Faculty of Eletrical Engineering and Computer Science

Duality-based domain decomposition with natural coarse-space for variational inequalities and its parallel implementation

Diploma thesis
2000

Made by:	David Horák
Branch:	Computer Science and Applied Mathematics
Advisor:	Prof. RNDr. Zdeněk Dostál, CSc.
	Department of Applied Mathematics of VŠB - Technical University of Ostrava
Opponent:	Prof. RNDr. Radim Blaheta, CSc.
	Institute of Geonics at Czech Academy of Sciences

Acknowledgments

The aim of this diploma thesis is to put the genius idea proposed by prof. Zdeněk Dostál and his colleagues into the life. However in advance I would like to thank above all my advisor prof. Zdeněk Dostál, that he has entrusted this theme to solve just to me, for his help during the solution of this problem and for everything, what he has taught me during all my studies and I hope he will still teach. At the same time I thank also to opponent of this work, prof. Radim Blaheta. The numerical experiments were very exacting on configuration of parallel computer, installations and various changes and that's why I thank in this place to ing. Jan Haluza, ing. Karel Krečmer and RNDr. Ondřej Jakl, without them the successful realization would be very time-consuming. Last but not least, I thank to my family and all my friends for their love and support.

I declare, that I have made all my diploma thesis on my own and only with the application of the literature presented in references.

Rychvald, 4th May 2000

Contents

List of Figures

List of Tables

Notation

Following symbols keep their meaning in this work:

$a(\cdot,\cdot)$	symmetric bilinear form
$l(\cdot)$	linear form
u	displacements field
f	density of forces
λ	vector of Lagrange multipliers
A	stiffness matrix - symmetric positive definite
B	matrix with constraints
C^{-1}	preconditioner
R	null space of matrix A
g	gradient
Ω^i	domain or subdomain
$\Gamma^{i,j}$	boundary between subdomains Ω^i and Ω^j
$\mathbb{R}^n$	n-th dimensional real space
$\mathbb{L}^2(\Omega^i)$	space of functions on Ω^i whose squares are integrable in the sence of Lebesgue
$\mathbb{H}^1(\Omega^i)$	Sobolev space of first order
$\mathcal{K}_h$	coarse grid space
$J(\cdot)$	energy functional
$\|\cdot\|$	Euclidian norm

Applied abbreviations:

DD	Domain Decomposition
QP	Quadratic Programming
CG	Conjugate Gradient
PDEs	Partial Differential Equations
s.t.	so that

Chapter 1

Introduction

"If we perform large calculations occuring especially in astronomy, geonics, mechanics and elsewhere, we get as a rule bad results, because we are not appropriate practised in numerical reckonning. Sources of arithmetical faults are: confusion of calculations, negligent writing of figures, large haste and small heed during execution the calculations. That's why it is necessary to get used to reckon steadily, carefully, to write figures conspicously and one below another. A good reckonner writes about 40 figures per minute, otherwise the multiplication of 5-figured number by 5-figured one takes him about 1 minute in usually way. The famous reckonner Dase needed for multiplication of two 100-figured numbers about 9 hours. The calculation by a machine is for skilled reckonner faster and more comfortable and takes him about 1/4 or 1/5 of time needed for the same calculation in common way"

This is only a part of Introduction of charming book: Theory and Practice of Numerical Reckonning [1] from the year 1927. It is admirable, what our ancestors knew and it is amazing at the same time, that mathematics and computing made enormous progress and improvement during the last 3/4 of century. Our ancestors would have been doubtless shocked, if they had seen the results this branch had reached to. We are aware, that we would not be able to build it up to the present level without their firm and good fundamentals. Many people are convinced, that the development in applied mathematics is at the end. I am aware of further development in this branch, although each age has its charm, positives and negatives. I am very happy, that the birth was granted me just to this age, when mathematics and computer science complement perfectly each other.

Partial Differential Equations (PDEs) are encountered in all the sciences and engineering disciplines. An effort to implement the solution of these equations on parallel computers leads to the domain decomposition, that consists in division of global domain over which the solution is sought into subdomains and allowing each processor of parallel computer to handle individual subdomain. The PDEs are traditionally clasified into 3 categories: elliptic (from numerical viewpoint belonging to static equations), parabolic and hyperbolic (both belonging to time dependent

equations).

The objective of this work is to explain the basic principles of recently suggested efficient domain decomposition algorithm for the solution of variational inequalities arising from elliptic problems with inequality boundary conditions (suggested by Dostál, Gomes Neto and Santos) and to present the parallelization strategy that has been employed for implementation of this FETI related solver (FETI means the Finite Element Tearing and Interconnecting method called also as the dual Schur complement method) in PETSc and the numerical experiments reached by its variants.

Discretized problem is first turned by duality theory of convex programming into a quadratic programming problem and modified by means of orthogonal projectors to the natural coarse space (suggested by Mandel, Farhat and Roux). The problem is then solved by augmented Lagrangian algorithm with an outer loop for Lagrange multipliers for equality constraints, ensuring continuity among non-overlapping subdomains, and inner loop for solution of bound constrained quadratic programming problems. The projectors and preconditioning - that all guarantee an optimal convergence of iterative solution of auxiliary linear problems. Presented theoretical results and numerical experiments obtained by parallel program using PETSc indicate high numerical and parallel scalability of this algorithm. The aplications include e.g. problem of finding stresses and displacements of a system of linear elastic bodies without friction or the contact problem with Coulomb friction.

The work is organized as follows: chapter 2 describes model problem and its continuous formulation, chapter 3 explains discretization and domain decomposition, chapter 4 recalls the principles of dual formulation, chapter 5 presents modifications of problem accelerating the copmutation, chapter 6 is devoted to the suggested algorithm, chapter 7 introduces the tool for parallel implementation - PETSc, chapter 8 details the complete parallel implementation of algorithm, chapter 9 presents numerical experiments executed on IBM SP2 and chapter 10 concludes all the work.

Chapter 2

Model Problem and Continuous Formulation

This chapter introduces variational inequality model problem, which shall simplify the exposition and together with it represents problem for numerical experiments (see Figure 2.1).

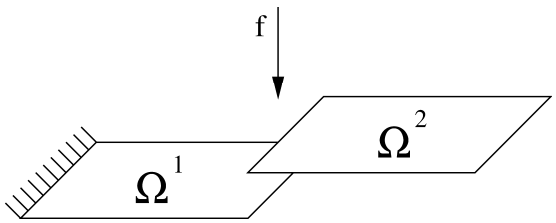


Figure 2.1: Model Problem

This problem arising from variational formulation of problem with inequality boundary conditions reads:

find sufficiently smooth $u(x, y)$ so that:

$$-\triangle u = f \quad \text{in} \quad \Omega = \Omega^1 \cup \Omega^2 \tag{2.1}$$

$$u^1 = 0 \quad \text{on} \quad \Gamma_u^1 \tag{2.2}$$

$$\frac{\partial u^i}{\partial n_i} = 0 \quad \text{on} \quad \Gamma_f^i, \ i = 1, 2 \tag{2.3}$$

$$u^2 - u^1 \geq 0 \quad \text{on} \quad \Gamma_c = \Gamma_c^1 = \Gamma_c^2 \tag{2.4}$$

$$\frac{\partial u^2}{\partial n_2} \geq 0 \quad \text{on} \quad \Gamma_c \tag{2.5}$$

$$\frac{\partial u^2}{\partial n_2}(u^2 - u^1) = 0 \quad \text{on} \quad \Gamma_c \tag{2.6}$$

$$\frac{\partial u^1}{\partial n_1} + \frac{\partial u^2}{\partial n_2} = 0 \quad \text{on} \quad \Gamma_c \tag{2.7}$$

here $\Omega^1 = (0,1) \times (0,1), \Omega^2 = (1,2) \times (0,1)$ denote open domains with boundaries Γ^1, Γ^2 and their parts $\Gamma^1_u, \Gamma^i_f, \Gamma^i_c$, formed by the sides of $\Omega^i, i = 1,2$, and $n_k(x,y)$ denote components of the outer unit normal at $(x,y) \in \Gamma^i$ (see Figure 2.2).

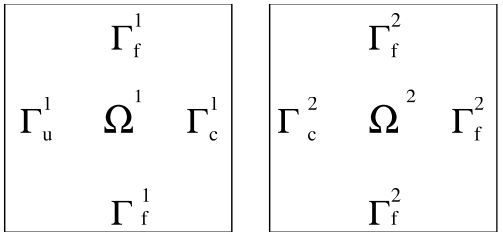


Figure 2.2: Domain of model problem

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 $f(x,y)$. Relation:

- (2.1) - is elliptic Poisson equation
- (2.2) - is boundary condition of Dirichlet type
- (2.3) - is boundary condition of Neumann type
- (2.4) - describes the non-interpenetration of the adjacent edges of the membranes with the edge of the right membrane above the edge of the left membrane
- (2.5) - expresses that right membrane can press the left one down at the contact points
- (2.6)-(2.7) - define points, that are in contact - if there is no contact at $(x,y) \in \Gamma_c$, i.e. $u^2(x,y) > u^1(x,y)$, then the membranes are stretched by the horizontal force in the same way as at $(x,y) \in \Gamma^i_f$.

To derive the variational inequality whose smooth solutions satisfy (2.1)-(2.7), let $\mathbb{H}^1(\Omega^i)$ denote Sobolev space of first order on the space $\mathbb{L}^2(\Omega^i)$ of the functions on Ω^i whose squares are integrable in the sence of Lebesgue. Then $f \in \mathbb{H}^1(\Omega^i)$ iff both f and its first derivates belong to $\mathbb{L}^2(\Omega^i)$.

Let's define following requisities: $\mathbb{V}^1 = \{v \in \mathbb{H}^1(\Omega^1) : v^1 = 0 \text{ on } \Gamma_u^1\} \subseteq \mathbb{H}^1(\Omega^1)$ denotes the closed subspace of $\mathbb{H}^1(\Omega^1)$, $\mathbb{V}^2 = \mathbb{H}^1(\Omega^2)$, $\mathbb{V} = \mathbb{V}^1 \times \mathbb{V}^2$ denotes the closed subspace and $\mathcal{K} = \{(v^1, v^2) \in \mathbb{V} : v^2 - v^1 \geq 0 \text{ on } \Gamma_c\}$ closed convex subset of $\mathcal{H} = \mathbb{H}^1(\Omega^1) \times \mathbb{H}^1(\Omega^2)$. On $\mathcal{H}$ lets define a symmetric bilinear form

$$a(u, v) = \sum_{i=1}^{2} \int_{\Omega^i} \left(\frac{\partial u^i}{\partial x} \frac{\partial v^i}{\partial x} + \frac{\partial v^i}{\partial y} \frac{\partial v^i}{\partial y} \right) d\Omega$$

and a linear form

$$\ell(v) = \sum_{i=1}^{2} \int_{\Omega^i} f^i v^i \, d\Omega.$$

Let u denote a smooth solution of (2.1)-(2.7). After multiplication of (2.1) by $v \in \mathbb{V}$ and application of the Green theorem and using relations (2.2)-(2.3) we obtain

$$a(u, v) - \ell(v) = \int_{\Gamma_c} \left\{ \frac{\partial u^1}{\partial n_1} v^1 + \frac{\partial u^2}{\partial n_2} v^2 \right\} d\Gamma$$

and for $v = w - u$, $w \in \mathcal{K}$

$$a(u, w - u) - \ell(w - u) = \int_{\Gamma_c} \left\{ \frac{\partial u^1}{\partial n_1} (w^1 - u^1) + \frac{\partial u^2}{\partial n_2} (w^2 - u^2) \right\} d\Gamma. \tag{2.8}$$

At the points of Γ_c with $u^1 > u^2$ there is , due to (2.6)-(2.7),

$$\frac{\partial u^1}{\partial n_1} = \frac{\partial u^2}{\partial n_2} = 0, \tag{2.9}$$

so that the integrand in (2.8) vanishes at such points. At the points of Γ_c with $u^1 = u^2$ there is , due to (2.5) and (2.7),

$$\frac{\partial u^1}{\partial n_1} (w^1 - u^1) + \frac{\partial u^2}{\partial n_2} (w^2 - u^2) = \frac{\partial u^1}{\partial n_1} w^1 + \frac{\partial u^2}{\partial n_2} w^2 = \frac{\partial u^2}{\partial n_2} (w^2 - w^1) \geq 0. \tag{2.10}$$

The integral in (2.8) is nonnegative for any $w \in \mathcal{K}$ and the solution u of problem defined by relations (2.1)-(2.7) solves also problem defined in following way

$$\text{find } u \in \mathcal{K} \text{ s.t. } a(u, w - u) - \ell(w - u) \geq 0 \text{ for all } w \in \mathcal{K} \tag{2.11}$$

and opposite |3|.

The expression on the left of inequality in relation (2.11) is gradient of the energy functional

$$J(v) = \frac{1}{2}a(v, v) - \ell(v)$$

at u and that's why the problem given by (2.11) is equivalent to the minimization problem

$$\min\ J(v)\ \text{ s.t. }\ v \in \mathcal{K}. \tag{2.12}$$

It can be proved that functional $J(v)$ is convex and coercive on $\mathcal{K}$ and theorem for existence and uniqueness [4] of minimum for such functionals then guarantees that problems (2.11)-(2.12) have unique solution if f satisfies $\int_{\Omega^2} f\, d\Omega < 0$, what is assumed in all following expositions.

Chapter 3

Discretization and Domain Decomposition

For the numerical proccesing of this problem it is necessary to execute the linearization and finite element discretization. To this purpose let's define (η_h, τ_h) a partitioning of Ω into triangles $T_j \in \tau_h$, with suitable numbering vertices at $N_k \in \eta_h$.

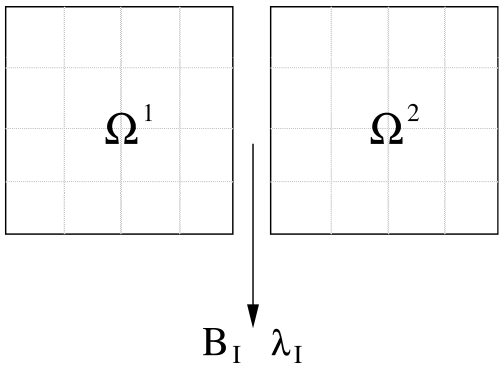


Figure 3.1: Discretization and domain decomposition

For $i = 1, 2$ (see Figure 3.1) let $\mathbb{P}_h^i$ denote the piecewise linear finite element subspaces of $\mathbb{H}^1(\Omega^i)$, let $\mathbb{V}_h^i = \mathbb{P}_h^i \cap \mathbb{V}^i$ and define $\mathbb{V}_h = \mathbb{V}_h^1 \times \mathbb{V}_h^2$ and $\mathcal{K}_h = \mathcal{K} \cap \mathbb{V}_h$. The problem (2.12) is then aproximated by the finite element problem

$$\min\ J(v_h)\ \text{ s.t. }\ v_h \in \mathcal{K}_h. \tag{3.1}$$

The functions $p_h^i \in \mathbb{P}_h^i$ are fully determined by values $u_k^i = p_h^i(N_k^i)$ at nodes $N_k^i \in \overline{\Omega^i}$, while $\overline{\Omega^i}$ denotes a closure of Ω^i. Assuming the independ indexing of nodes of $\Omega^i \setminus \Gamma_u$ by indices $1, 2, ..., s_i$ and denoting of standard basis functions of $\mathbb{V}_k^i$ by e_k^i, so that $e_k^i(N_j^i) = \delta_{kj}$ (Kronecker symbol), it is possible to write any $v_h^i \in \mathbb{V}_h^i$ in

following form

$$v_h^i = \sum_{k=1}^{s_i} u_k^i e_k^i. \tag{3.2}$$

Substituting (3.2) into expression for functional $J(v)$ gives

$$J(v_h) = \frac{1}{2} u^T A u - f^T u,$$

where $A = \begin{bmatrix} A^1 & 0 \\ 0 & A^2 \end{bmatrix}$ is symmetric positive semidefinite block-diagonal stiffness matrix of order n (symmetric means that $a_{ij} = a_{ji}$ for $i, j = 1, ..., n$, while positive semidefinite means that $u^T A u \geq 0$ for all vectors u), $f = \begin{bmatrix} f^1 \\ f^2 \end{bmatrix}$ vector of nodal forces of size n and $u = \begin{bmatrix} u^1 \\ u^2 \end{bmatrix}$ vector of nodal unknowns of size n (note: in previous chapters the symbols u, f denote continuous functions but from this place on, after discretization because of simplicity, u, f represent the vectors), with $A^i = [a_{jk}^i]$, $a_{jk}^i = a(e_j^i, e_k^i)$, $f^i = [f_j^i]$, $f_j^i = \ell(e_j^i)$ and $u^i = [u_j^i]$.

For completion of discretization it is necessary to describe conditions on u_k^i corresponding to

$$\sum_{i=1}^{2} \sum_{k=1}^{s_i} u_k^i e_k^i \in \mathcal{K}_h.$$

Because of doubled nodes on interface Γ_c, the m inequalities $u_j^1 \leq u_k^2$ can be substituted by exprression $B_I u \leq 0$, with $m \times n$ full rank matrix B_I consisting of m rows of form

$$b = \begin{bmatrix} \cdots & 1 & \cdots & -1 & \cdots \end{bmatrix},$$

that means zero entries except positions $j : b_j^1 = 1$ and $k : b_k^2 = -1$ (see Figure 3.2).

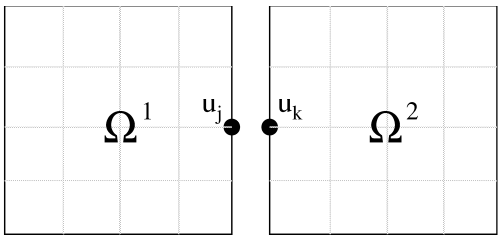


Figure 3.2: Doubled nodes on interface Γ_c

The discretized version of problem (2.12) then reads

$$\min \frac{1}{2}u^T Au - f^T u \text{ s.t. } B_I u \leq 0. \tag{3.3}$$

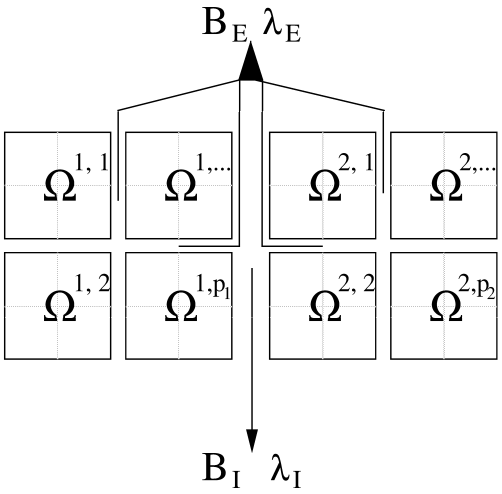


Figure 3.3: Discretization and auxiliary domain decomposition

Above presented exposition takes into account just the decomposition in two subdomains Ω^1 and Ω^2, but it is possible each subdomain Ω^i auxiliary decompose into p_i subdomains Ω^{i,p_i} with interfaces $\Gamma^{i,jk}$ (see Figure 3.3), so that each of these subdomains is partitioned by a subset of (η_h, τ_h). After using the finite element discretization of problem (2.12) with the basis functions that are zero extensions of $\mathbb{P}_h(\Omega^{i,j})$ for $i = 1, 2$ and $j = 1, ..., p_i$, the functional has for all piecewise linear functions v_h continuous in subdomains $\Omega^{i,j}$ the form

$$J(v_h) = \frac{1}{2}u^T Au - f^T u,$$

where $A = \begin{bmatrix} A^{1,1} & 0 & \dots & \dots & \dots & 0 \\ 0 & \ddots & 0 & \dots & \dots & 0 \\ 0 & 0 & A^{1,p_1} & 0 & \dots & 0 \\ 0 & \dots & 0 & A^{2,1} & 0 & 0 \\ 0 & \dots & \dots & 0 & \ddots & 0 \\ 0 & \dots & \dots & \dots & 0 & A^{2,p_2} \end{bmatrix}$, $f = \begin{bmatrix} f^{1,1} \\ \vdots \\ f^{1,p_1} \\ f^{2,1} \\ \vdots \\ f^{2,p_2} \end{bmatrix}$ and $u = \begin{bmatrix} u^{1,1} \\ \vdots \\ u^{1,p_1} \\ u^{2,1} \\ \vdots \\ u^{2,p_2} \end{bmatrix}$

have the same interpretation as above.

The enforcing of continuity across $\Gamma^{i,jk}$ is more complicated in this case. For this purpose let's define matrix B_E, which has for each node $N \in \eta_h \cap \Gamma^{i,jk}$ a row of zero entries except $1, -1$ at the positions corresponding to the indices of N in $\overline{\Omega^{i,j}}$ and

$\overline{\Omega^{i,k}}$. In the case of corner nodes that belong to four subdomains, the problem is solved by means of following three rows

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} \cdots & \frac{\sqrt{2}}{2} & \cdots & \frac{\sqrt{2}}{2} & \cdots & -\frac{\sqrt{2}}{2} & \cdots & -\frac{\sqrt{2}}{2} & \cdots \\ \cdots & 1 & \cdots & -1 & \cdots & 0 & \cdots & 0 & \cdots \\ \cdots & 0 & \cdots & 0 & \cdots & 1 & \cdots & -1 & \cdots \end{bmatrix},$$

whose four nonzero columns i, j, k, l correspond to four nodes joined with global indices i, j, k, l. If these nodes belong to the contact interface Γ_c, rows b_1, b_2, b_3 are used to form matrix B_E, otherwise row b_1 moves to B_I possibly replacing all other rows with nonzero columns i, j, k, l and rows b_2 and b_3 moves to B_E (see Figure 3.4).

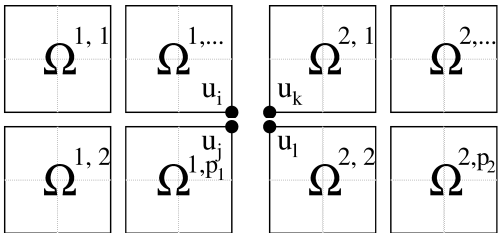


Figure 3.4: Corner nodes belonging to four subdomains

The modified problem after above described refinement then reads

$$\min \frac{1}{2} u^T A u - f^T u \text{ s.t. } B_I u \leq 0 \text{ and } B_E u = 0. \tag{3.4}$$

Chapter 4

Dual Formulation

Although (3.4) is a typical quadratic programming problem, it is not very suitable for numerical computation especially because of ill conditioning, singularity of stiffness matrix A or complicated feasible set. How to avoid above introduced complications is to apply duality theory of convex programming.

The Lagrangian of problem (3.4) looks

$$L(u, \lambda_I, \lambda_E) = \frac{1}{2} u^T A u - f^T u + \lambda_I^T B_I u + \lambda_E^T B_E u, \tag{4.1}$$

where λ_I and λ_E denote Lagrange multipliers associated with inequalities and equalities. Using notation $\lambda = \begin{bmatrix} \lambda_I \\ \lambda_E \end{bmatrix}$ and $B = \begin{bmatrix} B_I \\ B_E \end{bmatrix}$ the Lagrangian gets simplified form

$$L(u, \lambda) = \frac{1}{2} u^T A u - f^T u + \lambda^T B u.$$

The problem (4.1) is then equivalent to the saddle point problem

$$\text{find } (\overline{u}, \overline{\lambda}) \text{ s.t. } L(\overline{u}, \overline{\lambda}) = \sup_{\lambda_I > 0} \inf_u L(u, \lambda). \tag{4.2}$$

For fixed λ, the Lagrange function is convex in the first variable and the minimizer u satisfies equation

$$Au = f - B^T \lambda, \tag{4.3}$$

which expresses, from a physical viewpoint, the subdomain equation of equilibrium with Neumann boundary conditions. Equation (4.3) has solutiom iff

$$f - B^{I'} \lambda \in \mathrm{Im} A, \tag{4.4}$$

which can be expressed by means of a matrix R whose rows span the null space of matrix A

$$R(f - B^T\lambda) = 0. \tag{4.5}$$

Matrix R can be formed directly so that each floating subdomain is assigned to a row of R with ones at positions of nodes belonging to this subdomain and zeros elsewhere. The solution u of (4.3) with assumption that λ satisfies (4.4) can be evaluated by formula

$$u = A^+(f - B^T\lambda) + R^T\alpha. \tag{4.6}$$

Here A^+ denotes matrix satisfying $AA^+A = A$ such as generalized inverse or Moore-Penrose pseudoinverse, considering effective solution of given problem - Farhat and Roux proposed to use left generalized inverse with

$$A^{i^+} = A^{i^{-1}}$$

for A^i non-singular considering fixed subdomain and

$$A^{i^+} = S^{i^T} \begin{bmatrix} K_{11}^{i^{-1}} & 0 \\ 0 & 0 \end{bmatrix} S^i$$

with permutation matrix S^i and non-singular matrix K_{11}^i for $A^i = \begin{bmatrix} K_{11}^i & K_{12}^i \\ K_{21}^i & K_{22}^i \end{bmatrix}$ singular considering floating subdomain - in this case matrix A^i is symmetric positive indefinite matrix because of absence of Dirichlet boundary conditions, while with FETI method the local boundary conditions are of Neumann type, and matrix K_{11}^i is positive definite with dimension equal to the rank of A^i and therefore the Schur complement $K_{22}^i - K_{21}^i K_{11}^{i^{-1}} K_{12}^i$ have to be zero.
Further

$$\alpha = -(R\widetilde{B}^T\widetilde{B}R^T)^{-1}R\widetilde{B}^T\widetilde{B}A^+(f - B^T\overline{\lambda})$$

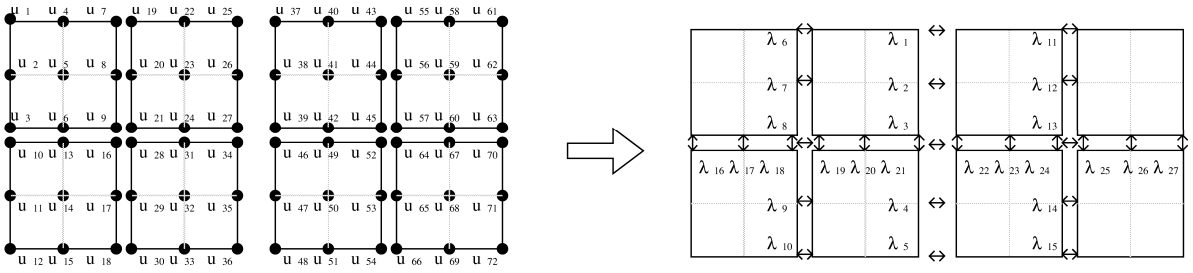
denotes vector of constants determined by relation $\widetilde{B}A^+(f - B^T\overline{\lambda}) + \widetilde{B}R^T\alpha = 0$, with $\widetilde{B} = \begin{bmatrix} \widetilde{B_I} \\ B_E \end{bmatrix}$ and $\widetilde{B_I}$ formed by rows of B_I, that correspond to active constraints latter characterized by $\overline{\lambda_i} = 0$.

Substitution of (4.6) into problem (4.2) and some manipulations lead to minimization problem

$$\min \ \frac{1}{2}\lambda^T BA^+B^T\lambda - \lambda^T BA^+f \ \text{ s.t. } \ \lambda_I \geq 0 \ \text{ and } \ R(f - B^T\lambda) = 0, \tag{4.7}$$

whose Hessian is positive definite and conforms with one from FETI basic method by Farhat and Roux [5], having relatively favorable distributed spectrum for application of the conjugate gradient method.

The illustration of the main idea of dual formulation is showed in the Figure 4.1 - the large problem with primary variable u representing displacement in subdomains nodes is transformed to the significanty smaller problem with dual variable λ representing the "gluing" forces on interfaces.



$Au = f$ s.t. $B_I u \leq 0$ and $B_E u = 0$ $\qquad$ $F\lambda = d$ s.t. $\lambda_I \geq 0$ and $G\lambda = e$

Figure 4.1: Main idea of dual formulation

Chapter 5

Modifications of Problem

Even though the problem (4.7) is much more suitable for computation than (3.4), further improvement can be achieved by adapting simple observations and results of Farhat, Mandel and Roux [6] consisting in formulation of equivalent problem with augmented Lagrangian that has spectral properties guaranting optimal convergence of conjugate gradient method. Before incorporation of these observations let's establish following notation

$$\begin{aligned} F &= BA^{+}B^{T}, & \widetilde{G} &= RB^{T}, \\ \widetilde{d} &= BA^{+}f, & \widetilde{e} &= Rf \end{aligned}$$

and let T denote a regular matrix that defines the orthonormalization of the rows of $\widetilde{G}$ used for definition of matrix

$$G = T\widetilde{G},$$

and vector

$$e = T\widetilde{e}.$$

The problem (4.7) after this denoting has form

$$\min \frac{1}{2}\lambda^{T}F\lambda - \lambda^{T}\widetilde{d} \text{ s.t. } \lambda_{I} \geq 0 \text{ and } G\lambda = e. \tag{5.1}$$

The problem of minimization on the subset of the affine space is transformed to the problem on subset of vector space by means of arbitrary $\widetilde{\lambda}$ which satisfies

$$G\widetilde{\lambda} = e,$$

while the solution is looked for in the form $\lambda = \widehat{\lambda} + \widetilde{\lambda}$. Because of relation

$$\frac{1}{2}\lambda^{T}F\lambda - \lambda^{T}\widetilde{d} = \frac{1}{2}\widehat{\lambda}^{T}F\widehat{\lambda} - \widehat{\lambda}^{T}(\widetilde{d} - F\widetilde{\lambda}) + \frac{1}{2}\widetilde{\lambda}^{T}F\widetilde{\lambda} - \widetilde{\lambda}^{T}\widetilde{d},$$

using old notation and denoting $d = \widetilde{d} - F\widetilde{\lambda}$, the problem (5.1) is equivalent to

$$\min \frac{1}{2}\lambda^T F\lambda - \lambda^T d \text{ s.t. } \lambda_I \geq -\widetilde{\lambda}_I \text{ and } G\lambda = 0. \tag{5.2}$$

Further improvement is based on the observation, that the augmented Lagrangian for problem (5.2) can be decomposed by orthogonal projectors

$$Q = G^{I^{'}}G \qquad \text{and} \qquad P = I - Q$$

on the image space of $G^{I^{'}}$ and on the kernel of G, while they satisfy among others following relations

$$\begin{aligned} \mathrm{Im}Q = \mathrm{Ker}G \qquad & \text{and} \qquad \mathrm{Im}P = \mathrm{Im}G^{I^{'}} \\ P = P^T = P^T P \qquad & \text{and} \qquad Q = Q^T = Q^T Q. \end{aligned}$$

The modified dual formulation of problem (5.2) has then the form

$$\min \frac{1}{2}\lambda^T PFP\lambda - \lambda^T Pd \text{ s.t. } \lambda_I \geq -\widetilde{\lambda}_I \text{ and } G\lambda = 0. \tag{5.3}$$

Let's investigate for a comparison the distributions of the spectrums of Hessians

$$H_1 = F + \rho\widetilde{G}^T\widetilde{G}, \quad H_2 = F + \rho G^T G \text{ and } H_3 = PFP + \rho Q$$

of the augmented Lagrangians corresponding to the problems (5.1), (5.2) and (5.3), with the assumption that the eigenvalues of F are in the interval $[a, b]$, the eigenvalues of $\widetilde{G}^{I^{'}}\widetilde{G}$ are in $[\gamma, \delta]$ and $\sigma(A)$ denotes a spectrum of square matrix A. Using the analysis of [7] it is possible to obtain following estimates

$$\begin{gathered} \sigma(H_1) \subseteq [a, b] \cup [a + \rho\gamma, b + \rho\delta], \ \sigma(H_2) \subseteq [a, b] \cup [a + \rho, b + \rho] \text{ and} \\ \sigma(H_3) \subseteq [a_P, b_P] \cup \{\rho\} \end{gathered}$$

where $[a_P, b_P] \subseteq [a, b]$ denotes the interval of non-zero eigenvalues of PFP (see Figure 5.1). If ρ is sufficiently large and $\gamma < \rho$, then the spectrum of H_1 is distributed in two intervals with the larger one on the right, while the rate of convergence of conjugate gradients for minimization of quadratic function with H_1 depends on the penalization parameter ρ. In the second case the situation is much more favorable for H_2, because the spectrum is always distributed in two intervals of the same length and the rate of convergence is governed by the effective condition number $\overline{\kappa}(H_2) = \frac{4b}{a}$, so that the number k_2 of conjugate gradient iterations for reducing the gradient of the augmented Lagrangian $L_2(\lambda, \mu, \rho) = \frac{1}{2}\lambda^T(F + \rho Q)\lambda - \lambda^T d + \mu^T G\lambda$ for (5.2) by ϵ satisfies

$$k_2 \leq \frac{1}{2}\mathrm{int}\left(\sqrt{\frac{4b}{a}}\ln\left(\frac{2}{\epsilon}\right) + 1\right)$$

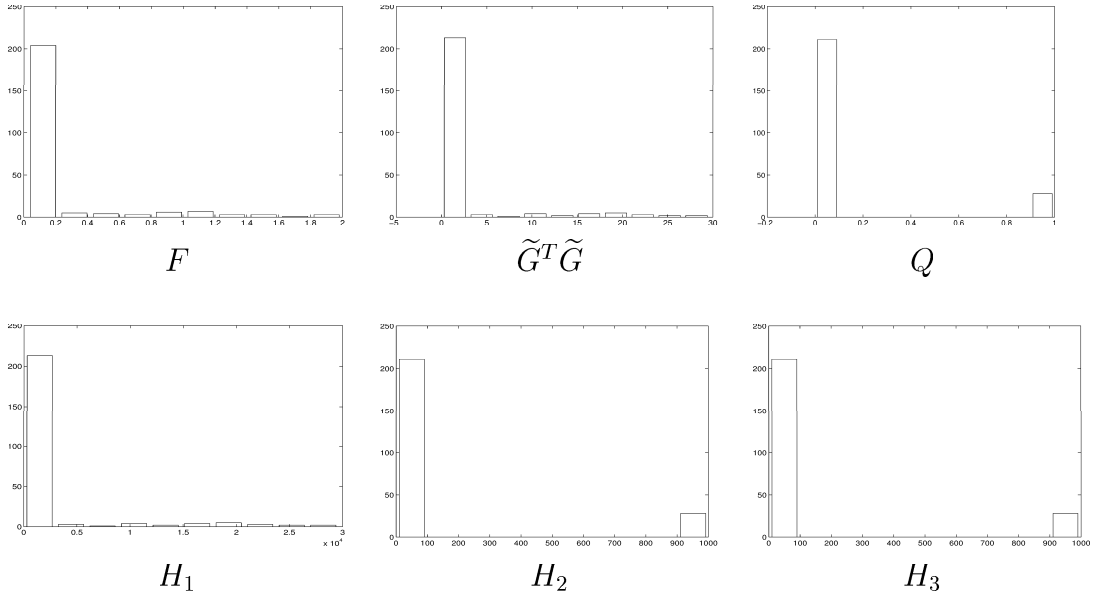


Figure 5.1: Example of spectrum distributions of Hessians ($H = 1/8, h = 1/2, \rho = 10^3$)

and does not depend on the penalization parameter ρ. In the third case the Hessian H_3 of the augmented Lagrangian $L_3(\lambda, \mu, \rho) = \frac{1}{2}\lambda^T(PFP + \rho Q)\lambda - \lambda^T Pd + \mu^T G\lambda$ is decomposed by projectors P and Q whose image spaces are invariant subspaces of H_3 and the number k_3 of conjugate gradient iterations for reducing the gradient of $L_3(\lambda, \mu, \rho)$ for (5.3) by ϵ satisfies

$$k_3 \leq \frac{1}{2}\mathrm{int}\left(\sqrt{\frac{b_P}{a_P}}\ln\left(\frac{2}{\epsilon}\right) + 3\right),$$

while according to the analysis of the FETI method by Farhat, Mandel and Roux following relation is valid

$$\frac{a_P}{b_P} \leq \mathrm{const}\frac{H}{h}$$

with h denoting the mesh and H subdomain diameter (see Figure 5.2). Thus the rate of convergence does not depend on penalization parameter ρ or discretization parameter h, but it is bounded by the constant given by the ratio $\frac{H}{h}$.

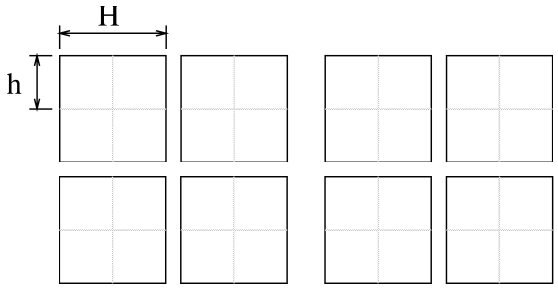


Figure 5.2: Mesh and subdomain parameters

Chapter 6

Algorithm for Quadratic Programming with Equality Constraints and Simple Bounds

This chapter focuses to the description of the algorithm for augmented Lagrangian, that is a variant of algorithm proposed by Conn, Gould and Toint [8] for identification of stacionary points of more general problems and supplied by the adaptive precision control of auxiliary problems in Step 1, so that the algorithm aproximates new Lagrange multipliers for equality constraints in the outer loop, while quadratic programming problems with simple bounds are solved in the inner loop. The algorithm treats each type of constraints separately, so that efficient algorithm using projections and adaptive precision control in the active set strategy [9] may be used for the bound constrained quadratic programming problems.

First let's define the augmented Lagrangian for problem (5.3)

$$L(\lambda, \mu, \rho) = \frac{1}{2}\lambda^T PFP\lambda - \lambda^T Pd + \mu^T G\lambda + \frac{1}{2}\rho \left\|Q\lambda\right\|^2,$$

its gradient

$$g(\lambda, \mu, \rho) = PFP\lambda - Pd + G^T(\mu + \rho G\lambda)$$

and projected gradient $g^P = g^P(\lambda, \mu, \rho)$ of L at λ given componentwise by

$$g_i^P = \begin{cases} g_i \quad \text{for} \quad \lambda_i > -\widetilde{\lambda}_i \quad \text{or} \quad i \notin I \\ \min(g_i, 0) \quad \text{for} \quad \lambda_i = -\widetilde{\lambda}_i \quad \text{and} \quad i \in I \end{cases},$$

with I denoting the set containing the indices of constrained entries of λ.

Algorithm for simple bound and equality constraints

Step 0 : Inicialization of parameters:

parameter	typical value of parameter	role of parameter
$0 < \alpha < 1$	$\alpha = 0.1$	for equality precision update
$\beta > 1$	$\beta = 10$	for penalty update
$\rho_0 > 0$	$\rho_0 = 10^4$	initial penalty parameter
$\eta_0 > 0$	$\eta_0 = 0$	initial equality precision
$M > 0$	$M = 10^4$	for balancing ratio
μ^0	$\mu^0 = o$	initial vector
k	$k = 0$	number of iterations

Step 1 : Find λ^k so,that $\left\|g^P(\lambda^k, \mu^k, \rho_k)\right\| \leq M \left\|G\lambda^k\right\|$
Step 2 : If $\left\|g^p(\lambda^k, \mu^k, \rho_k)\right\|$ and $M \left\|G\lambda^k\right\|$ are sufficiently small then $\lambda^k$is the solution
Step 3 : If $\left\|G\lambda^k\right\| \leq \eta_k$ then
$\mu^{k+1} = \mu^k + \rho_k G\lambda^k, \quad \rho_{k+1} = \rho_k, \quad \eta_{k+1} = \alpha\eta_k$
else $\rho_{k+1} = \beta\rho_k, \quad \eta_{k+1} = \eta_k$
Step 4 : Update: $k = k + 1$ and return to Step 1

Note: The salient feature of this algorithm is that it deals with each type of constraint completely separately and that it accepts inexact solutions for the auxiliary box consrained problems in Step 1. The algorithm has been proved to converge for any combination of parameters satisfying given conditions, while only parameters ρ_0 and M are problem dependent (ρ_0 should be larger than M - the algorithm is designed so that it warrants the adaption of ρ to all parameters including M).

Just the instruction for the gain of λ^k in the Step 1 remains for completeness of presented algorithm. Step 1 can be realized by the minimization of the augmented Lagrangian L subject to $\lambda_I \geq -\widetilde{\lambda}_I$ by efficient algorithm for the solution of convex quadratic pragramming problems with simple bounds proposed independly by Dostál and Friedlander and Martínez, which can be considered a modification of the Polyak algorithm.

If we assume that μ and ρ are fixed, we are allowed to write $\theta(\lambda) = L(\lambda, \mu, \rho)$. Let I and E denote the sets of indices corresponding to the entries of dual vectors λ_I and λ_E and let $\mathcal{A}(\lambda)$ and $\mathcal{F}(\lambda)$ define the *active set* and *free set* of indices of λ

$$\mathcal{A}(\lambda) = \left\{i \in I : \lambda_i = -\widetilde{\lambda}_i\right\} \qquad \text{and} \qquad \mathcal{F}(\lambda) = \left\{i : \lambda_i > -\widetilde{\lambda}_i \text{ or } i \in E\right\},$$

determining the chopped gradient g^C and the inner gradient g^I of $\theta(\lambda)$ in following way

$$g_i^I = g_i \quad \text{for} \quad i \in \mathcal{F}(\lambda) \quad \text{and} \quad g_i^I = 0 \quad \text{for} \quad i \in \mathcal{A}(\lambda)$$
$$g_i^C = 0 \quad \text{for} \quad i \in \mathcal{F}(\lambda) \quad \text{and} \quad g_i^C = \min(g_i, 0) \quad \text{for} \quad i \in \mathcal{A}(\lambda).$$

The unique solution $\overline{\lambda} = \overline{\lambda}(\mu, \rho)$ of auxiliary problems satisfies the Karush-Kuhn-Tucker conditions expressed by the relation

$$g^P(\overline{\lambda}, \mu, \rho) = 0,$$

hence the solution of problem

$$\min \theta(\lambda) \quad \text{s.t.} \quad \lambda_I \geq -\widetilde{\lambda}_I$$

satisfies the Karush-Kuhn-Tucker conditions if the following relation is valid

$$g^P = g^I + g^C.$$

The precision of the solution of auxiliary problems is controlled by norm of violation of Karush-Kuhn-Tucker condition g^C in each inner iterate y^i by

$$\left\| g^C(y^i) \right\| \leq \Gamma \left\| g^I(y^i) \right\|,$$

while $\Gamma > 0$ and y^i satisfying this inequality is called as proportional. The algorithm explores the face

$$W_J = \left\{ y : y_i = -\widetilde{\lambda}_i \quad \text{for} \quad i \in I \right\}$$

with a given active set $J \subseteq I$ until the iterates are proportional, otherwise if y^i is not proportional, a new y^{i+1} is generated by means of descent direction $d^i = -g^C(y^i)$ (without violating any constraint) in step called proportioning, that is succeeded by exploring the new face defined by $J = \mathcal{A}(y^{i+1})$.

Algorithm of General Proportioning Scheme

Let a feasible λ^0 and $\Gamma > 0$, $[\Gamma = 1]$ be given. For $i > 0$ choose λ^{i+1} by following rules

Step 1 : If λ^i is not proportional, define λ^{i+1} by proportioning

Step 2 : If λ^i is proportional, choose a feasible λ^{i+1} so that $\theta(\lambda^{i+1}) \leq \theta(\lambda^i)$ and λ^{i+1} satisfies either condition: $\mathcal{A}(\lambda^i) \subset \mathcal{A}(\lambda^{i+1})$ while λ^{i+1} is not proportional, or λ^{i+1} minimizes θ subject to $\lambda \in W_J, J = \mathcal{A}(\lambda^i)$.

This algorithm is applicable for the minimization of convex quadratic functional $\theta(\lambda)$ with Step 2 implemented by means of conjugate gradient method.

Algorithm of Conjugate Gradient Method for Proportioning

Step 0 : $y^0 = \lambda^k$

Step 1 : Generate by conjugate gradient method the aproximation y^i for $\min\{\theta(y) : y \in W_J, J = \mathcal{A}(y^0)\}$

Step 2 : If y^i is feasible or proportional or not minimizes $\theta(\lambda)$ subject to $\lambda_I \geq -\widetilde{\lambda}_I$ then
$i = i + 1$ and return to Step 1

Step 3 : If y^i is feasible then
$\lambda^{k+1} = y^i$
else $y^i = y^{i-1} - \alpha^i_{cg} p^i$ is not feasible but we can find $\widetilde{\alpha}^i$ so that
$\lambda^{k+1} = y^i - \widetilde{\alpha}^i p^i$ is feasible and $\mathcal{A}(\lambda^k) \notin \mathcal{A}(\lambda^{k+1})$

The resulting algorithm has name feasible proportioning [**?**].

To touch up the performance of conjugate gradient method it is possible through the preconditioning consisting in finding a suitable matrix C^{-1} that aproximates in our case matrix F^{-1}, so that modified system is more easy to solve than the original one. The use of lumped preconditioner in the form $C^{-1} = PBAB^{T}P + Q$ offers for this purpose.

Chapter 7

PETSc 2.0 - Equipment for Parallel Implementation

Before the description of the parallel implementation I would like to introduce first a tool, that was instrumental in all the realization - its name is PETSc. What does the word "PETSc" mean? The name PETSc is the abbreviation formed by the initial letters of the Portable Extensible Toolkit for Scientific Computation, developed by Argonne National Laboratory by group formed by Balay, Gropp, McInnes and Smith [10].

PETSc is a suite of data structures and routines that provide the building blocks for the implementation of large-scale application codes on parallel and sequential computers especially used for the numerical solution of partial differential equations and related problems on high-performance computers. A number of parallel linear and non-linear equation solvers and unconstrained minimization modules using modern programming paradigms enables development of large scientific codes written in C, C++ or Fortran. PETSc 2.0 uses the MPI standard for all message-passing communication and routines from BLAS, LAPACK, MINPACK, SPARSPAK and BlockSolve95. Technique of object oriented programming provides enormous flexibility and code reuse. The library is hierarchically organized according to level of abstraction (see Figure 7.1).

PETSc consists of a variety of components, which manipulate a particular family of objects. These components are:

- index sets
- vectors
- matrices (sparse and dense)
- distributed arrays (useful for parallelizing regular grid-based problems)
- Krylov subspace methods

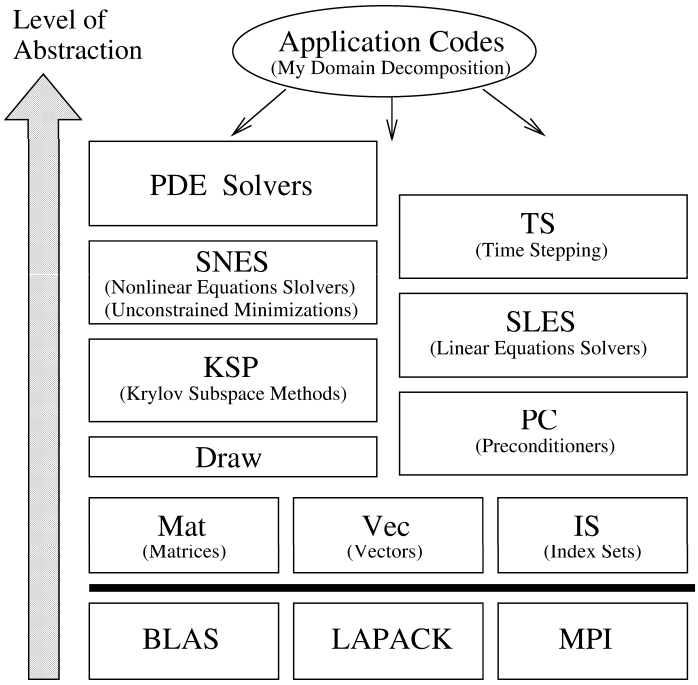


Figure 7.1: Organization of the PETSc Library

- preconditioners
- non-linear solvers
- unconstrained minimization
- timesteppers for solving time dependent PDEs
- graphics devices.

Each of these components consists of an abstract interface and one or more implementations using particular data structures. Thus PETSc provides clean and effective codes for various phases of solving PDEs, with a uniform approach for each class of problems, as well as a rich environment for modeling scientific applications and for algorithm design and prototyping.

Chapter 8

Parallel Implementation

This chapter gives a simplified overview of the most relevant phases of the algorithm and the description of their implementations.

The program consists of two parts, namely from generator of input objects (see function `Chess()` in the Appendix) and from the solver of model problem given by these input objects (see function `FETI()` in the Appendix).

Each processor works with local part associated with its subdomains. For simplification let $X^{[rank]}$ denote this local part or the restriction of object X on current processor specifying by $rank$ ($rank = 0, ..., size - 1$, with $size$ denoting number of processors being at disposal in communicator for our computation).

Most of computations appearing in this program are purely local and therefore parallelizable, but some operations requires data transfers. I distinguish two kinds of these data transfers in my program, namely:

- data transfer that picks up the contributions from all processors and then distributes the complete result to each of processors - such a transfer is denoted in description of parallel scheme as $\lceil \cdot \rceil$ (e.g. computation of dot products of parallel distributed vectors, computation of the norm of parallel vector, matrix by vector multiplication $G\lambda$ etc.)

- data transfer that picks up the contributions from all processors and then distributes the appurtenant part of result to appurtenant processor - such a transfer is denoted in description of parallel scheme as $\lfloor \cdot \rfloor$ (e.g. assembling of sequential vector on one of processors from particular results located on other processors and conversion of this sequential vector by means of scatter functions into parallel vector - matrix by vector multiplication Bf etc.)

8.1 Objects Defining Model Problem

I would like first to describe a part generating input objects - stiffness matrix A, matrix B describing the interconnectivity of subdomains, vector of forces f and matrix

R representing the null space of A. These objects are distributed over the processors, so that their localy stored portions of the same size correspond to relevant subdomains. The data distribution of various types of objects over the processors showing the local portions is then presented in the Figure 8.1 - matrices and vectors are simply divided into $size$-equal length segments. Let N denote the primal dimension (number of primal variables) and N_{dual} denote the dual dimension (number of dual variables size of vector of Lagrange multipliers).

The allocation of memory needed for storage of these matrices including distribution is realized by PETSc function

```
MatCreateMPIAIJ(PETSC_COMM_WORLD,int m,int n,int M,int N,
                int dnz,int *dnz,int onz,int *onz,Mat *mat)
```

where `PETSC_COMM_WORLD` is a communicator comprising all processors being at disposal for computation, then follows numbers of local rows `m`, local columns `n`, global rows `M` and global columns `N`, parameters `dnz,*dnz,onz,*onz` have to do with a control dynamic allocation of matrix memory space, specifying by `MPI` the fact, that `mat` is parallel matrix and by `AIJ` its sparse format (all the matrices used in program are of this format). If the communicator consists only of one processor then following function is used

```
MatCreateSeqAIJ(PETSC_COMM_SELF,int M,int N,
                int nz,int *nz,Mat *mat)
```

only with indications of global dimensions and other parameters of the same meaning as above.

For vectors the situation is analogous - it is possible to use two functions according to size of communicator, namely

```
VecCreateMPI(PETSC_COMM_WORLD,int n,int N,Vec *vec),
VecCreateSeq(PETSC_COMM_SELF,int N,Vec *vec).
```

So the components defining the model problem include:

- stiffness matrix A - global number of rows N, local number of rows stored on processor $N/size$, global number of columns N, local number of columns stored on processor N
- matrix B^T for interconnectivity of subdomains - global number of rows N, local number of rows stored on processor $N/size$, global number of columns N_{dual}, local number of columns stored on processor N_{dual}

- matrix R^T of null space of A - global number of rows N, local number of rows stored on processor $N/size$, global number of columns N_e, local number of columns stored on processor N_e, with N_e denoting the number of floating subdomains
- vector of forces f - global number of elements N, local number of elements stored on processor $N/size$.

Dirichlet boundary conditions for nodes of fixed subdomains are incorporated, so that rows of matrix A corresponding to these nodes are set to 0.0, on diagonal is put 1.0 and the same positions in vector f are set to 0.0.

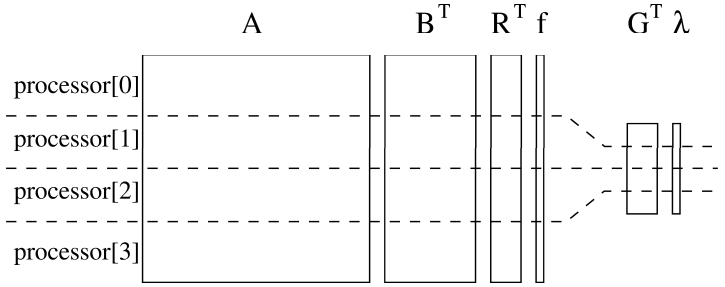


Figure 8.1: Example of data distribution over 4 processors

8.2 Implementation of Dual Formulation and Modifications

Objects presented in previous section are input parameters for domain decomposition solver - function `FETI()`. Before the start of algorithm it is necessary to execute preparatory phase consisting in dual formulation (presented in chapter 4) and modifications (presented in chapter 5). The descriptions and comments of the realizations follow:

- Computation of matrix

$$F = BA^+B^T$$

  - the principle of efficient programming is to avoid time-consuming operations and that is beyond dispute that the matrix by matrix multiplication is one of them. Fortunately, only matrix by vector multiplication crops up in this algorithm and it is performed as $Fvec_d = B(A^+(B^T vec_d))$. The action of A^+ can be evaluated by means of Cholesky or LU decomposition of 2 stiffness submatrices on each processor (factorization of 2 blocks is sufficient in our

case, because there are only 2 types of subdomains - fixed and floating) and the storing of this matrices is also acceptable. These two submatrices can be obtained on each processor by means of PETSc function

```
MatGetSubMatrices(Mat mat,int n,IS *irow,IS *icol,
                  MatGetSubMatrixCall scall,Mat *submat),
```

which exploits on the basis of arrays of index sets (`IS` is data type containing indices) to extract `n` submatrices from a matrix `mat`, `submat` then points to an array of current matrices. In this case two `IS` are created on each processor - `IS[0]` defines global indices of rows and columns of stiffness submatrix first stored on processor - candidate for generalized inverse of fixed subdomain, and `IS[1]` defines global indices of rows and columns of stiffness submatrix last stored on processor - candidate for generalized inverse of floating subdomain, identical to submatrix K_{11} defined in chapter 4 (that means last row and last column of this submatrix are cut off). For purpose of factorization PETSc provides functions

```
MatCholeskyFactor(Mat mat,IS perm,double fill),
MatLUFactor(Mat mat,IS rowperm,IS colperm,double fill)
```

performing in-place Cholesky factorization of a symmetric matrix or in-place LU factorization of matrix respectivly, while index sets define permutations of possible orderings and `fill`>1 is the predicted fill expected in factored matrix, as a ratio of the original fill. Having factored matrices $L[k]$ (such that $A^i = L[k]L[k]^T$, where $k = 0$ if i indicates fixed subdomain and $k = 1$ if i indicates floating subdomain), one can then solve system $A^i x[k] = b[k]$ by means of the function

```
MatSolve(Mat L[k],Vec b[k],Vec x[k]).
```

If it is necessary to compute product $vec_f = A^+ vec_f$ (where $vec_f = B^T vec_d$ denotes vector of the same type as vector f, and vec_d vector of the same type as vector d), the procedure is following: on each processor is created an array `is` of index sets `IS` (for one subdomain one index set of global indices of `vecf` of the same size as corresponding `L[k]`), `is[i]` defines the part from vector `vecf` for scatter into vector `b[k]` at positions defined by index set `isb[k]` (`k=0` if `i` indicates fixed subdomain and `k=1` if `i` indicates floating subdomain), while the following functions are used

```
VecScatterCreate(Vec vecf,Vec b[k],IS is[i],IS isb[k],
                 VecScatter *ctx),
VecScatterBegin(Vec vecf,Vec b[k],INSERT_VALUES,
```

```
                    SCATTER_FORWARD,VecScatter ctx),
VecScatterEnd(Vec vecf,Vec b[k],INSERT_VALUES,
                  SCATTER_FORWARD,VecScatter ctx),
```

followed by solution of system above introduced `MatSolve(Mat L[k],Vec b[k],Vec x[k])`. Acquired solution `x[k]` is then put back into `vecf` at positions defined by index set `is[i]` using calls of scatter functions with changed arguments

```
VecScatterBegin(Vec x[k],Vec vecf,INSERT_VALUES,
                    SCATTER_REVERSE,VecScatter ctx),
VecScatterEnd(Vec x[k],Vec vecf,INSERT_VALUES,
                  SCATTER_REVERSE,VecScatter ctx),
VecScatterDestroy(VecScatter ctx),
```

while the last function destroys a scatter context created by `VecScatterCreate()`. In the case of floating subdomain we have to set the position in vector `vecf` corresponding to cut off row and column to value 0.0, this is value of Schur complement $K_{22}^i - K_{21}^i K_{11}^{i^{-1}} K_{12}^i$ presented in chapter 4. All these operations concerning A^+ are performed on each of processors with own local portions without any communication. The communication is then required for matrix by vector multiplications $vec_f^{[rank]} = B^{T[rank]} \left[vec_d^{[rank]} \right]$ and $vec_d^{[rank]} = \left[B^{[rank]} vec_f^{[rank]} \right]$. The product $Fvec_d$ is computed at least once per cg-iteration and although it is efficiently performed in parallel way, it dominates the overall time because of large primal dimension.

- Computation of matrix

$$\widetilde{G} = RB^T$$

- since the size of the dual problem can be still considerable large, I have decided to parallelize the vector of Lagrange multipliers λ (and of course vectors of the same type denoting as vec_d) and matrix $\widetilde{G}^T$ in such way that each of processors owns nearly the same local portion $\widetilde{G}^{T[rank]}$, namely for vector λ - global number of elements N_{dual}, local number of elements stored on processor $N_{dual}/size$, for matrix $\widetilde{G}^T$- global number of rows N_{dual}, local number of rows stored on processor $N_{dual}/size$, global number of columns N_e, local number of columns stored on processor N_e. Although a matrix by matrix multiplication, as was mentioned above, does not belong to efficient operations, there is no way how to avoid it. Also PETSc provides no function for this multiplication and so nothing else is left, as tranform matrix by matrix multiplication $\widetilde{G}^T = BR^T$ to matrix by vector multiplication $Bvec_f$, while vec_f is vector created by $i-th$ column of matrix R^T and obtained by function

`MatGetColumnVector(Mat RT,Vec vecf,int i),`

the result then forms $i-th$ column of matrix $\widetilde{G}^T$ - fortunately number of columns of matrix R^T is small - equal to number of floating subdomains N_e. All the computation is done in parallel, the communication is required only during the distribution of these N_e result vectors over the processors in communicator. For more details see the function `ComputeMatG()`.
Parallel scheme:

$$\widetilde{G}_{icol}^{T[rank]} = \left\lfloor B^{[rank]} R_{icol}^{T[rank]} \right\rfloor .$$

- Computation of vector

$$\widetilde{d} = BA^{+} f$$

- matrix by vector multiplication is realized according to scheme $A^{+}vec_f$ introduced in item concerning a matrix F. Matrix by vector multiplication $Bvec_f$ is performed in parallel but the communication is necessary to convert the result into the parallel vector $\widetilde{d}$ - for more details see function `BLLvec()`.
Parallel scheme:

$$\widetilde{d}^{[rank]} = \left\lfloor B^{[rank]} \left(A^{+[rank]} f^{[rank]} \right) \right\rfloor .$$

- Computation of vector

$$\widetilde{e} = Rf$$

- in line with the formats of R and f, this matrix by vector multiplication is also parallelized.
Parallel scheme:

$$\widetilde{e} = \left\lceil R^{\lfloor rank \rfloor} f^{\lfloor rank \rfloor} \right\rceil .$$

All presented operations have to be done how in case of basic version so in case of projected and projected-preconditioned one. However, for last two named versions it is necessary to execute still following modifications:

- Computation of matrix

$$G = T\widetilde{G}$$

with T denoting a regular matrix that defines the orthonormalization of the rows of $\widetilde{G}$. Further acceleration of computation reached via projectors built

thanks to matrix G is paid by orthonormalization of columns of matrix $\widetilde{G}^T$. For this purpose the classical Gram-Schidt algorithm was chosen, that appears more suitable for parallelization of this problem than modified or iterated classical Gram-Schmidt [11] (classical Gram-Schmidt requires half the number of floating-point operations, on parallel computers it can be much faster than modified Gram-Schmidt and its parallel efficiency equals that of iterated classical Gram-Schmidt). The columns of matrix $\widetilde{G}^T$ are copied into the array $g[]$ of vectors of type vec_d (local size $N_{dual}/size$, global size N_{dual}) and process of orthonormalization is performed according to

$$g[i] = g[i] - \sum_{j=0}^{i-1}(g[i]^T g[j])g[j], g[i] = \frac{g[i]}{||g[i]||}, i = 0, ..., N_e - 1,$$

while through this way gained vectors form columns of required matrix G^T. For more details see the function `OrthogMatG()`.
Parallel scheme:

$$g[i]^{[rank]} = g[i]^{[rank]} - \sum_{j=0}^{i-1} \left[g[i]^{T[rank]} g[j]^{[rank]}\right] g[j]^{[rank]},$$
$$g[i]^{[rank]} = \frac{g[i]^{[rank]}}{\left[||g[i]^{[rank]}||\right]}, i = 0, ..., N_e - 1.$$

- Computation of vector

$$e = T\widetilde{e}$$

- problem of finding vector e transforms after some manipulations to problem to solve the small system of equations $\widetilde{G}G^T e = \widetilde{e}$, while the product $\widetilde{G}G^T$ is computed in similar way as product BR^T - see the function `ComputeMatGGort()`.

- Computation of vector $\widetilde{\lambda}$ that satisfies relation $G\widetilde{\lambda} = e$, but being aware of features of matrix G, the vector $\widetilde{\lambda}$can be easy obtained as

$$\widetilde{\lambda} = G^T e.$$

Parallel scheme:

$$\widetilde{\lambda}^{[rank]} = G^{T[rank]} e.$$

- Computation of vector

$$d = \widetilde{d} - F\widetilde{\lambda}$$

- vector d is given by difference of vector $\widetilde{d}$ and vector acquired as the the above explained product $F\widetilde{\lambda}$.
Parallel scheme:

$$d^{[rank]} = \widetilde{d}^{[rank]} - \left\lfloor B^{[rank]} \left(A^{+[rank]} \left(B^{T[rank]} \left\lceil \widetilde{\lambda}^{[rank]} \right\rceil \right) \right) \right\rfloor .$$

- Creation of active $\mathcal{A}(\lambda)$ and free set $\mathcal{F}(\lambda)$ - vector J is used for this purpose having ones at positions i belonging to $\mathcal{F}(\lambda)$, where $\lambda_i > c_i$, with $c_i = -\widetilde{\lambda}_i$ for $i \in I$ and $c_i = -\infty$ for $i \in E$, and zeros at positions belonging to $\mathcal{A}(\lambda)$.
  Parallel scheme:

$$J_i^{[rank]} = \begin{cases} 0 & \text{for} \quad \lambda_i^{[rank]} \leq c_i^{[rank]} \\ 1 & \text{for} \quad \lambda_i^{[rank]} > c_i^{[rank]} \end{cases} .$$

The course of computation is appreciably influenced by initial aproximation of vector λ^0 (comparison of various aproximations is shown in Tables 9.2-9.4). I take following aproximations into account in my program:

1. $\lambda_i^{0,\mathrm{I}} = \max\left(0, -\widetilde{\lambda}_i\right)$
2. $\lambda_i^{0,\mathrm{II}} = -\widetilde{\lambda}_i$
3. $\lambda_i^{0,\mathrm{III}} = \max\left(-\widetilde{\lambda}_i, \underline{\lambda}_i\right)$

here we can look at $-\widetilde{\lambda}_I$ as at the hindrance and at

$$\underline{\lambda} = \frac{1}{2} B f$$

computed according to parallel scheme

$$\underline{\lambda}^{[rank]} = \frac{1}{2} \left\lfloor B^{[rank]} f^{[rank]} \right\rfloor$$

as at the optimal estimate of resulting vector λ, taking into cinsideration distribution of forces f on interfaces and arising from relation $B^T\lambda = f$ using orthogonality $BB^T = 2I_B$ with I_B denoting square unit matrix.

8.3 Implementation of Algorithm

Step 0 contains initialization of following needful parameters - parameters for adaptive precision control: $\alpha = 10^{-1}, \beta = 10$, various precisions: $10^{-4}\,\|d\|$, $10^{-4}\,\|e\|$, $\eta_0 = 0.01$, initial vector μ^0 and vector c, which serves for comparison with vector λ^k and subsequent update of active $\mathcal{A}(\lambda)$ and free $\mathcal{F}(\lambda)$ sets (problem dependent parameters M, ρ_0 and Γ are set by user from command line).

Step 1 consists of QP problem to find λ^k so that $\left\|g^P(\lambda^k, \mu^k, \rho_k)\right\| \leq M \left\|\widetilde{G}\lambda^k - \widetilde{e}\right\|$ (or $\left\|g^P(\lambda^k, \mu^k, \rho_k)\right\| \leq M \left\|G\lambda^k\right\|$) and it is realized by function `QP()`. This step contains:

- Computation of vector

$$b = \widetilde{d} - \widetilde{G}^T(\mu - \rho\widetilde{e}) \quad \text{or} \quad b = Pd - G^T\mu$$

for basic version or for projected and projected-preconditioned versions respectively.
Parallel scheme:

$$\begin{aligned} b^{[rank]} &= \widetilde{d}^{[rank]} - \widetilde{G}^{T[rank]}(\mu - \rho\widetilde{e}) \quad \text{or} \\ b^{[rank]} &= d^{[rank]} - G^{T[rank]}\left\lceil G^{[rank]}d^{[rank]}\right\rceil - G^{T[rank]}\mu. \end{aligned}$$

- Computation of residuum

$$r = (F + \rho\widetilde{G}^T\widetilde{G})\lambda - b \quad \text{or} \quad r = (PFP + \rho Q)\lambda - b$$

for basic version or for projected and projected-preconditioned versions respectively (with relevant vector b). The product $(F + \rho\widetilde{G}^T\widetilde{G})\lambda$ or $(PFP + \rho Q)\lambda$ is computed by the function `FrhoGGvec()` employing the function `BLLvec()`.
Parallel scheme:

$$\begin{aligned} r^{[rank]} &= \left\lfloor B^{[rank]}\left(A^{+[rank]}\left(B^{T[rank]}\left\lceil\lambda^{[rank]}\right\rceil\right)\right)\right\rfloor + \\ &\quad +\rho\left(\widetilde{G}^{T[rank]}\left\lceil\widetilde{G}^{[rank]}\lambda^{[rank]}\right\rceil\right) - b^{[rank]}, \text{ or} \\ r^{[rank]} &= v^{[rank]} - G^{T[rank]}\left\lceil G^{[rank]}v^{[rank]}\right\rceil + \\ &\quad +\rho\left(G^{T[rank]}\left\lceil G^{[rank]}\lambda^{[rank]}\right\rceil\right) - b^{[rank]}, \\ \text{with } v &= \left\lfloor B^{[rank]}\left(A^{+[rank]}\left(B^{T[rank]}\left\lceil\lambda^{[rank]} - G^{T[rank]}\left\lceil G^{[rank]}\lambda^{[rank]}\right\rceil\right\rceil\right)\right)\right\rfloor \end{aligned}$$

- If $M > 0$ computation of the norm $n_{feas} = \left\|\widetilde{G}\lambda - \widetilde{e}\right\|$in case of basic version or $n_{feas} = \|G\lambda\|$in case of another one else $n_{feas} = 0$ - see function `ComputeNfeas()`.

- Computation of projected g^P, inner g^I and chopped g^C gradients and their norms using the PETSc functions

```
VecPointwiseMult(Vec r,Vec J,Vec gI),
VecNorm(Vec g,Scalar norm)
```

and my function `ChoppGrad()`.
Parallel scheme:

$$\begin{aligned} g^{I[rank]} &= J^{[rank]}.*r^{[rank]}, \\ g_i^{C[rank]} &= \min\left(\overline{J}_i^{[rank]} * r_i^{[rank]}, 0\right), \\ g^{P[rank]} &= g^{I[rank]} + g^{C[rank]}, \end{aligned}$$

with $\overline{J}$ denoting the negation of vector J.

Iteration $k+1$ of QP problem while $\left\|g^P\right\| > \max\left(10^{-4}\left\|\widetilde{d}\right\|, M * \left\|\widetilde{G}\lambda - \widetilde{e}\right\|\right)$ (or $\left\|\begin{matrix} g^P \\ g^C \end{matrix}\right\| > \max\left(10^{-4}\left\|d\right\|, M * \left\|G\lambda\right\|\right)$) (note: this test realizes Step 2) consists in case $\left\|\begin{matrix} g^P \\ g^C \end{matrix}\right\| \leq \Gamma \left\|g^I\right\|$ of following steps:

- Initialization of initial solution y and starting direction vector p

$$y = \lambda, p = g^I$$

  Parallel scheme:

$$y^{[rank]} = \lambda^{[rank]}, p^{[rank]} = g^{I[rank]}.$$

- Performance of CG-iteration $n_{cg}+1$ while $\left\|g^P\right\| > 10^{-4}\left\|\widetilde{d}\right\|$ (or $\left\|g^P\right\| > 10^{-4}\left\|d\right\|$) and $\alpha_{cg} = \widetilde{\alpha}$ and $\left\|g^C\right\| \leq \Gamma \left\|g^I\right\|$ consists in following updates

  $Ap = (F + \rho\widetilde{G}^T\widetilde{G})p$ or $Ap = (PFP + \rho Q)p$
  $pAp = p^T Ap$
  $\alpha_{cg} = r^T p / pAp$
  `Proportioning()` - this function returns values $\widetilde{\alpha}, k_{max}, proc$
  $y = y - \widetilde{\alpha}p$
  $r = r - \widetilde{\alpha}Ap$
  compute g^I, g^C, g^P and their norms
  $w = g^I$ or $w = J. * (C^{-1}g^I)$ for basic and projected or preconditioned version
  $\beta_{cg} = Ap^T w / pAp$
  $p = w - \beta_{cg}p$

  Parallel scheme:

  $Ap^{[rank]} = \left\lfloor B^{[rank]}\left(A^{+[rank]}\left(B^{T[rank]}\left\lceil p^{[rank]}\right\rceil\right)\right)\right\rfloor + \rho\left(\widetilde{G}^{T[rank]}\left\lceil\widetilde{G}^{[rank]}p^{[rank]}\right\rceil\right)$,
  or $Ap^{\lfloor rank\rfloor} = v^{\lfloor rank\rfloor} - G^{T\lfloor rank\rfloor}\left\lceil G^{\lfloor rank\rfloor}v^{\lfloor rank\rfloor}\right\rceil + \rho\left(G^{T\lfloor rank\rfloor}\left\lceil G^{\lfloor rank\rfloor}p^{\lfloor rank\rfloor}\right\rceil\right)$,
  with $v = \left\lfloor B^{[rank]}\left(A^{+[rank]}\left(B^{T[rank]}\left\lceil p^{[rank]} - G^{T[rank]}\left\lceil G^{[rank]}p^{[rank]}\right\rceil\right\rceil\right)\right)\right\rfloor$
  $pAp = \left\lceil p^{T[rank]}Ap^{[rank]}\right\rceil$
  $\alpha_{cg} = \left\lceil r^{T[rank]}p^{[rank]}\right\rceil / pAp$
  `Proportioning()` - this function returns values $\widetilde{\alpha}, k_{max}, proc$
  $y^{\lfloor rank\rfloor} = y^{\lfloor rank\rfloor} - \widetilde{\alpha}p^{\lfloor rank\rfloor}$
  $r^{[rank]} = r^{[rank]} - \widetilde{\alpha}Ap^{[rank]}$
  compute $g^{I[rank]}, g^{C[rank]}, g^{P[rank]}$ and their norms according presented scheme
  $w^{[rank]} = g^{I[rank]}$ for basic and projected version or
  $w = J^{[rank]}. * \left(C^{-1\lfloor rank\rfloor}\left\lceil g^{I[rank]}\right\rceil\right)$ for preconditioned one
  $\beta_{cg} = \left\lceil Ap^{T[rank]}w^{[rank]}\right\rceil / pAp$
  $p^{[rank]} = w^{[rank]} - \beta_{cg}p^{[rank]}$

- Function `Proportioning()` explores the decline of energy functional in $y-\alpha_{cg}p$ compared with the value of energy functional in y - if $\theta(y-\alpha_{cg}p) \leq \theta(y)$ then is used steplength $\widetilde{\alpha} = \alpha_{cg}$ and there is found k_{max} as the largest index i such that $y_i - \alpha_{cg}p_i$ and $J_i = 1$, else if $y_i - \alpha_{cg}p_i < c_i$ and $p_i > 0$ then $k_{max} = i$ and $0 < \widetilde{\alpha} < \alpha_{cg}$ is found as smallest $(y_i - c_i)/p_i$, otherwise $\widetilde{\alpha} = 0$. Return value *proc* indicates *rank* containing index k_{max}. So if $k_{max} \neq -1 \&\& rank = proc$ then $y_{k_{max}} = c_{k_{max}}$.

- Update of vector λ according to

$$\lambda_i = \max(y_i, c_i)$$

  Parallel scheme:

$$\lambda_i^{[rank]} = \max\left(y_i^{[rank]}, c_i^{[rank]}\right).$$

- Update of $\mathcal{A}(\lambda)$ and $\mathcal{F}(\lambda)$ - that means update of vector J realized by function `UpdateSets()`.

$$\begin{aligned}
&\text{if } \lambda_i = c_i \\
&\quad J_i = 0 \\
&\text{else} \\
&\quad J_i = 1 \\
&\quad \lambda_i = \max(\lambda_i, c_i)
\end{aligned}$$

  Parallel scheme:

$$\begin{aligned}
&\text{if } \lambda_i^{[rank]} = c_i^{[rank]} \\
&\quad J_i^{[rank]} = 0 \\
&\text{else} \\
&\quad J_i^{[rank]} = 1 \\
&\quad \lambda_i^{[rank]} = \max(\lambda_i^{[rank]}, c_i^{[rank]})
\end{aligned}$$

- Computation of new residuum

$$r = (F + \rho\widetilde{G}^T\widetilde{G})\lambda - b \quad \text{or} \quad r = (PFP + \rho Q)\lambda - b$$

  according to parallel scheme presented already in Step 1.

- Computation of g^I, g^C, g^P and their norms - already presented

Iteration $k+1$ of QP problem while $\left\|g^P\right\| > \max\left(10^{-4}\left\|\widetilde{d}\right\|, M * \left\|\widetilde{G}\lambda - \widetilde{e}\right\|\right)$ (or while $\left\|g^P\right\| > \max\left(10^{-4}\left\|d\right\|, M * \left\|G\lambda\right\|\right)$) (note: this test realizes Step 2) consists in case $\left\|g^C\right\| > \Gamma\left\|g^I\right\|$ of following steps:

- Initialization of direction vector

$$p_i = \min(r_i, 0)$$

  Parallel scheme:

$$p_i^{[rank]} = \max\left(r_i^{[rank]}, 0\right).$$

- Performance of one CG-iteration $n_{cg} + 1$ concerning following updates

$Ap = (F + \rho\widetilde{G}^T\widetilde{G})p$ or $Ap = (PFP + \rho Q)p$
$pAp = p^T Ap$
$\alpha_{cg} = r^T p / pAp$
$y = y - \alpha_{cg} p$
$r = r - \alpha_{cg} Ap$
update of $\mathcal{A}(\lambda)$ and $\mathcal{F}(\lambda)$ - vector J
compute g^I, g^C, g^P and their norms

  Parallel scheme:

$Ap^{[rank]} = \left\lfloor B^{[rank]}\left(A^{+[rank]}\left(B^{T[rank]}\left\lceil p^{[rank]}\right\rceil\right)\right)\right\rfloor + \rho\left(\widetilde{G}^{T[rank]}\left\lceil\widetilde{G}^{[rank]}p^{[rank]}\right\rceil\right),$
or $Ap^{[rank]} = v^{[rank]} - G^{T[rank]}\left\lceil G^{[rank]}v^{[rank]}\right\rceil + \rho\left(G^{T[rank]}\left\lceil G^{[rank]}p^{[rank]}\right\rceil\right),$
with $v = \left\lfloor B^{[rank]}\left(A^{+[rank]}\left(B^{T[rank]}\left\lceil p^{[rank]} - G^{T[rank]}\left\lceil G^{[rank]}p^{[rank]}\right\rceil\right\rceil\right)\right)\right\rfloor$
$pAp = \left\lceil p^{T[rank]}Ap^{[rank]}\right\rceil$
$\alpha_{cg} = \left\lceil r^{T[rank]}p^{[rank]}\right\rceil / pAp$
$y^{[rank]} = y^{[rank]} - \alpha_{cg}p^{[rank]}$
$r^{[rank]} = r^{[rank]} - \alpha_{cg}Ap^{[rank]}$
update of $\mathcal{A}(\lambda)$ and $\mathcal{F}(\lambda)$ - vector $J^{[rank]}$ according to presented scheme
compute $g^{I[rank]}, g^{C[rank]}, g^{P[rank]}$ and their norms according to presented scheme

Step 3 of Algorithm for simple bound and equality constraints then consists in update of parameters for adaptive precision control and vector of secondary Lagrange multipliers μ.

- Adaptive precision control

if $\left\|\widetilde{G}\lambda - \widetilde{e}\right\| \leq \eta$ (or $\|G\lambda\| \leq \eta$ for proj. and prec. version)
    $\mu = \mu + \rho(\widetilde{G}\lambda)$ (or $\mu = \mu + \rho(G\lambda)$ for proj. and prec. version)
    $\eta = \alpha\eta$
else
    $\rho = \beta\rho$

Parallel scheme: the only communication is required for computation of

$$\left\lceil \widetilde{G}^{[rank]}\lambda^{[rank]} \right\rceil \quad \text{or} \quad \left\lceil G^{[rank]}\lambda^{[rank]} \right\rceil .$$

Steps 1,2,3 are repeated while $\left\|g^P(\lambda,\mu,0)\right\| > 10^{-4}\left\|\widetilde{d}\right\|$ and $\left\|\widetilde{G}\lambda-\widetilde{e}\right\| > 10^{-4}\left\|\widetilde{e}\right\|$ (or while $\left\|g^P(\lambda,\mu,0)\right\| > 10^{-4}\left\|d\right\|$ and $\left\|G\lambda\right\| > 10^{-4}\left\|e\right\|$) . In case of projected or projected-preconditioned version there is executed operation

$$\lambda = \lambda - \lambda^0$$

at the end of algorithm.

Chapter 9

Numerical Experiments

Numerical experiments should confirm the theoretical results and ilustrate the behaviour of the algorithm on model problem presented in chapter 1 with given burden (see Figure 9.1)

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

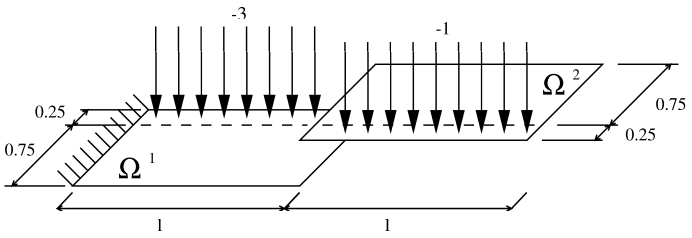


Figure 9.1: Model problem with concrete parameters

Model problem was discretized by the regular grid defined by mesh parameter $h = 1/n$ and with $n+1$ nodes in each direction per subdomain $\Omega^i, i = 1, 2$. Each of these was decomposed into $n_x \times n_y$ rectangles of dimensions $H_x = 1/n_x, H_y = 1/n_y$.

All the experiments were performed on 8-processors parallel computer RISC System/6000 Scalable POWER parallel System with distributed memory = IBM SP2. It consists from nodes, each node creates self-contained Unix workstation with own local memory. The nodes are put through by switch-HPS, ATM and Ethernet interfaces. The nodes are processors POWER2 SuperChip (P2SC) based on architecture RS/6000, which are able to perform 4 floating point operations in one cycle. This computer appears as system of one-processor computers with shared disk space.

During my experiments the IBM SP2 has following configuration

	node1	node2	node3	node4	node5	node6	node7	node8
Type	Thin2	Thin2	Thin	Thin	Thin2	Thin2	Thin2	Thin2
MHz	66.7	66.7	66.7	66.7	160	160	79.2	79.2
MFlops	267	267	267	267				
RAM-MB	128	128	128	128	512	512	256	256
Disk-GB	9	9	9	4.5	4.5	9	9	9
Swap-MB	384	256	256	256	768	768	512	512
DNS	sp1	sp2	sp3	sp4	sp5	sp6	sp7	sp8

I used 1, 2, 4 or 8 processors and have at disposal for this purpose nodes:

- node5
- node5 and node6
- node5, node6, node1 and node2
- node5, node6, node1, node2, node3, node4, node7 and node8

with possibility to use for the communication over the switch the default ip-protocol or to set up us-protocol (user-space) - for the imagery the measured bandwidth between node2 and node3 was with ip-protocol 10.24 MB/sec and with us-protocol 26.22 MB/sec.

The stopping criterium $\left\|g^P(\lambda,\mu,0)\right\| \leq 10^{-4}\,\|d\|$ and $\|G\lambda\| \leq 10^{-4}\,\|e\|$ was used for all presented calculations obtained by basic, projected and projected-preconditioned version of algorithm. The results are then summarized in following tables.

Table 9.1 shows for given parameters the number of iterations in the outer loop of augmented Lagrangian algorithm, number of conjugate gradient iterations in the inner loops for the solution of bound constrained quadratic programming problems and time taking by this computation using for the communication over the switch ip- and then us-protocol, that all for basic, projected and projected-preconditioned version of algorithm. Especially the effect force of projection and preconditioning and difference between the use of ip- and us-protocol are perceptible from this table.

Tables 9.2-9.4 then ilustrate the sensitivity of the algorithm with projection to the choice of problem dependent parameters: the balancing parameter M and penalization parameter ρ_0 and the impact of initial aproximation λ^0. It is possible to say, comparing these three tables, that the most suitable choice are parameters having values $M = 10^2, \rho_0 = 10^2$ and as the best initial aproximation seems to be $\lambda^{0,III}$ derived from the above presented estimate and being that's why the cause of small number of CG-iterations and small number of updates of active and free sets (perceptible from the large number of CG-iterations per face).

Quality of DD-based algorithm hinges on two important properties: *numerical scalability* with respect to the mesh parameter h and the subdomain parameter H,

and *parallel scalability* with respect to the number of processors. The first property characterizes the rate of convergence - number of iterations should decrease with decreasing value H (that means with increasing number of subdomains) and fixed mesh parameter h. The second property characterizes the measure of algorithm's applicability for parallel processing and the fact, that parallel implementation produces larger speedups when larger number of processors are used. It follows that numerical scalability is a necessary condition for parallel scalability. These both scalabilities of this algorithm are demonstrated in the Table 9.5 and 9.6 (with respect to the ratio of the decomposition parameter H and discretization parameter h, with observable time dependence to number of processors), and graphicaly ilustrated in Figures . This graphs show necessity of finding suitable size of "coarse grid" problem given by subdomain parameter H that is large enough to accelerate convergence and small enough to keep all additional computations and first of all interprocessor communications acceptable.

Solutions of some problems are then for ilustration drawn near in the Figures 9.2 (the solution corresponding to domain decomposition presented in Figure 3.3) and 9.3.

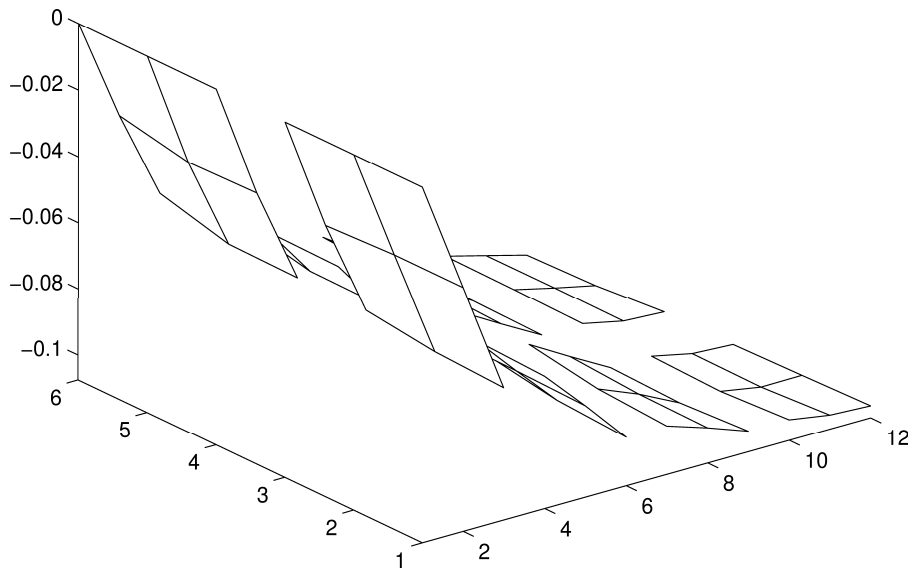


Figure 9.2: Solution of model problem with $h = 1/2$ and $H = 1$

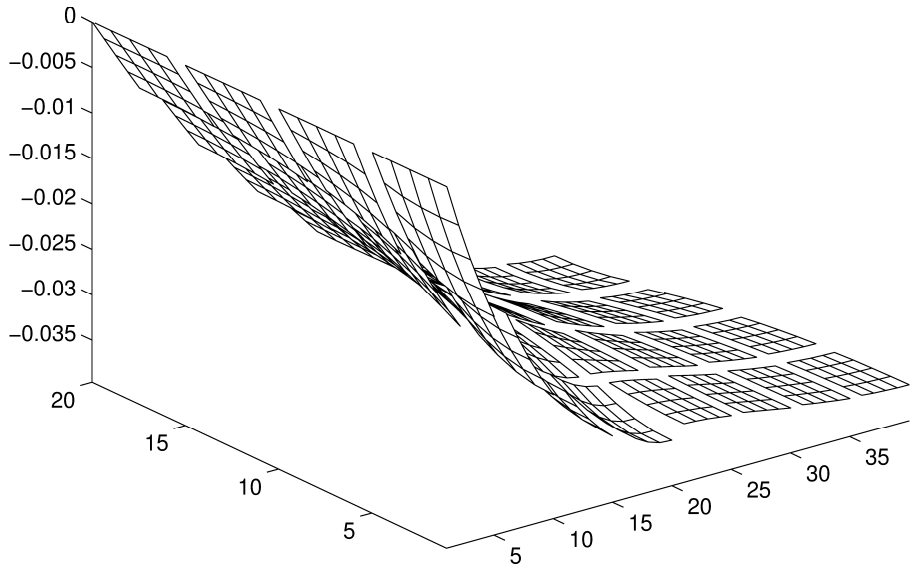


Figure 9.3: Solution of model problem with $h = 1/16$ and $H = 1/4$

Table 9.1: Comparison of basic, projected, projected-preconditioned version with ip & us protocol for regular decomposition $H = H_x = H_y$ with parameters $M = 10^2, \rho_0 = 10^2, \Gamma = 1$ on 1,2,4,8 processors with initial aproximation $\lambda^{0,\mathrm{III}}$

h	H	primal dim.	dual dim.	FETI version	procs	out. iter.	cg. iter.	cg.iter. /face	time ip	time us
1/16	1	578	17	basic	1	6	26	3.25	0.11	0.10
					2	6	28	3.5	1.35	0.23
				proj.	1	3	13	2.6	0.05	0.05
					2	3	13	2.6	0.43	0.11
				projprec.	1	3	14	2.8	0.06	0.06
					2	3	14	2.8	0.53	0.15
1/32	1/2	2312	167	basic	1	5	95	8.6	0.62	0.62
					2	5	88	8.0	2.43	0.83
					4	5	88	8.8	7.41	1.63
				proj.	1	4	34	3.8	0.32	0.32
					2	4	34	3.8	1.15	0.43
					4	4	34	3.8	3.01	0.87
				projprec.	1	4	29	3.2	0.34	0.34
					2	4	29	3.2	1.31	0.50
					4	4	29	3.2	3.11	1.00
1/64	1/4	9248	863	basic	1	6	550	26.2	12.20	12.08
					2	6	546	26.0	26.69	12.80
					4	6	549	26.1	55.10	21.42
					8	6	539	25.7	89.66	25.55
				proj.	1	4	32	8.0	2.94	2.94
					2	4	32	8.0	6.46	1.98
					4	4	32	8.0	5.11	2.96
					8	4	32	8.0	6.93	2.51
				projprec.	1	4	27	6.8	3.08	3.05
					2	4	27	6.8	5.55	2.21
					4	4	27	6.8	6.17	2.80
					8	4	27	6.8	8.21	3.02

Table 9.2: Comparison of the impact of given problem dependent parameters ρ_0 and M for projected version with us-protocol for regular decomposition for $h = 1/128$ and $H = 1/4$ (primal dimension 34848, dual dimension 1695) on 8 processors with initial aproximation $\lambda^{0,\mathrm{I}}$

M	ρ_0	out. iter.	cg. iter.	cg.iter. /face	time us
10^0	10^0	8	33	3.7	8.30
	10^1	7	36	4.0	9.12
	10^2	6	37	4.1	8.28
	10^3	5	40	5.0	10.72
10^1	10^1	6	36	5.1	9.59
	10^2	5	34	4.9	7.91
	10^3	4	38	5.4	8.26
	10^4	3	39	6.5	12.29
10^2	10^2	4	35	5.8	7.60
	10^3	3	37	7.4	9.36
	10^4	2	37	7.4	7.94
	10^5	1	38	9.5	11.62
10^3	10^3	3	34	8.5	7.79
	10^4	2	35	8.8	7.78
	10^5	1	36	9.0	10.61

Table 9.3: Comparison of the impact of given problem dependent parameters ρ_0 and M for projected version with us-protocol for regular decomposition for $h = 1/128$ and $H = 1/4$ (primal dimension 34848, dual dimension 1695) on 8 processors with initial aproximation $\lambda^{0,\mathrm{II}}$

M	ρ_0	out. iter.	cg. iter.	cg.iter. /face	time us
10^0	10^0	8	57	2.0	14.08
	10^1	7	59	2.0	12.38
	10^2	6	69	2.3	13.48
	10^3	5	77	3.0	14.20
10^1	10^1	6	64	2.6	12.85
	10^2	5	67	2.4	13.12
	10^3	4	77	3.0	13.15
	10^4	3	159	2.3	27.05
10^2	10^2	4	71	3.1	12.77
	10^3	3	71	3.1	17.60
	10^4	2	157	2.3	26.66
	10^5	1	296	2.3	50.58
10^3	10^3	2	155	2.3	26.26
	10^4	2	94	2.4	16.68
	10^5	1	332	2.3	73.61

Table 9.4: Comparison of the impact of given problem dependent parameters ρ_0 and M for projected version with us-protocol for regular decomposition for $h = 1/128$ and $H = 1/4$ (primal dimension 34848, dual dimension 1695) on 8 processors with initial aproximation $\lambda^{0,\mathrm{III}}$

M	ρ_0	out. iter.	cg. iter.	cg.iter. /face	time us
10^0	10^0	8	46	5.1	8.31
	10^1	7	40	5.0	8.03
	10^2	6	40	5.0	7.54
	10^3	5	43	6.1	7.23
10^1	10^1	6	39	6.5	7.12
	10^2	5	39	6.5	6.87
	10^3	4	41	7.0	7.25
	10^4	3	75	3.0	14.71
10^2	10^2	4	40	10.0	6.64
	10^3	3	40	10.0	6.65
	10^4	2	66	2.8	14.08
	10^5	1	40	13.3	6.49
10^3	10^3	3	68	3.0	12.16
	10^4	2	73	3.0	12.46
	10^5	1	40	13.3	8.87

Table 9.5: Comparison of projected, projected-preconditioned version with us-protocol for regular decomposition $H = H_x = H_y$ with parameters $M = 10^2, \rho_0 = 10^2, \Gamma = 1$ on 1,2,4,8 processors with initial aproximation $\lambda^{0,\mathrm{III}}$

h	H	primal dim.	dual dim.	FETI version	procs	out. iter.	cg. iter.	cg.iter. /face	time us
1/64	1	8450	65	proj.	1	3	25	2.3	3.70
					2	3	25	2.3	2.16
				projprec.	1	3	31	2.2	4.35
					2	3	31	2.2	2.67
1/64	1/2	8712	327	proj.	1	4	44	3.4	2.32
					2	4	44	3.4	1.67
					4	4	44	3.4	3.07
				projprec.	1	4	45	3.2	2.58
					2	4	45	3.2	2.01
					4	4	45	3.2	3.22
1/64	1/4	9248	863	proj.	1	4	32	8.0	2.94
					2	4	32	8.0	1.99
					4	4	32	8.0	2.43
					8	4	32	8.0	2.63
				projprec.	1	4	27	6.8	3.06
					2	4	27	6.8	2.21
					4	4	27	6.8	2.90
					8	4	27	6.8	3.00
1/64	1/8	10368	1983	proj	1	4	37	9.1	42.85
					2	4	37	9.1	21.44
					4	4	36	9.0	23.13
					8	4	34	8.5	13.83
				projprec	1	4	33	6.6	52.11
					2	4	33	6.6	20.84
					4	4	33	6.6	22.96
					8	4	33	6.6	14.47

Table 9.6: Comparison of projected, projected-preconditioned version with us-protocol for regular decomposition $H = H_x = H_y$ with parameters $M = 10^2, \rho_0 = 10^2, \Gamma = 1$ on 1,2,4,8 processors with initial aproximation $\lambda^{0,\mathrm{III}}$

h	H	primal dim.	dual dim.	FETI version	procs	out. iter.	cg. iter.	cg.iter. /face	time us
1/128	1	33282	129	proj	1				
					2	3	40	1.9	25.31
				projprec	1				
					2	3	54	2.5	29.97
1/128	1/2	33800	647	proj	1				
					2	3	62	3.0	11.97
					4	3	62	3.0	14.67
				projprec	1				
					2	3	54	2.7	12.25
					4	3	54	2.7	14.42
1/128	1/4	34848	1695	proj	1				
					2	4	38	9.5	7.65
					4	4	40	10.0	8.25
					8	4	40	10.0	6.64
				projprec	1				
					2	4	38	7.6	8.97
					4	4	38	7.6	9.88
					8				
1/256	1/1	132098	257	proj	1				
					2	3	62	1.8	399.8
					4				
					8				
1/256	1/2	133128	1287	proj	1				
					2				
					4				
					8				
1/256	1/4	135200	3359	proj.	4	4	48	8.0	49.23
					8	4	47	7.8	31.25
1/512	1/8	540800	14975	proj.	8				316.13

Chapter 10

Conclusion

Untill recently the complicated problems were very difficult and lengthy to solve. The presented algorithm is able to solve problems with hundreds thousands unknowns in a few secondes. At practical testing of functionality of this program the fact was proved, that the algorithm designed by prof. Dostál appears as the most effective tool for solution of variational inequalities and the genius of this idea was fully confirmed. The problems as e.g. computation of the tension of system of elastic bodies in contact or contact shape optimization would not be effective solved without this method, but that is not true in this time. From presented tables the speed up of computations with more processors is good seen and that's why next apparent improvements consists in the use of more processors. I am very pleased to contribute perhaps to the realization of this method with my, even small part.

Epilogue

I have started my work with the citation from the book Theory and Practice of Numerical Reckonning. I would like to finish it in the same mind: "If we are not appropriate practised in numerical reckonning, we get as a rule bad results. The mental depression appearing in consequence of made mistakes becomes a source of new mistakes." My large desire is just this depression to be unknown for us during our work.

Bibliography

[1] V. Láska, V. Hruška, Theory and Practice of Numerical Reckonning, Prague 1927.

[2] Z. Dostál, F. A. M. Gomes Neto, S. A. Santos, Duality-based Domain Decomposition with Natural Coarse-space for Variational Inequalities.

[3] G. Duvant, J. L. Lions, Inequalities in Mechanics and Physics, Springer Verlag, Berlin 1976.

[4] I. Hlaváček, J. Haslinger, J. Nečas, J. Lovíšek, Solution of Variational Inequalities in Mechanics, Springer Verlag, Berlin 1988.

[5] Ch. Farhat, P. Chen, F. Risler, F. X. Roux, Simple and Unified Framework for Accelarating the Convergence of Iterative Substructuring Methods with Lagrange Multipliers, SIAM J. Sci. Stat. Comput.

[6] C. Farhat, J. Mandel, F. X. Roux, Optimal convergence properties of the FETI domain decomposition method, Comput. Methods Appl. Mech. Eng. 115, 1994, 365-385

[7] O. Axelson, A class of iterative methods for finite element equations, Comp. Meth. in. Appl. Mech. and Eng., 9, 1976, 127-137

[8] A. R. Conn, N. I. M. Gould, Ph. L. Toint, A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds, SIAM J. Num. Anal. 28, 1991, 545-572

[9] Z. Dostál, Box constrained quadratic programming with proportioning and projections, SIAM J. Opt.. 7 (1997), 871-887.

[10] S. Balay, W. Gropp, L. C. McInnes, B. Smith, PETSc 2.0 Users Manual, http://www.mcs.anl.gov/petsc/, Argonne National Laboratory.

[11] F. J. Lingen, Efficient Gram-Schmidt orthonormalization on parallel computers, Research report, Department of Aerospace Engineering, Delft University of Technology, 1999

[12] Parasol, An Integrated Programming Environment for Parallel Sparse Matrix Solvers, Project No. 20160, Onera 1996

[13] Z. Dostál, Duality-based Domain Decomposition with Proportioning for the Solution of Free Boundary Problems, Journal of Computational and Applied Mathematics 63 (1995) 203-208

Appendix

example