based model has been used. The original mesh consists of 16772 linear tetrahedral elements and 1456 nodes, which was subsequently refined in 20 steps into a final mesh with 125400 elements and 22441 nodes. For the solution of the linearized system, PETSc library has been used (sparse, coordinate based storage scheme (compressed row format), conjugate gradient iterative solver with incomplete Cholesky preconditioner).

The problem has been solved on SGI Altix 4700 machine installed at CTU computing centre. Similarly to previous example, to assess the behaviour and performance of the proposed methodology, the case study analyzes were run without the dynamic load balancing (static partitioning was employed) and with dynamic load balancing performed before or after the error assessment. The obtained solution times (averaged over two or three analysis runs) and corresponding speedups (relative to 4 CPUs) are summarized in Figure 13.

The achieved results reveal that the effect of the dynamic load balancing is quite substantial. When no load balancing is applied the solution times are decreasing with the number of CPUs only slightly, which is the direct consequence of heavy imbalance due to the localized refinement resulting in dramatic increase of number of elements in one

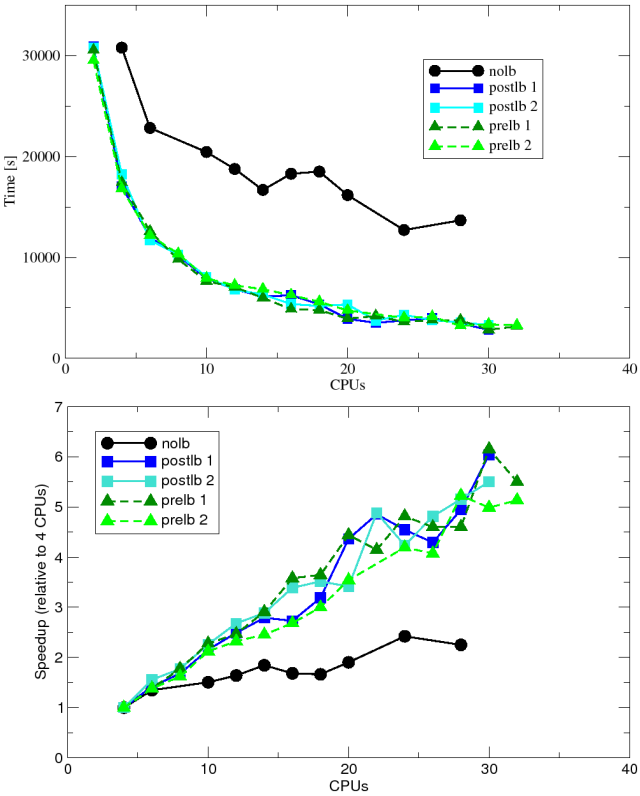

Figure 13: Anchor pullout test: solution times and speedups .

or a few subdomains. With the dynamic load balancing, this effect is alleviated. The obtained speedup of the load-balanced computation is showing clear linear trend, indicating a very good scalability of the parallel agorithm. Also, the absolute values of attained speedups are very good.

6 Conclusions

This work presents a glimpse into our recent work on the development of finite element framework for parallel adaptive load balanced computations. It begins with the introduction into the parallel computing,

outlining the existing parallel computer architectures and approaches to parallel algorithm design. The parallelization concept, based on domain decomposition is presented. The approaches to domain partitioning are outlined, together with discussion on concepts of static and dynamic load balancing. In the next part, a general framework for parallel adaptive finite element computations is described and consequently discussed.

Particular attention is given to dynamic load balancing, as the continuous workload recovery is often necessary for optimal use of available resources and achieving reasonable parallel scalability. Finally, the potential and performance of parallel adaptive approach is demonstrated and discussed on two examples of nonlinear fracture analysis of concrete specimens. The presented examples clearly demonstrate that the dynamic load balancing is an essential part of any parallel adaptive simulation.

Acknowledgments

The financial support provided by the Ministry of Education of the Czech Republic - Project MSM 6840770003 and the Czech Science Foundation - grant no. 103/06/1845 is gratefully acknowledged. The computer resources on SGI Altix have been provided by CTU computing center. The author also thanks for collaboration to other members of research team, in particular to Daniel Rypl, Milan Jirásek, and Zdeněk Bittnar.

References

[1] G.M. Amdahl. Validity of the single-processor approach to achieving large scale computing capabilities. In *AFIPS Conference Proceedings*, volume 30, pages 483–485, Atlantic City, N.J., 1967. AFIPS Press, Reston.

[2] I. Babuška and W.C. Rheinboldt. A posteriori errors estimates for the finite element method. *International Journal for Numerical Methods in Engineering*, 12:1597–1615, 1978.

[3] S. Balay, K. Buschelman, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, B.F. Smith, and H. Zhang. Petsc web page, 2001. http://www.mcs.anl.gov/petsc.

[4] B. Barney. Introduction to parallel computing, 2010. https://computing.llnl.gov/tutorials/parallel_comp/.

[5] T. Belytschko and M. Tabbara. H-adaptive finite element methods for dynamic problems with emphasis on localization. *International Journal for Numerical Methods in Engineering*, 36:4245–4265, 1993.

[6] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. ScaLAPACK users' guide. Technical report.

[7] E. Boman, K. Devine, L.A. Fisk, R. Heaphy, B. Hendrickson, V. Leung, C. Vaughan, U. Catalyurek, D. Bozdag, and W. Mitchell. Zoltan home page, 1999. http://www.cs.sandia.gov/Zoltan.

[8] C. Comi and U. Perego. Numerical aspects of nonlocal damage analysis. *Revue Européeene des Éléments Finis*, 10:227–242, 2001.

[9] C. Comi and U. Perego. Finite element strategies for damage assessment up to failure. In *Proceedings of the Sixth National Congress SIMAI*, Italy, 2002.

[10] A.W. Craig, M. Ainsworth, Z.J. Zhu, and O.C. Zienkiewicz. H and hp version error estimation and adaptive procedures from theory to practice. *Engineering with Computers*, 5:221–234, 1989.

[11] K.D. Devine, E.G. Boman, R.T. Heaphy, B.A. Hendrickson, J.D. Teresco, J. Faik, J.E. Flaherty, and L.G. Gervasio. New challenges in dynamic load balancing. *Applied Numerical Mathematics*, 52(2–3):133–152, 2005.

[12] P. Diez and A. Huerta. A unified approach to remeshing strategies for finite element h-adaptivity. In *Publication CIMNE*, volume 132. Barcelona, 1998.

[13] C. Farhat. A simple and efficient automatic fem domain decomposer. *Computers and Structures*, 28:579–602, 1988.

[14] M. Flynn. Some computer organizations and their effectiveness. *IEEE Trans. Comput.*, C-21, 1972.

[15] Message Passing Interface Forum. MPI: A message-passing interface standard. Technical report, University of Tennessee, 1995.

[16] L. Gallimard, P. Ladeveze, and J.P. Pelle. Error estimation and adaptivity in elasto-plasticity. *International Journal for Numerical Methods in Engineering*, 39:189–217, 1996.

[17] John L. Gustafson. Reevaluating amdahl's law. *Commun. ACM*, 31(5):532–533, 1988.

[18] M.T. Heath and P. Raghavan. A cartesian parallel nested dissection algorithm. *SIAM Journal on Matrix Analysis and Applications*, 16(1):235–253, 1995.

[19] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1999.

[20] P. Krysl and Z. Bittnar. Parallel explicit finite element solid dynamics with domain decomposition and message passing: Dual partitioning scalability. *Computers and Structures*, 79(3):345–360, 2001.

[21] P. Ladeveze and D. Leguillon. Error estimate procedure in the finite element method and application. *SIAM Journal of Numerical Analysis*, 20:485–509, 1983.

[22] J.T. Oden, L. Demkowitz, W. Rachowitz, and T.A. Westermann. Towards a universal h-p adaptive finite element strategy, part 2. a posteriori error estimation. *Computer Methods in Applied Mechanics and Engineering*, 77:113–180, 1989.

[23] B. Patzák. OOFEM home page. http://www.oofem.org, 2000.

[24] B. Patzák and Z. Bittnar. Design of object oriented finite element code. *Advances in Engineering Software*, 32(10-11):759–767, 2001.

[25] B. Patzák and M. Jirásek. Adaptive resolution of localized damage in quasibrittle materials. *Journal of Engineering Mechanics ASCE*, 130:720–732, 2004.

[26] B. Patzák and D. Rypl. Parallel adaptive finite element computations with dynamic load balancing. In L.F. Costa Neves B.H.V. Topping and R.C. Barros, editors, *Proceedings of the Twelfth International Conference on Civil, Structural and Environmental Engineering Computing*, Stirlingshire, Scotland, 2009. Civil-Comp Press.

[27] A. Pothen, H.D. Simon, and K. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal of Matrix Analysis and Applications*, 11(3):430–452, 1990.

[28] M.C. Rivara. Local modification of meshes for adaptive and/or multigrid finite-element methods. *Journal of Computational and Applied Mathematics*, 36:79–89, 1991.

[29] A. Rodriguez-Ferran and A. Huerta. Error estimation and adaptivity for nonlocal damage models. *International Journal of Solids and Structures*, 37:7501–7528, 2000.

[30] K. Schloegel, G. Karypis, and V. Kumar. Parallel static and dynamic multi-constraint graph partitioning. *Concurrency and Computation: Practice and Experience*, 14(3):219–240, 2002.

[31] Kirk Schloegel, George Karypis, and Vipin Kumar. Graph partitioning for high-performance scientific simulations. pages 491–541, 2003.

[32] H.D. Simon. Partitioning of unstructured meshes for parallel processing. *Computing Systems in Engineering*, 2:135–148, 1991.

[33] Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker, and Jack Dongarra. *MPI: The Complete Reference*. MIT Press, Boston, 1996.

[34] B.A. Szabo. Estimation and control of error based on p convergence. In *Accuracy Estimates and Adaptive Refinements in Finite Element Computations*, page 61–70. John Wiley, New York, 1986.

[35] C. Walshaw and M. Cross. Multilevel mesh partitioning for heterogeneous communication networks. *Future Generation Computer Systems*, 17(5):601–623, 2001.

[36] C. Walshaw and M. Cross. Jostle: Parallel multilevel graph-partitioning software — an overview. In F. Magoules, editor, *Mesh Partitioning Techniques and Domain Decomposition Techniques*, page 27–58. Civil-Comp Ltd, 2007.

[37] O.C. Zienkiewicz and J.Z. Zhu. A simple error estimator and adaptive procedure for practical engineering analysis. *International Journal for Numerical Methods in Engineering*, 24:337–357, 1987.

[38] O.C. Zienkiewicz, J.Z. Zhu, and N.G. Gong. Effective and practical h-p version adaptive analysis procedures for the finite element method. *International Journal for Numerical Methods in Engineering*, 28:879–891, 1989.

Bořek Patzák

Curicullum Vitae

Department of Mechanics ∘ Faculty of Civil Engineering ∘ Czech Technical University in Prague ∘ E-mail: borek.patzak@fsv.cvut.cz

Education

- 2002 - Associate Professor (Doc.)
  CTU in Prague, Faculty of Civil Engineering
  Thesis title: Computational Aspects of Nonlocal Material Models
- 1997 - Doctor of Philosophy (Dr.)
  CTU in Prague, Faculty of Civil Engineering
  Thesis title: Materials models for concrete
  Advisor: Prof. Z. Bittnar
- 1993 - Master of Science (Ing.)
  CTU in Prague, Faculty of Civil Engineering
  Thesis title: Semianalytical solution with independent rotation field
  Advisor: Prof. Z. Bittnar
  Finished with honors

Professional qualifications

- (1997 – 2000) **Assistant Professor**, *CTU in Prague, Faculty of Civil Engineering, Department of Mechanics*
- (2000 – 2002) **Research Engineer**, *EPFL, Department of Civil Engineering, Laboratory of Structural and Continuum Mechanics*
- (2002 – present) **Associate Professor**, *CTU in Prague, Faculty of Civil Engineering, Department of Mechanics*

Main scientific interests

- Dynamic load balancing on heterogeneous parallel architectures
- Adaptive techniques and constitutive modeling of quasibrittle materials
- High performance computing and software development - object oriented and parallel programming
- Modeling of fresh concrete casting
- Development of open source finite element toolkit OOFEM (www.oofem.org)

Professional and scientific activities

- Author and co-author of over 50 professional publications, of which 10 are articles in international impact or refereed journals, 3 articles in monographs, and over 30 are articles in international conferences
- 74 citations in SCI-EXP (excluding self-citations), h-index is 5
- Principal investigator of 3 grants (GAČR 103/04/1394, GAČR 103/-06/1845, GAČR 105/10/1402), collaboration on 2 international grants, 13 national grants.
- Member of scientific board of 7 international conferences
- Lead developer of OOFEM finite element code (www.oofem.org)

Teaching activities

- Lectures in Structural Mechanics, Numerical methods in mechanics, Stability theory.
- Advisor of **4** diploma theses
- Advisor of doctoral students
  - Defended doctoral theses: **2**
  - Current number of doctoral students: **2**

Selected journal publications from the past 10 years

[1] B.Patzák and Z.Bittnar. Modeling of fresh concrete flow. Computers and Structures, 87(15-16):962–969, 2009.

[2] Z. Hora and B. Patzák. Analysis of long-term behaviour of nuclear reactor containment. Nuclear Engineering and Design, 273(3):253–259, February 2007.

[3] J. Němeček, P. Padevět, B. Patzák, and Z. Bittnar. Effect of transversal reinforcement in normal and high strength concrete columns. Materials and Structures, 38(281):665–671, 2005. ISSN 1359-5997.

[4] B.Patzák and M.Jirásek. Adaptive resolution of localized damage in quasibrittle materials. Journal of Engineering Mechanics Division ASCE, 130:720–732, 2004.

[5] B.Patzák and M.Jirásek. Process zone resolution by extended finite elements. Engineering Fracture Mechanics, 70(7-8):837–1097, May 2003

[6] M.Jirásek and B.Patzák. Consistent tangent stiffness for nonlocal damage models. Computers and Structures, 80(14-15):1279–1293, June 2002.

[7] B.Patzák, D.Rypl, and Z.Bittnar. Parallel explicit finite element dynamics with nonlocal constitutive models. Computers and Structures, 79(26-28):2287–2297, 2001.

[8] B.Patzák and Z.Bittnar. Design of object oriented finite element code. Advances in Engineering Software, 32(10-11):759–767, 2001.