Solving contact mechanics problems with PERMON

Vaclav Hapla[1,2], David Horak[1,2], Lukas Pospisil[1,2], Martin Cermak[1], Alena Vasatova[1,2], and Radim Sojka[1,2]

[1] IT4Innovations National Supercomputing Center
[2] Department of Applied Mathematics
VSB - Technical University of Ostrava, Czech Republic

Abstract. PERMON makes use of theoretical results in quadratic programming algorithms and domain decomposition methods. It is built on top of the PETSc framework for numerical computations. This paper describes its fundamental packages and shows their applications. We focus here on contact problems of mechanics decomposed by means of a FETI-type non-overlapping domain decomposition method. These problems lead to inequality constrained quadratic programming problems that can be solved by our PermonQP package.

1 Introduction

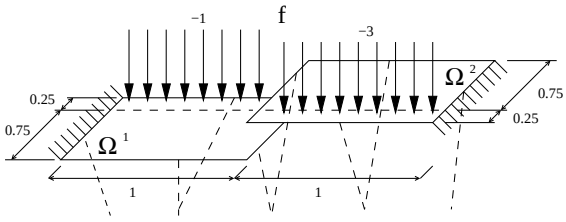
We shall present our new software called PERMON (Parallel, Efficient, Robust, Modular, Object-oriented, Numerical) toolbox [17] and show its capabilities. PERMON extends PETSc [3] with support for quadratic programming (QP) and non-overlapping domain decomposition methods (DDM), namely of the FETI (Finite Element Tearing and Interconnecting) [13,12,5,24] type.

This paper presents the process of solving contact problems using PERMON (Section 3). We consider three model problems: two scalar contact problems of two membranes (coercive and semi-coercive) and a contact problem of a 3D linear elastic cube with an obstacle (Section 2). The mesh is "teared" into subdomains and each of them is discretized separately and sequentially with the FEM (Finite Element Method). This decomposition and discretization is implemented by the PermonMembrane and PermonCube (Section 4) packages. The subdomain problems are then "interconnected" by means of FETI using PermonFLLOP (Section 5). Finally, the resulting QP problem is solved by the PermonQP module (Section 6). It contains implementations of specific algorithms for bound and equality constrained problems, particularly the SMALBE and MPRGP algorithms (Sections 7 and 8).

2 Model contact problems

We consider three model problems depicted in Fig. 1. First two are scalar problems consisting of two membranes in mutual contact at adjacent edges. The solution $u(x, y)$ can be interpreted as a vertical displacement of two membranes stretched by normalized horizontal forces and pressed together by vertical forces with density $f(x, y)$. The inequality constraints result from requiring nonpenetration of the adjacent edges of the membranes, with the edge of the right membrane above the edge of the left membrane and by pressing the left membrane down by the right one at the contact points. The first problem, where the right membrane has its right edge fixed, is coercive

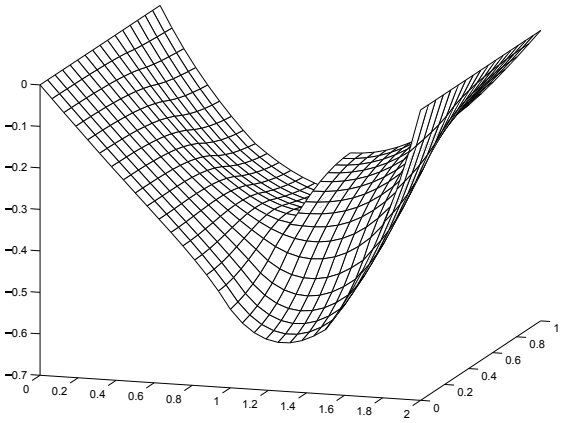
The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-40361-8_7.



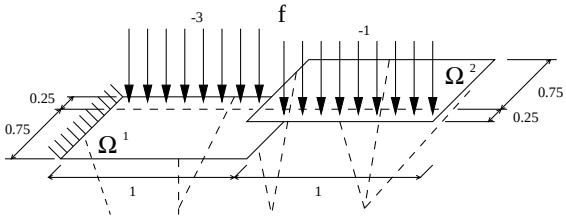
$$\Gamma_u^1 \quad \Gamma_f^1 \quad \Gamma_c \quad \Gamma_f^2 \quad \Gamma_u^2$$

$$f(x,y) = \begin{cases} -1 & \text{for} \quad (x,y) \in (0,1) \times [0.75,1) \\ 0 & \text{for} \quad (x,y) \in (0,1) \times [0,0.75) \\ & \text{and} \quad (x,y) \in (1,2) \times [0.25,1) \\ -3 & \text{for} \quad (x,y) \in (1,2) \times [0,0.25) \end{cases}$$

(a)



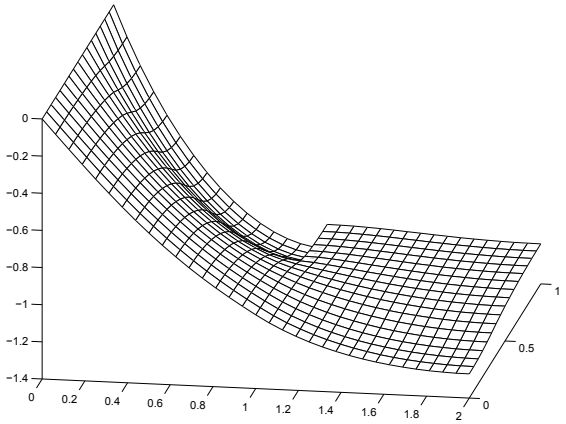
(b)



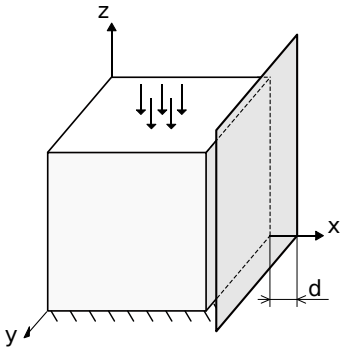
$$\Gamma_u^1 \quad \Gamma_f^1 \quad \Gamma_c \quad \Gamma_f^2$$

$$f(x,y) = \begin{cases} -3 & \text{for} \quad (x,y) \in (0,1) \times [0.75,1) \\ 0 & \text{for} \quad (x,y) \in (0,1) \times [0,0.75) \\ & \text{and} \quad (x,y) \in (1,2) \times [0.25,1) \\ -1 & \text{for} \quad (x,y) \in (1,2) \times [0,0.25) \end{cases}$$

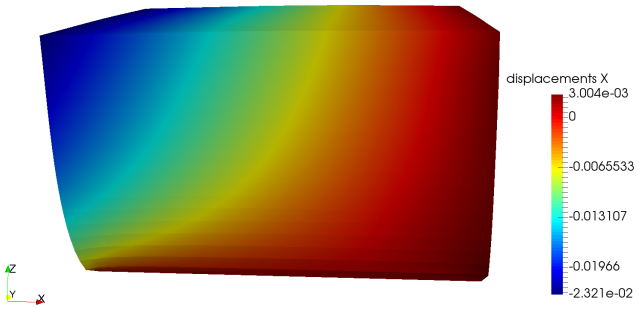
(c)



(d)



(e)



(f)

Fig. 1: Model problems: coercive (1a, 1b) and semicoercive (1c, 1d) scalar contact problem of two membranes, and elastic cube with a rigid obstacle (1e, 1f) – problem specification (left) and solution (resulting displacements, right).

(Fig. 1a, 1b). The second problem is semicoercive since the right membrane is completely floating (Fig. 1c, 1d).

As a model 3D linear elasticity contact problem, we consider an elastic cube with the bottom face fixed, the top one loaded with a vertical surface force directed downwards, and the right one in contact with a rigid obstacle (Fig. 1e, 1f). The loading force density is $f_z = 465$ N/mm^2, Young's modulus $E = 2 \cdot 10^5$ MPa, Poisson's ratio $\mu = 0.33$.

3 Solution process

This paper presents the process of solving contact problems using the PERMON Toolbox (Fig. 2). The body is decomposed in a non-overlapping way using the FETI methodology. FETI methods represent non-overlapping DDMs. They can be divided into three parts: (1) meshing part, (2) assembly part and (3) algebraic part.

Let us describe the first part. Mesh partitioning is performed first, then degrees of freedom (DOFs) on submesh interfaces are uniquely numbered (we call the resulting numbering the "undecomposed numbering"), DOFs on submesh interfaces are copied to each respective submesh, and finally the DOFs are renumbered to restore global uniqueness (resulting in the "decomposed numbering"). The resulting submeshes are completely self-contained, i.e. there is no cell overlapping or ghost-layer.

The second part, FEM assembly of algebraic objects, is performed completely separately for each submesh using any existing (even sequential) FEM code. Stiffness matrix and load vector of each submesh form a "local problem"

The third part is essentially a mathematical approach how to deal with the mesh decomposition described above so that a correct solution, continuous across submesh interfaces, is obtained. Typically, only this final part is actually called a non-overlapping DDM. To "glue" the subdomains together, the solver needs to know a mapping between the duplicated DOFs on neighbouring submesh interfaces. This mapping results from the first part, and is simply a many-to-one mapping from the decomposed numbering to the undecomposed one provided the submesh interfaces conform. This mapping is a minimal additional information needed in contrast to "parallel linear system solvers" that just act on one distributed global stiffness matrix and load vector which come from an "undecomposed" mesh.

The first and second parts can be covered by the PermonCube and PermonMembrane packages (Section 4), while the third one is realized using the PermonFLLOP package (Section 5).

4 PermonMembrane and PermonCube

For rapid development and testing of our solvers, PermonMembrane and PermonCube packages [26] were developed. They implement the first and second parts in a massively parallel way for simple benchmarks generated in runtime. PermonCube is similar by focus to the software package Pamgen [30,19]. Although it provides so far only a cubical mesh, the FEM part of the code does not rely on this specific type of mesh, and works with that as if it were an unstructured mesh, simulating decomposed FEM processing of real world problems. Extending the first part to real world meshes is work in progress. Moreover, we strive to support widely available FEM libraries such as Elmer [27].

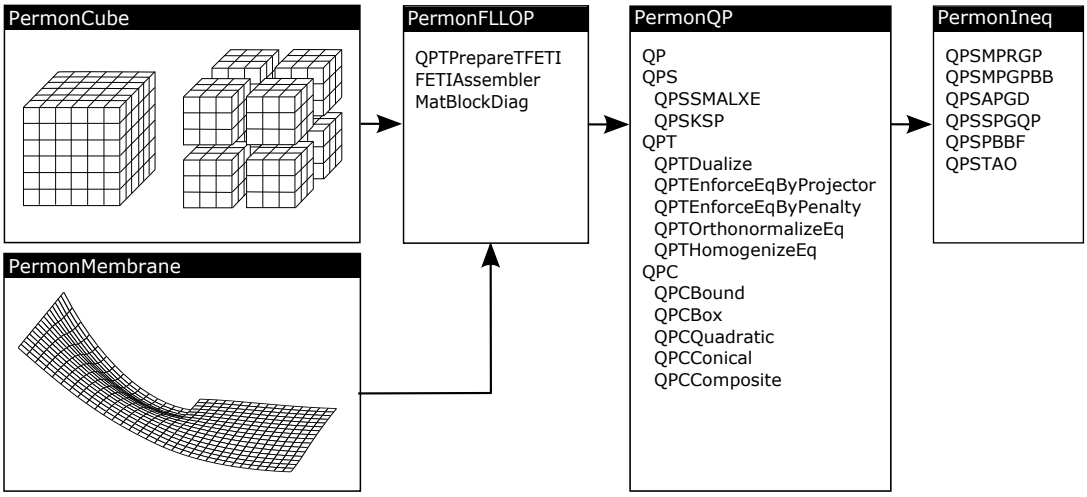


Fig. 2: PERMON life cycle.

The parallel mesh generation is controlled by two groups of parameters. In PermonCube, the number of subdomains is managed by parameters X, Y, Z, and similarly the number of elements per subdomain is given by x, y, z (both considered in the respective axis directions). In PermonMembrane, the situation is similar, only parameters Z and z are missing. The decomposition parameter H and the discretization parameter h is given as $H = \frac{L}{X}$ and $h = \frac{H}{x}$, respectively, where L denotes the size of the whole domain.

Essential data, generated by PermonCube, PermonMembrane or any other FEM software, are the subdomain stiffness matrices $\mathbf{K}^s$ and the subdomain right-hand side vectors $\mathbf{f}^s$, $s = 1, \ldots, N_S$ where N_S denotes the total number of subdomains, $N_S = XYZ$ for PermonCube and $N_S = 2XY$ for PermonMembrane. In the DDM context, an additional object, the previously described local-to-global interface DOF mapping $l2g$, has to be created. These data are passed to PermonFLLOP, described in the next section.

5 PermonFLLOP

Our PermonFLLOP package implements the algebraic part of non-overlapping DDMs of the FETI type. We shall firstly briefly introduce the FETI-1 and Total-FETI (TFETI) methods.

FETI-1 [13,14,12,23] is a non-overlapping DDM [16]. Thus, it is based on decomposing the original spatial domain into non-overlapping subdomains. They are "glued together" by Lagrange multipliers which have to satisfy certain equality constraints, discussed later. The original FETI-1 method assumes that the boundary subdomains inherit Dirichlet conditions from the original problem where the conditions are embedded into the linear system arising from FEM. This means physically that subdomains, whose interfaces intersect the Dirichlet boundary, are fixed while others are kept floating; in the linear algebra speech, the corresponding subdomain stiffness matrices are non-singular and singular, respectively.

The basic idea of the TFETI method [5,7,32] is to keep all the subdomains floating and enforce the Dirichlet boundary conditions by means of a constraint matrix and Lagrange multipliers, sim-

ilarly to the gluing conditions along subdomain interfaces. This simplifies the implementation of the stiffness matrix pseudoinverse. The key point is that kernels of subdomain stiffness matrices are known a priori, have the same dimension and can be formed without any computation from the mesh data. Furthermore, each local stiffness matrix can be regularized cheaply, and the inverse of the resulting nonsingular matrix is a pseudoinverse of the original singular one [6,4].

Let us consider a partitioning of the global domain Ω into N_S subdomains $\Omega^s, s = 1, \ldots, N_S$. To each subdomain Ω^s there corresponds the subdomain stiffness matrix $\mathbf{K}^s$, the subdomain nodal load vector $\mathbf{f}^s$, the matrix $\mathbf{R}^s$ whose columns span the nullspace (kernel) of $\mathbf{K}^s$, and the constraint matrix $\mathbf{B}^s$. The latter consists of the equality and inequality parts, $(\mathbf{B}^s)^T = \left[(\mathbf{B}_E^s)^T \ (\mathbf{B}_I^s)^T \right]$. The equality part is $(\mathbf{B}_E^s)^T = \left[(\mathbf{B}_g^s)^T \ (\mathbf{B}_d^s)^T \right]$, where $\mathbf{B}_g^s$ is a signed Boolean matrix defining connectivity of the subdomain Ω^s with all its neighbouring subdomains, and $\mathbf{B}_d$ is a Boolean matrix describing Dirichlet boundary conditions (empty if the TFETI approach is not used). The inequality part $\mathbf{B}_I^s$ (possibly empty) describes linearized non-penetration conditions [7] of the subdomain Ω^s on the corresponding part of the contact zone. The global constraint right-hand side vector $\mathbf{c}$ and vector of Lagrange multipliers $\boldsymbol{\lambda}$ possess analogous structure.

The local objects $\mathbf{K}^s$, $\mathbf{f}^s$, $\mathbf{R}^s$ and $\mathbf{B}^s$ constitute global objects

$$\mathbf{K} = \mathrm{diag}(\mathbf{K}^1, \ldots, \mathbf{K}^{N_S}), \qquad \mathbf{B}_E = [\mathbf{B}_E^1, \ldots, \mathbf{B}_E^{N_S}], \qquad \mathbf{B} = [\mathbf{B}^1, \ldots, \mathbf{B}^{N_S}] = \left[\begin{smallmatrix} \mathbf{B}_E \\ \mathbf{B}_I \end{smallmatrix} \right],$$
$$\mathbf{R} = \mathrm{diag}(\mathbf{R}^1, \ldots, \mathbf{R}^{N_S}), \qquad \mathbf{B}_I = [\mathbf{B}_I^1, \ldots, \mathbf{B}_I^{N_S}], \qquad \mathbf{f} = [(\mathbf{f}^1)^T, \ldots, (\mathbf{f}^{N_S})^T]^T,$$

where diag means a block-diagonal matrix consisting of the diagonal blocks between parentheses. Note that columns of $\mathbf{R}$ also span the kernel of $\mathbf{K}$. The global discrete form of the contact problem can be written as the primal QP

$$\min \frac{1}{2}\mathbf{u}^T\mathbf{K}\mathbf{u} - \mathbf{u}^T\mathbf{f} \quad \text{s.t.} \quad \mathbf{B}_E\mathbf{u} = \mathbf{c}_E \quad \text{and} \quad \mathbf{B}_I\mathbf{u} \le \mathbf{c}_I \tag{1}$$

with $\mathbf{c}_E$ and $\mathbf{c}_I$ denoting prescribed gaps for equality and inequality constraints. Let us apply the convex QP duality theory and establish the following notation

$$\mathbf{F} = \mathbf{B}\mathbf{K}^\dagger\mathbf{B}^T, \quad \mathbf{G} = \mathbf{R}^T\mathbf{B}^T, \quad \mathbf{d} = \mathbf{B}\mathbf{K}^\dagger\mathbf{f}, \quad \mathbf{e} = \mathbf{R}^T\mathbf{f},$$

where $\mathbf{K}^\dagger$ denotes a pseudoinverse of $\mathbf{K}$, satisfying $\mathbf{K}\mathbf{K}^\dagger\mathbf{K} = \mathbf{K}$. We obtain the dual QP

$$\min \frac{1}{2}\boldsymbol{\lambda}^T\mathbf{F}\boldsymbol{\lambda} - \boldsymbol{\lambda}^T\mathbf{d} \quad \text{s.t.} \quad \mathbf{G}\boldsymbol{\lambda} = \mathbf{e} \quad \text{and} \quad \boldsymbol{\lambda}_I \ge \mathbf{o}. \tag{2}$$

After several manipulations [7] we get the final QP, suitable for numerical solution,

$$\min \frac{1}{2}\boldsymbol{\lambda}^T\mathbf{P}\mathbf{F}\mathbf{P}\boldsymbol{\lambda} - \boldsymbol{\lambda}^T\mathbf{P}\mathbf{d} \quad \text{s.t.} \quad \mathbf{G}\boldsymbol{\lambda} = \mathbf{o} \quad \text{and} \quad \boldsymbol{\lambda}_I \ge -\widetilde{\boldsymbol{\lambda}}, \tag{3}$$

where

$$\mathbf{P} = \mathbf{I} - \mathbf{Q} \quad \text{and} \quad \mathbf{Q} = \mathbf{G}^T(\mathbf{G}\mathbf{G}^T)^{-1}\mathbf{G}$$

denote the orthogonal projectors onto the kernel of $\mathbf{G}$ and image of $\mathbf{G}^T$, respectively, and $\widetilde{\boldsymbol{\lambda}}$ denotes an arbitrary vector satisfying the equality constraints of (2).

Let us show how PermonFLLOP is implemented from the user's perspective. First of all, it takes from the FEM software the subdomain stiffness matrices $\mathbf{K}^s$ and the subdomain load vectors $\mathbf{f}^s$ as sequential data for each subdomain Ω^s, $s = 1, \ldots, N_S$. Note that we assume here each

processor core owns only one subdomain, PermonFLLOP has nevertheless an experimental feature of allowing more than one subdomain per core, i.e. an array of $\mathbf{K}^s$ and $\mathbf{f}^s$ is passed per subdomain. PermonFLLOP enriches the independent subdomain data with the global context so that $\mathbf{K}$ and $\mathbf{f}$ are effectively created from $\mathbf{K}^s$ and $\mathbf{f}^s$, respectively.

The "gluing" signed Boolean matrix $\mathbf{B}_g$ is created based on the local-to-global mapping $l2g$ [31]. The FEM software can skip the processing of the Dirichlet conditions and rather hand it over to PermonFLLOP, resulting in greater flexibility. PermonFLLOP allows to enforce Dirichlet boundary conditions either by the constraint matrix $\mathbf{B}_d$ (TFETI approach), or by a classical technique of embedding them directly into $\mathbf{K}$ and $\mathbf{f}$ (FETI-1 approach). It is also possible to mix these two approaches.

Furthermore, PermonFLLOP assembles the nullspace matrix $\mathbf{R}$ using one of the following options. The first option is to use a numerical approach [16], and the second one is to generate $\mathbf{R}$ as rigid body modes from the mesh nodal coordinates [22]. The latter is typical for TFETI and is considered here.

Currently, PermonFLLOP requires $\mathbf{B}_I$ and $\mathbf{c}_I$ from the caller. We strive to overcome this limitation in the future so that the non-penetration conditions will be specified in a way more natural for engineers. Listing 1.1 shows how a FEM software (such as PermonCube) typically calls PermonFLLOP to solve a decomposed contact problem.

```
Mat Ks,BIs;  Vec fs,cI,coords;  IS l2g,dbcis;  MPI_Comm comm;  FLLOP fllop;

/*  Generate the data. */

/*  Create FLLOP living in communicator comm. */
FllopCreate(comm, &fllop);

/*  Set the subdomain stiffness matrix and load vector. */
FllopSetStiffnessMatrix(fllop, Ks);
FllopSetLoadVector(fllop, fs);

/*  Set the local-to-global mapping for gluing. */
FllopSetLocalToGlobalMapping(fllop, l2g);

/*  Specify the Dirichlet conditions in the local numbering
    and tell FLLOP to enforce them by means of the B matrix. */
FllopAddDirichlet(fllop, dbcis, FETI_LOCAL, FETI_DBC_B);

/*  Set vertex coordinates for rigid body modes. */
FllopSetCoordinates(fllop, coords);

/*  Set the non-penetration inequality constraints. */
FllopSetIneq(fllop, BIs, cI);

FllopSolve(fllop);
```

Listing 1.1: PermonCube calls PermonFLLOP

In the `FllopSolve` function, PermonFLLOP passes the global primal data $\mathbf{K}$, $\mathbf{f}$, $\mathbf{B}$ and $\mathbf{R}$ to PermonQP (Section 6), calls a specific series of QP transforms provided by PermonQP, resulting in the bound and equality constrained QP (3), i.e. (5) with $\mathbf{A} = \mathbf{PFP}$, $\mathbf{b} = \mathbf{Pd}$, $\mathbf{C} = \mathbf{G}$, $\boldsymbol{\ell} = -\widetilde{\boldsymbol{\lambda}}_I$, and $\mathbf{x} = \boldsymbol{\lambda}$, which is then solved with the `QPSSolve` function. Listing 1.2 presents a sketch of the `FllopSolve` function.

Open source DDM codes are relatively rare. Let us mention the Multilevel BDDC solver library (BDDCML) by J. Šístek et al. [28,29], PETSc BDDC preconditioner implementation by S. Zampini [1], and the HPDDM code by P. Jolivet and F. Nataf [21,20].

```
/* FllopSolve() function */

/*  Subdomain data. */
Mat Ks,BIs,Bgs,Bds,Rs;  Vec fs;
/*  Global data. */
Mat K, BI, Bg, Bd, R ;  Vec f, cI, cd;
/*  QP problem, QP solver. */
QP qp; QPS qps;

/*  Create a QP data structure. */
QPCreate(comm, &qp);

/*  Globalise the data. */
MatCreateBlockDiag(Ks, &K);
MatCreateBlockDiag(Rs, &R);
MatMerge(Bgs, &Bg); MatMerge(Bds, &Bd);
MatMerge(BIs, &BI); VecMerge(fs, &f);

/*  Set the QP data. */
QPSetOperator(qp, K);
QPSetOperatorNullspace(qp, R);
QPSetRHS( qp, f);
QPAddEq(  qp, Bg, NULL); // NULL means zero vector
QPAddEq(  qp, Bd, cd);
QPSetIneq(qp, BI, cI);
```

```
/*  Basic sequence of QP transforms
    giving (T)FETI method.
    QPTFetiPrepare() can be used
    instead for convenience.
    QP chain is created in backend. */
QPTScale(qp);
QPTDualize(qp);
QPTScale(qp);
QPTHomogenizeEq(qp);
QPTEnforceEqByProjector(qp);

/*  Create a PermonQP solver. */
QPSCreate(comm, &qps);

/*  Set the QP to be solved. */
QPSSetQP(qps, qp);

/*  Solve, i.e. hand over to PermonQP.
    The last QP in the chain is solved.
*/
QPSSolve(qps);
```

Listing 1.2: PermonFLLOP calls PermonQP

6 PermonQP

PermonQP [18] allows solving QPs with an SPS Hessian and any combination of linear equality and inequality constraints including unconstrained QP. It provides a basic framework for QP solution (data structures, transformations, and supporting functions), a wrapper of PETSc KSP linear solvers for solving unconstrained and equality-constrained QP, a variant of the augmented Lagrangian method called SMALBE discussed later in Section 8, and several concrete solvers for bound constrained minimization (PermonIneq) – here we consider the MPRGP algorithm (Section 7). Its programming interface (API) is designed to be easy-to-use, and at the same time efficient and suitable for HPC. PermonQP is under preparation for publishing under the BSD 2-Clause license.

A QP transform derives a new QP from the given QP, so that a doubly linked list is generated where each node is a QP. The solution process is divided into the following sequence of actions:

1. QP problem specification;
2. a chain of QP transforms generating a chain of QP problems where the last one is passed to the solver;
3. automatic or manual choice of an appropriate QP solver;
4. the QP solver is called to solve the last QP in the chain;
5. a chain of reconstructions in the reverse order of QP transforms in order to get a solution of the original QP.

7 MPRGP

MPRGP (Modified Proportioning and Reduced Gradient Projection) [15,8] is an efficient algorithm for solution of convex QP with simple bounds

$$\min \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} \quad \text{s.t.} \quad \mathbf{x}_I \geq \boldsymbol{\ell}, \tag{4}$$

where I denotes the index set corresponding to the inequality constrained entries of vector $\mathbf{x}$, and $\mathbf{x}_I$ denotes the subvector of $\mathbf{x}$ given by the index set I. This approach was introduced in [11]. The proportioning algorithm is combined with the gradient projections, a test to decide when to leave the active set, and three types of steps to generate a sequence of iterates $\mathbf{x}^k$ approximating the solution:

1. a proportioning step – removes indices from the active set,
2. a conjugate gradient (CG) step – generates the next approximation in the active set if the current approximation is proportional (i.e. meeting a special criterion related to chopped, free and reduced free gradients, see [8]),
3. an expansion step – defined by the free gradient projection with a fixed steplength $\bar{\alpha}$, expands the active set.

Instead of verifying the Karush-Kuhn-Tucker optimality conditions directly, the algorithm evaluates the *projected gradient* $\mathbf{g}^P$, given componentwise by

$$\mathbf{g}_i^P = \begin{cases} \mathbf{g}_i & \text{for} \quad x_i > l_i \quad \text{or} \quad i \notin I, \\ \min(\mathbf{g}_i, 0) & \text{for} \quad x_i = l_i \quad \text{and} \quad i \in I, \end{cases}$$

where x_i and l_i is the i-th component of $\mathbf{x}$ and $\boldsymbol{\ell}$, respectively, and $\mathbf{g} = \mathbf{A}\mathbf{x} - \mathbf{b}$ is the gradient of the objective function. The algorithm stops, when $||\mathbf{g}^P||$ is sufficiently small. MPRGP has a known rate of convergence given in terms of the spectral condition number of the Hessian, and may be comparable to the cost of solution of the corresponding unconstrained QP [8].

8 SMALBE

SMALBE (Semi-Monotonic Augmented Lagrangian with Bound and Equality) [8] is a variant of the inexact augmented Lagrangian algorithm, and can be viewed as an extension of the augmented Lagrangian method. It can be used to solve a box and equality constrained QP

$$\min \ \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} \quad \text{s.t.} \quad \mathbf{x}_I \geq \boldsymbol{\ell} \quad \text{and} \quad \mathbf{C}\mathbf{x} = \mathbf{o}. \tag{5}$$

Particularly, such QPs arise from applying the FETI methodology to contact problems. The SMALBE algorithm is based on the outer loop refining the Lagrange multipliers $\boldsymbol{\mu}$ related to the equality constraints. In each outer iteration, the inner loop solving an auxiliary minimization problem

$$\min_{\mathbf{x}} \ L(\mathbf{x}, \boldsymbol{\mu}, \rho) \quad \text{s.t.} \quad \mathbf{x}_I \geq \boldsymbol{\ell} \tag{6}$$

is performed, where L is the *augmented Lagrangian* defined as

$$L(\mathbf{x}, \boldsymbol{\mu}, \rho) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{b}^T\mathbf{x} + \boldsymbol{\mu}^T\mathbf{C}\mathbf{x} + \frac{\rho}{2}||\mathbf{C}\mathbf{x}||^2.$$

Using just a different bracketing, this inner problem is a QP with the penalized Hessian and updated right-hand side

$$\min_{\mathbf{x}} \ \frac{1}{2}\mathbf{x}^T(\mathbf{A} + \rho\mathbf{C}^T\mathbf{C})\mathbf{x} - (\mathbf{b} - \mathbf{C}^T\boldsymbol{\mu})^T\mathbf{x} \quad \text{s.t.} \quad \mathbf{x}_I \geq \boldsymbol{\ell}, \tag{7}$$

solvable by any solver for bound constrained QP such as MPRGP (Section 7). Here $\boldsymbol{\mu}$ is an approximation of the equality constraint Lagrange multipliers. The inner loop is stopped if $||\mathbf{g}^P|| < \varepsilon$ and $||\mathbf{Cx}|| < \varepsilon$ (the outer QP is already solved), or as early as when $||\mathbf{g}^P|| < \min\{M||\mathbf{Cx}||, \eta\}$, where $M > 0$ and $\eta > 0$ are algorithmic parameters.

The outer also updates the M parameter (SMALBE-M) or the penalty ρ (SMALBE-ρ) based on the increase of the augmented Lagrangian L with respect to the equality constraint Lagrange multiplier $\boldsymbol{\mu}$. SMALBE-M is preferred as it uses a fixed ρ and hence the Hessian and its spectrum is not altered. M is divided by the update constant $\beta > 1$ if the increase of the augmented Lagrangian with respect to $\boldsymbol{\mu}$ is not sufficient. Compared to the basic penalty method, the algorithm is able to find an approximation of $\boldsymbol{\mu}$ meeting the given precision with no need for a large penalty, avoiding ill-conditioning. Compared to the basic augmented Lagrangian method, the introduced adaptivity weakens the effect of the proper selection of the penalty sequence and eliminates necessity of exact solution of the inner problems. Optimality results for SMALBE were presented in [10,9,8,7].

9 Numerical experiments

We illustrate the numerical scalability of TFETI for contact problems combined with SMALBE and MPRGP and weak parallel scalability of their PERMON implementations on the three model problems introduced in Section 2. The descriptions of the problems and their respective solutions have been shown in Fig. 1.

Regarding the first two problems, the coercive and semicoercive membrane problems, each of two membranes was first partitioned into subdomains with the sidelengths $H \in \{1/8, 1/11, 1/16, 1/24, 1/32\}$. The square subdomains were then discretized by the regular grids with the discretization parameter $h = H/128$, so that the ratio H/h was kept constant. The third problem, the elastic cube, was decomposed into subdomains with sidelengths $H \in \{1/5, 1/6, 1/8, 1/10, 1/13\}$ and discretized with $h = 1/20$ and again with constant H/h. In all cases, the splitting was chosen in order to get the numbers of subdomains near the powers of two. The corresponding total numbers of subdomains as well as the primal, dual and kernel dimensions can be found in Tables 1 and 2. Let us remind that the dual dimension is the total number of gluing, Dirichlet and non-penetration interface constraints, i.e. number of rows of the matrix $\mathbf{B}$.

$1/H$	# subdomains	# DOFs	dual dim – coercive	dual dim – semicoercive	kernel dim
8	128	2,130,048	32,160	31,142	128
11	242	4,027,122	61,377	59,978	242
16	512	8,520,192	130,872	128,838	512
24	1,152	19,170,432	296,144	293,094	1,152
32	2,048	34,080,768	527,976	523,910	2,048

Table 1: Dimension settings for coercive and semicoercive problem with $h = H/128$.

We used the following parameters setting for the SMALBE and MPRGP algorithms:

$$M_0 = 100||\mathbf{PFP}||,\ \rho = 2||\mathbf{PFP}||,\ \eta = 0.1||\mathbf{Pd}|| \quad \text{and} \quad \beta = 10,$$

$1/H$	# subdomains	# DOFs	dual dim	kernel dim
5	125	3,472,875	469,392	750
6	216	6,001,128	832,194	1,296
8	512	14,224,896	2,035,950	3,072
10	1,000	27,783,000	4,051,602	6,000
13	2,197	61,039,251	9,055,080	13,182

Table 2: Dimension settings for cube in 3D with $h = H/24$.

where the matrix norms were approximated using the power method. These values are default in PermonQP and they have been chosen based on many comparative numerical tests. Needless to say, the optimal values for particular problems may slightly differ.

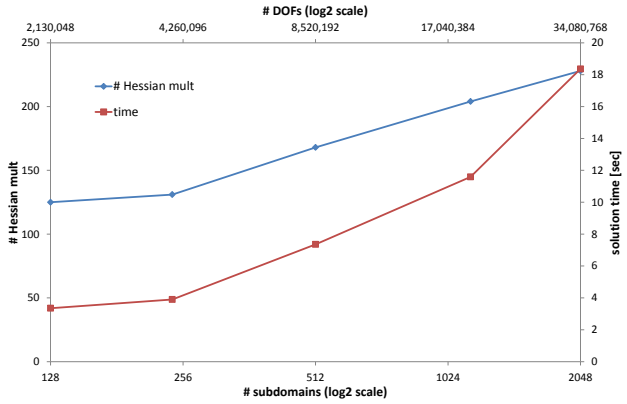
The stopping criterion was

$$||\mathbf{g}^P|| \leq \varepsilon||\mathbf{Pd}|| \ \wedge \ ||\mathbf{G}\boldsymbol{\lambda}|| \leq \varepsilon||\mathbf{Pd}||, \qquad \varepsilon = 10^{-4}.$$

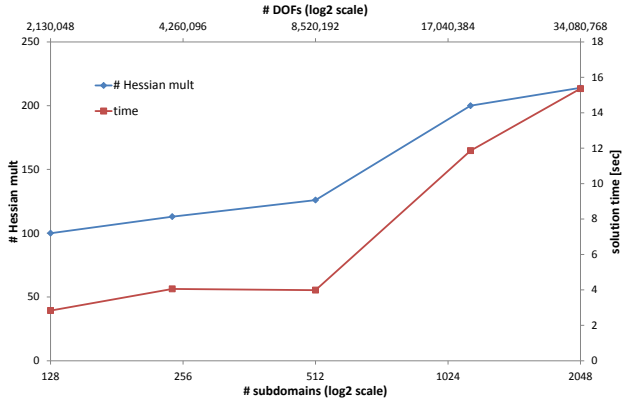
The stiffness matrix pseudoinverse $\mathbf{K}^\dagger$ was implemented using the Cholesky factorization from the MUMPS library [2]. The approach from [6] was used where the original matrix $\mathbf{K}$ is regularized and the inverse of the regularized matrix is a pseudoinverse of $\mathbf{K}$. The coarse problem (action of $(\mathbf{G}\mathbf{G}^T)^{-1}$) was solved by the LU factorization from the SuperLU_DIST library [25] in subcommunicators of size $N_S^{1/2}$ and $N_S^{2/3}$ for PermonMembrane and PermonCube, respectively, each subcommunicator solving the same coarse problem redundantly.

The performance results are shown in Fig. 3. All the graphs illustrate the numerical and weak parallel scalability up to more than 2000 cores. The numerical scalability of the used TFETI + SMALBE + MPRGP approach has been theoretically proven in [8]. It says that keeping the ratio H/h constant, the number of Hessian multiplications is bound by a constant for any problem size. The numerical scalability graphs (with circle marks) reveal the PermonQP implementation fulfils this fairly well. Parallel scalability graphs (with box marks) show the total solution times, i.e. time spent in PermonFLLOP and PermonQP including necessary pre- and post-processing steps before and after the iterative phase. Each parallel scalability graph follows the shape of the respective numerical scalability graph up to ca. 1000 subdomains. Then some worse scalable parts of the implementation start to spoil the parallel scalability. They include e.g. the implementation of the $\mathbf{B}$ actions and the matrix-matrix product $\mathbf{G} * \mathbf{G}^T$. Improving these parts is work-in-progress.

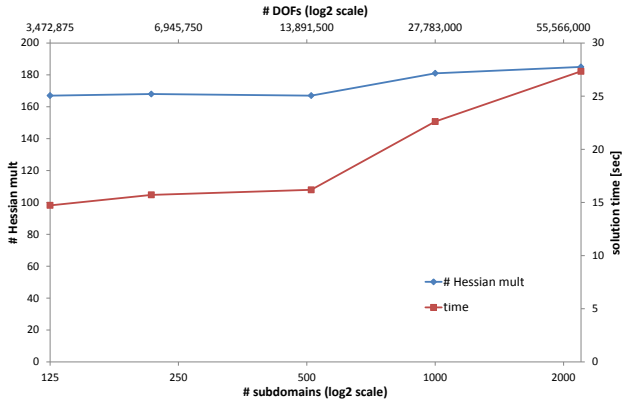
The results were obtained at the ARCHER, the latest UK National Supercomputing Service. It is based around a Cray XC30 supercomputer with 4920 nodes, 118,080 cores and 1.56 Petaflops of theoretical peak performance (4544 standard nodes with 64 GB memory (12 groups, 109,056 cores) and 376 nodes with 128 GB memory (1 group, 9,024 cores)). All compute nodes are connected together in the Dragonfly topology by the Aries interconnect. Each compute node contains two 2.7 GHz, 12-core E5-2697 v2 (Ivy Bridge) series processors. Within the node, the two processors are connected by two QuickPath Interconnect (QPI) links. The memory is arranged in a non-uniform access (NUMA) form: each 12-core processor is a single NUMA region with local memory of 32 GB (or 64 GB for high-memory nodes).



(a)



(b)



(c)

Fig. 3: Graphs of numerical and weak parallel scalability for the coercive (3a) and semicoercive (3b) membrane problems, and the cube problem (3c) .

10 Conclusion

We have presented our new PERMON toolbox and its packages. PermonMembrane and PermonCube were used to generate the model contact problems, PermonFLLOP generated extra data related to FETI, and PermonQP solved the resulting QP problem. We have briefly reviewed the TFETI method, and MPRGP and SMALBE QP algorithms. Finally, benchmarks of two membranes and of the elastic cube were presented to demonstrate efficiency of the PERMON tools for solution of variational inequalities.

Acknowledgements

We would like to thank our colleague Alexandros Markopoulos for developing most of the PermonCube package. This work made use of the facilities of ARCHER, the UK's national high-performance computing service, provided by The Engineering and Physical Sciences Research Council (EPSRC), The Natural Environment Research Council (NERC), EPCC, Cray Inc. and The University of Edinburgh. We also acknowledge use of Anselm and Salomon clusters, operated by IT4Innovations National Supercomputing Center, VSB - Technical University of Ostrava, Czech Republic, for development and testing of the PERMON software. This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science" (LQ1602), and from the "Large Infrastructures for Research, Experimental Development and Innovations" project "IT4Innovations National Supercomputing Center" (LM2015070); by the EXA2CT project funded from the EU's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 610741; by the internal student grant competition project "PERMON toolbox development II" (SP2016/178); by the POSTDOCI II project (CZ.1.07/2.3.00/30.0055) within Operational Programme Education for Competitiveness; and by the Grant Agency of the Czech Republic (GACR) project no. 15-18274S. This work was also supported by the READEX project the European Union's Horizon 2020 research and innovation programme under grant agreement No. 67157.

References

1. PETSc PCBDDC manual page, `http://www.mcs.anl.gov/petsc/petsc-current/docs/manualpages/PC/PCBDDC.html`
2. Amestoy, P., et al.: MUMPS web pages (2015), `http://mumps.enseeiht.fr/index.php?page=home`
3. Balay, S., Abhyankar, S., Adams, M.F., Brown, J., Brune, P., Buschelman, K., Eijkhout, V., Gropp, W.D., Kaushik, D., Knepley, M.G., McInnes, L.C., Rupp, K., Smith, B.F., Zhang, H.: PETSc web pages (2015), `http://www.mcs.anl.gov/petsc`
4. Brzobohatý, T., Dostál, Z., Kozubek, T., Kovář, P., Markopoulos, A.: Cholesky decomposition with fixing nodes to stable computation of a generalized inverse of the stiffness matrix of a floating structure. International Journal for Numerical Methods in Enginering 88(5), 493–509 (2011)
5. Dostál, Z., Horák, D., Kučera, R.: Total FETI – an easier implementable variant of the FETI method for numerical solution of elliptic PDE. Communications in Numerical Methods in Engineering 22(12), 1155–1162 (2006)
6. Dostál, Z., Kozubek, T., Markopoulos, A., Menšík, M.: Cholesky decomposition of a positive semidefinite matrix with known kernel. Applied Mathematics and Computation 217(13), 6067–6077 (2011)
7. Dostál, Z., Kozubek, T., Vondrák, V., Brzobohatý, T., Markopoulos, A.: Scalable TFETI algorithm for the solution of multibody contact problems of elasticity. International Journal for Numerical Methods in Engineering 82(11), 1384–1405 (2010)

8. Dostál, Z.: Optimal Quadratic Programming Algorithms, with Applications to Variational Inequalities, vol. 23. SOIA, Springer, New York, US (2009)
9. Dostál, Z., Horák, D.: Theoretically supported scalable FETI for numerical solution of variational inequalities. SIAM Journal on Numerical Analysis 45(2), 500–513 (2007)
10. Dostál, Z., Horák, D., Kučera, R., Vondrák, V., Haslinger, J., Dobiáš, J., Pták, S.: FETI based algorithms for contact problems: scalability, large displacements and 3d coulomb friction. Computer Methods in Applied Mechanics and Engineering 194(2–5), 395–409 (2005)
11. Dostál, Z., Schöberl, J.: Minimizing quadratic functions subject to bound constraints. Computational Optimization and Applications 30(1), 23–43 (January 2005)
12. Farhat, C., Mandel, J., Roux, F.X.: Optimal convergence properties of the FETI domain decomposition method. Computer Methods in Applied Mechanics and Engineering 115, 365–385 (1994)
13. Farhat, C., Roux, F.X.: A method of finite element tearing and interconnecting and its parallel solution algorithm. International Journal for Numerical Methods in Engineering 32(6), 1205–1227 (1991)
14. Farhat, C., Roux, F.X.: An unconventional domain decomposition method for an efficient parallel solution of large-scale finite element systems. SIAM Journal on Scientific and Statistical Computing (1) (1992)
15. Friedlander, A., Martínez, J.M., Raydan, M.: New method for large-scale box constrained convex quadratic minimization problems. Optimization Methods and Software 5(1), 57–74 (1995)
16. Gosselet, P., Rey, C.: Non-overlapping domain decomposition methods in structural mechanics. Archives of Computational Methods in Engineering 13(4), 515–572 (2006)
17. Hapla, V., et al.: PERMON (Parallel, Efficient, Robust, Modular, Object-oriented, Numerical) web pages (2015), http://industry.it4i.cz/en/products/permon/
18. Hapla, V., et al.: PermonQP web pages (2015), http://industry.it4i.cz/en/products/permon/qp/
19. Hensinger, D.M., Drake, R.R., Foucar, J.G., Gardiner, T.A.: Pamgen, a library for parallel generation of simple finite element meshes. Tech. Rep. SAND2008-1933, Sandia National Laboratories technical report (2008)
20. Jolivet, P., Hecht, F., Nataf, F., Prud'homme, C.: Scalable domain decomposition preconditioners for heterogeneous elliptic problems. In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. pp. 80:1–80:11. SC '13, ACM, New York, NY, USA (2013)
21. Jolivet, P., et al.: HPDDM ? high-performance unified framework for domain decomposition methods, https://github.com/hpddm/hpddm
22. Kozubek, T., Vondrák, V., Menšík, M., Horák, D., Dostál, Z., Hapla, V., Kabelíková, P., Čermák, M.: Total FETI domain decomposition method and its massively parallel implementation. Advances in Engineering Software 60-61, 14–22 (2013)
23. Kruis, J.: Domain Decomposition Methods for Distributed Computing. Saxe-Coburg Publications (2006)
24. Kruis, J.: The FETI method and its applications: A review. In: Topping, B., Iványi, P. (eds.) Parallel, Distributed and Grid Computing for Engineering. vol. 21, pp. 199–216. Saxe-Coburg Publications, Stirling, Scotland (2009)
25. Li, X.S., et al.: SuperLU, http://acts.nersc.gov/superlu/
26. Markopoulos, A., Hapla, V., Cermak, M., Fusek, M.: Massively parallel solution of elastoplasticity problems with tens of millions of unknowns using PermonCube and FLLOP packages. Applied Mathematics and Computation (2015)
27. Raback, P., et al.: Elmer web pages (2015), http://www.csc.fi/english/pages/elmer/
28. Šístek, J., et al.: The Multilevel BDDC solver library (BDDCML), http://users.math.cas.cz/~sistek/software/bddcml.html
29. Sousedík, B., Šístek, J., Mandel, J.: Adaptive-multilevel bddc and its parallel implementation. Computing 95(12), 1087–1119 (2013)
30. The Trilinos Project: PAMGEN web pages (2015), http://trilinos.org/packages/pamgen/

31. Vašatová, A., Čermák, M., Hapla, V.: Parallel implementation of the FETI DDM constraint matrix on top of PETSc for the PermonFLLOP package. In: Parallel Processing and Applied Mathematics 2015. Lecture Notes in Computer Science (accepted 2015)
32. Čermák, M., Hapla, V., Horák, D., Merta, M., Markopoulos, A.: Total-FETI domain decomposition method for solution of elasto-plastic problems. Advances in Engineering Software 84(0), 48–54 (2015)