UNIVERZITA PALACKÉHO V OLOMOUCI
PŘÍRODOVĚDECKÁ FAKULTA
KATEDRA MATEMATICKÉ ANALÝZY A APLIKACÍ MATEMATIKY

DIPLOMOVÁ PRÁCE

Metody Newtonova typu pro řešení úloh nelineárního programování

Vedoucí práce:
Doc. RNDr. Radek Kučera, Ph.D.
Rok odevzdání: 2010

Vypracovala:
Kristina Rádková
MPM, II. ročník

Prohlášení

Prohlašuji, že jsem vytvořila tuto diplomovou práci samostatně za vedení Doc. RNDr. Radka Kučery, Ph.D. a že jsem v seznamu použité literatury uvedla všechny zdroje použité při zpracování práce.

V Olomouci dne 30. března 2010

Poděkování

Mé poděkování směřuji především vedoucímu této diplomové práce Doc. RNDr. Radku Kučerovi, Ph.D za výbornou spolupráci, ochotu, trpělivost, podporu a čas tomu věnovaný. Dále bych ráda poděkovala RNDr. Jitce Machalové, Ph.D za pomoc a podporu při psaní této práce. Neméně vřelé poděkování věnuji mým rodičům, rodině, Ivanovi směrem vzhůru, Přemkovi, všem mým kamarádům, spolužákům, vyučujícím a samozřejmě pánu Bohu za to, že mohu žít tak krásný a šťastný život. Děkuji.

Contents

Notation

$\mathbb{N}$	set of all natural numbers, e.g. $1, 2, 3, \ldots$
$\mathbb{R}$	set of all real numbers
$\mathbb{R}^n$	set of all vectors $x = (x_1, x_2, \ldots, x_n)^\top$ with real componenets
$\mathbb{R}^{m \times n}$	set of all matrixes with m rows and n columns, with real components
I	identity matrix
0	zero matrix
$C^m(D)$	space of all functions with m continuous (partial) derivatives on D, usually $D \subseteq \mathbb{R}^n$
$f(x) = o(g(x)),\ \text{for } x \to a$	$\lim_{x \to a} f(x)/g(x) = 0$
$B(x^*, r)$	$\{x \in D :\ \|\|x - x^*\|\| \leq r\}$, i.e. the ball of the radius r with the center at the point x^*, where $x^* \in D$ and usually $D \subseteq \mathbb{R}^n$
$L(\mathbb{R}^n, \mathbb{R}^n)$	set of all bounded linear mappings of $\mathbb{R}^n$ into $\mathbb{R}^n$

$A \in \mathbb{R}^{n \times n}$ is called symmetric	if $A^\top = A$
$A \in \mathbb{R}^{n \times n}$ is called positive definite	if $\forall x \neq 0,\ x \in \mathbb{R}^n :\ x^\top A x > 0$

Introduction

Nemyslete na to, co máte dělat.
Neuvažujte, jak to máte dělat.
Šíp vylétne hladce jen tehdy,
když překvapí i samotného lukostřelce.
(zenová moudrost)

Newtonova metoda je jednou z nejefektivnějších iteračních metod pro řešení systémů nelineárních rovnic. Její výhodou je rychlost konvergence, která nám umožňuje vypočítat řešení v několika málo iteracích. Klasická Newtonova metoda se potýká s několika nevýhodami, které se staly motivací pro tuto práci. Hlavním cílem práce je sestrojení efektivního řešiče opírajícího se o vhodnou variantu Newtonovy metody, kterým lze řešit jistou třídu minimalizačních úloh vznikajících při duální formulaci kontakních úloh se třením ve třech prostorových dimenzích (3D). Kontaktní úlohy hrají důležitou roli jak v průmyslových, tak ve zdravotnických aplikacích. Návrhy co nejefektivnějších technik diskretizací a strategií řešení kontaktních úloh jsou nyní velkou výzvou pro matematiky i inženýry.

Práce je rozdělena do čtyř kapitol. První kapitola pojednává o Newtonově metodě. Nejprve je popsána klasická Newtonova metoda a je dokázána věta o její konvergenci. S upozorněním na nevýhody klasické Newtonovy metody kapitola přechází k definici slanting funkcí a k nehladké Newtonově metodě. Stěžejní část této kapitoly tvoří důkazy vět o konvergenci jak klasické, tak nehladké Newtonovy metody.

V druhé části práce je formulována úloha nelineárního programování. Jedná se o minimalizaci kvadratického funkcionálu se separovatelnými kvadratickými omezeními. Je popsáno, jak řešení nelineárního problému přechází elegantním způsobem přes nediferencovatelný problém až k řešení posloupnosti systémů lineárních rovnic. Kapitola vyúst'uje schématem algoritmu nehladké Newtonovy metody s použitím techniky aktivních množin.

Třetí kapitola je věnována implementaci algoritmu a směřuje k jeho co největší efektivitě. Nejprve je odůvodněno rozdělení na dva případy - nesymetrický a sy-

metrický, a jsou navrženy tři varianty implementace. Poté nastupuje detailní popis možných cest, jak řešit vnitřní soustavy. Je popsán Schurův doplněk, metoda bikonjugovaných gradientů, LU rozklad a metoda konjugovaných gradientů. Neméně důležitou část zde tvoří podkapitola o adaptivním řízení přesnosti vnitřních soustav.

Vyvrcholením práce je poslední - čtvrtá kapitola, kde se řeší dva modelové příklady. První příklad v jedné dimenzi slouží hlavně k odladění programu a ověření jeho efektivity. Nyní můžeme přejít k hlavnímu vytyčenému cíli práce, a tím je 3D kontaktní úloha s třením. Stručně uvádíme diskrétní formulaci úlohy s Trescovým třením (po diskretizaci metodou konečných prvků), kterou převádíme do duálního tvaru. Pro tuto úlohu pak porovnáváme tři navržené varianty implementace našeho algoritmu (samozřejmě s tím, že používáme znalosti získané prozkoumáváním příkladu prvního). Pokud se dostanete až sem, dozvíte se, která varianta vyhrává.

Přesto, že hlavním úsilím práce je sepsání a vyladění programu v Matlabu, nenajdete v práci (až na dvě výjimky) žádné matlabovské kódy. Podle mého názoru to k porozumění a přehlednosti práce nijak nepřispívá. V kapitole o implementaci jsou shrnuty všechny hlavní myšlenky, podle kterých je program sestaven. K práci je přiložen CD nosič, který všechny kódy obsahuje. Jelikož se jedná o velmi rozsáhlou problematiku, použila jsem již vytvořené m-fily týkají se formulace problému pomocí metody konečných prvků, zadání dat a některé další. Dále také program QPC vytvořený vedoucím této práce (všechny tyto m-fily poskytnul vedoucí práce). M-fily týkající se všeho výše teoreticky popsaného jsou pak samozřejmě vlastní prací.

Čtenářům této práce přeji dobrou náladu při čtení. Tak tedy hurá do toho a v 0.49999998988999... je hotovo.

The Newton method is one of the most effective iterative methods for solving systems of nonlinear equations. The main advantage is a rapid rate of convergence, which enables us to solve the system in few iterations. As a disadvantage we can mention local convergence results and quite high demands on smoothness. This work is motivated by the effort to appropriately modify (to set lower the demands on smoothness) the Newton method in relation to the problem of nonlinear programming. The goal of this work is the completion of an effective solver based on a suitable variant of the Newton method for solving a particular class of minimization problems arising from the dual formulation of contact problems with friction in 3D. The design of discretization techniques and efficient solution strategies for contact problems is a challenging task from both the engineering and the mathematical point of view.

The work is divided into four chapters. The first chapter deals with the Newton method. First the classical Newton method is described and the theorem of its convergence is proved. After pointing out the disadvatages of the classical Newton method, slanting functions and the Semi-Smooth Newton method are introduced. The convergence of the Semi-Smooth Newton method is proved.

The problem of nonlinear programming is discussed in the second part of this work. We consider a minimization problem of a quadratic functional with separable quadratic constraints. It is argued how to compute the solution to this nonlinear problem by a sequence of linear equations. To this end the algorithm of the Semi-Smooth Newton method is reformulated in an active set terminology.

The third chapter is dedicated to the implementation of the algorithm and leads to its high efficiency. We distinguish two cases - nonsymmetric and symmetric and three implementation variants. It is described in details how to deal with the inner problems - the Schur complement, the bi-conjugate gradient method, LU–factorization and the conjugate gradient method. Moreover, the adaptive precision control is mentioned.

The most important part of this work is the fourth chapter, where two model problems are solved. The first 1D example is used mainly for adjusting and ver-

ifying the efficiency of the program. Now, we are ready to deal with the aim of this work - the 3D contact problem with friction. First we introduce the discrete formulation of the problem with Tresca friction (after finite element discretization) and than we derive its dual formulation. For this problem we compare our three proposed variants of the algorithm (using the gained knowledges from the first example). If you read up to this point, you will find out which variant is the winner.

Although the main result of this work is the Matlab program you will not find (except two) any Matlab code in the text. According to my opinion it does not help to understanding. In the attachment you can find the CD with all m-files. As we are concerned with a vast subject I have used some already completed m–files (the formulation of the problem using finite element method and data, all provided by the supervisor of this work). Furthermore, I have used a program QPC completed by the supervisor of this work.

1 The Newton Method and Its Modifications

The Newton method belongs to numerical methods for a calculation of roots of nonlinear functions. For the sake of simplicity, we will distinguish two cases:

a) $f : \mathbb{R} \to \mathbb{R}$, i.e. we are seeking for $x^* \in \mathbb{R}$ such that $f(x^*) = 0$,

b) $F : \mathbb{R}^n \to \mathbb{R}^n$, i.e. we are seeking for $x^* \in \mathbb{R}^n$ such that $F(x^*) = 0$.

Numerical methods for an approximation of roots of a nonlinear real-valued function are usually iterative. An initial iteration $x^{(0)}$ is chosen and the aim is to generate a sequence of values $\{x^{(k)}\}$ such that

$$\lim_{k\to\infty} x^{(k)} = x^*.$$

Among numerical methods of this type belong, for example, the bisection method, the chord method, the secant method, the Regula Falsi method, the Newton method, etc. The efficiency of the method depends on its rate of convergence.

Definition 1.1

(i) An iterative method generating a sequence $\{x^{(k)}\}$ is of order $p \geq 1$, if

$$\exists C > 0 : \ |x^{(k+1)} - x^*| \leq C|x^{(k)} - x^*|^p, \ \forall k \geq k_0, \ k_0 \in \mathbb{N}. \tag{1.1}$$

(ii) If it holds:

$$\lim_{k\to\infty} \frac{|x^{(k+1)} - x^*|}{|x^{(k)} - x^*|} = 0, \tag{1.2}$$

then we call the convergence superlinear.

Remark 1.1

(i) If $p = 1$, then for the convergence of the sequence $\{x^{(k)}\}$ it is necessary that $C < 1$. If $\lim_{k\to\infty} x^{(k)} = x^*$ and $p = 1$, we call the convergence *linear*. The convegence occurs for any choice of the initial iteration $x^{(0)}$ so that the method is then said to be *globally convergent*.

(ii) If $p = 2$, then we call the convergence *quadratic*. There is no demand on the constant C so that the convergence result depends on the choice of the initial iteration $x^{(0)}$. The method is said to be *locally convergent* in this case.

(iii) The condition (1.2) implies that

$$\forall \varepsilon > 0 : \ |x^{(k+1)} - x^*| \leq \varepsilon |x^{(k)} - x^*|, \ \forall k \geq k_0, \ k_0 \in \mathbb{N}.$$

Comparing with the formula (1.1), we can conclude that the superlinear convergence is a certain compromise between the linear and the quadratic convergence.

1.1 One Nonlinear Equation

Assume that $f \in C^1(\mathcal{I})$, $\mathcal{I} \subseteq \mathbb{R}$, $x^* \in \mathcal{I}$ and $f'(x^*) \neq 0$ (i.e. x^* is a simple root). A sequence of values generated by the Newton method is computed by the following rule:

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}, \quad k = 0, 1, 2, \ldots . \tag{1.3}$$

Geometrically, the iteration $x^{(k+1)}$ is a point of intersection of the tangent constructed in the point $[x^{(k)}, f(x^{(k)})]$ to the graph of the function f with the x–axis.

Let us suppose, moreover, that $f \in C^2(\mathcal{I})$, $f'(x) \neq 0$, $\forall x \in \mathcal{I}$ and that the sequence $\{x^{(k)}\}$ generated by (1.3) converges to x^*. By taking the limit of both sides of (1.3) we obtain $f(x^*) = 0$. Consider the Taylor expansion of the function f in the neighbourhood of $x^{(k)}$ for $x = x^*$, i.e.

$$f(x^*) = 0 = f(x^{(k)}) + f'(x^{(k)})(x^* - x^{(k)}) + \frac{1}{2} f''(\alpha)(x^* - x^{(k)})^2,$$

where α is between x^* and $x^{(k)}$. By dividing this equation by $f'(x^{(k)})$ and inserting (1.3) we get

$$x^* - x^{(k+1)} + (x^* - x^{(k)})^2 \frac{f''(\alpha)}{2f'(x^{(k)})} = 0.$$

We can conclude that

$$|x^* - x^{(k+1)}| = \left|\frac{f''(\alpha)}{2f'(x^{(k)})}\right||x^* - x^{(k)}|^2 \leq C|x^* - x^{(k)}|^2,$$

where $C = \max\limits_{\alpha,x\in\mathcal{I}} \left|\frac{f''(\alpha)}{2f'(x)}\right|$. It is easy to deduce that under the previous assumptions, the Newton method is of the second order.

To ensure the convergence of the Newton method it is necessary to choose the initial iteration $x^{(0)}$ from a subinterval of $\mathcal{I}$. This will be discussed in the next chapter.

Remark 1.2 The way how to aproximate f by a straight line going through the point $[x^{(k)}, f(x^{(k)})]$ is not unique. The Newton method is actually a special case of a substitution of a linear function for the function f:

$$l_k(x) := f(x^{(k)}) + (x - x^{(k)})q_k,$$

where the slope q_k is chosen as

$$q_k := f'(x^{(k)}).$$

Remark 1.3 The generalization of the Newton method for a multiple root x^* with multiplicity m, $m > 1$, is also of the second order. It is given by the following rule

$$x^{(k+1)} = x^{(k)} - m\frac{f(x^{(k)})}{f'(x^{(k)})}, \quad k = 0, 1, 2 \ldots.$$

1.2 System of Nonlinear Equations

The Newton method theory concerened with a system of nonlinear equations is analogical to Section 1.1.

Assume that $F : \mathbb{R}^n \to \mathbb{R}^n$, i.e. $F = (F_1, \ldots, F_n)^\top$, $F \in C^1(D)$, where $D \subseteq \mathbb{R}^n$. We denote by F' the Jacobi matrix of F defined as $F' : \mathbb{R}^n \to \mathbb{R}^{n\times n}$

$$(F'(x))_{ij} = \left(\frac{\partial F_i}{\partial x_j}\right)(x), \quad i, j = 1, 2, \ldots, n.$$

For a chosen initial iteration $x^{(0)}$ we write the Newton method for n-dimensional case as follows

$$x^{(k+1)} = x^{(k)} + \delta x^{(k)}, \quad k = 0, 1, 2, \ldots, \tag{1.4}$$

where $\delta x^{(k)}$ is computed in each step by solving the system of linear equations

$$F'(x^{(k)})\delta x^{(k)} = -F(x^{(k)}). \tag{1.5}$$

Remark 1.4 The main idea how to derive the Newton method is based on the fact, that we substitute a sequence of linear problems $L_k(x) = 0$, $L_k : \mathbb{R}^n \to \mathbb{R}^n$, $k = 0, 1, \ldots$ for the nonlinear problem $F(x) = 0$. The solutions of these linear systems $x^{(k)}$ generate a sequence of values $\{x^{(k)}\}$ that converges to the solution of the problem $F(x) = 0$. Here, $L_k(x)$ are chosen as the linear parts of the Taylor expansion of the function F at the points $x^{(k)}$, i.e.

$$F(x) = F(x^{(k)}) + F'(x^{(k)})(x - x^{(k)}) + o(|x - x^{(k)}|^2)$$

and

$$L_k(x) = F(x^{(k)}) + F'(x^{(k)})(x - x^{(k)}).$$

Putting $L_k(x)$ equal to zero we obtain the formula for the $k + 1$ iteration, i.e.

$$x^{(k+1)} = x^{(k)} - (F'(x^{(k)}))^{-1}F(x^{(k)}).$$

Let us note this formula is equivalent to (1.4) (1.5), where the inverse matrix is replaced by its action computed by solving the linear system.

Theorem 1.1 *Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be a continuously differentiable function in a convex open set $D \subseteq \mathbb{R}^n$ and let $x^* \in D$ be a solution to the nonlinear equation $F(x) = 0$. Suppose that the second derivative of F is continuous at a neighbourhood of x^*. Moreover, suppose that $(F'(x^*))^{-1}$ exists and that there are positive constants R, C and L, such that*

$$||(F'(x^*))^{-1}|| \leq C,$$

and

$$||F'(x) - F'(y)|| \leq L||x - y|| \quad \forall x, y \in B(x^*; R), \tag{1.6}$$

having denoted by the same symbol $||\cdot||$ two consistent norms.

Then, there exists $r > 0$ such that for any initial iteration $x^{(0)} \in B(x^*, r)$, the sequence of values $\{x^{(k)}\}$ generated by (1.4) is uniquely defined and converges to x^* quadratically. Moreover, it holds that

$$||x^{(k+1)} - x^*|| \leq CL||x^{(k)} - x^*||^2. \tag{1.7}$$

Proof: In the statement of the theorem is said that there exists $r > 0$, i.e. r sufficiently small and we choose $x^{(0)}$ in $B(x^*, r)$. (In other words there exists $r > 0$ such that $(F'(x^{(0)}))^{-1}$ exists for any chosen $x^{(0)} \in B(x^*, r)$). Below, we will show that a good choice of r is $r = min(R, 1/(2CL))$.

Firstly, we will demonstrate that for an arbitrary initial iteration $x^{(0)} \in B(x^*, r)$ matrix $(F'(x^{(0)}))^{-1}$ exists. Indeed,

$$||(F'(x^*))^{-1}(F'(x^{(0)}) - F'(x^*))|| \leq ||(F'(x^*))^{-1}||\ ||F'(x^{(0)}) - F'(x^*)|| \leq CLr \leq \frac{1}{2}.$$

Let us set

$$A = I - (F'(x^*))^{-1}F'(x^{(0)}) = -(F'(x^*))^{-1}(F'(x^{(0)}) - F'(x^*)).$$

Then

$$(I - A) = (F'(x^*))^{-1}F'(x^{(0)}) \quad \text{and} \quad (I - A)^{-1} = (F'(x^{(0)}))^{-1}F'(x^*).$$

Therefore Lemma 4.1 (see Appendix) implies that $(F'(x^{(0)}))^{-1}$ exists and it holds

$$||(F'(x^{(0)}))^{-1}|| \leq ||(F'(x^{(0)}))^{-1}F'(x^*)||\ ||(F'(x^*))^{-1}|| \leq$$

$$\leq \frac{||(F'(x^*))^{-1}||}{1 - ||(F'(x^*))^{-1}(F'(x^{(0)}) - F'(x^*))||} \leq 2||(F'(x^*))^{-1}|| \leq 2C.$$

Thus the iteration $x^{(1)}$ is well defined and it holds

$$x^{(1)} - x^* = x^{(0)} - x^* - (F'(x^{(0)}))^{-1}(F(x^{(0)}) - F(x^*)). \tag{1.8}$$

By taking a norm in (1.8) we get

$$||x^{(1)} - x^*|| \leq ||(F'(x^{(0)}))^{-1}||\ ||F(x^*) - F(x^{(0)}) - F'(x^{(0)})(x^* - x^{(0)})||. \tag{1.9}$$

From the Taylor expansion of F it follows that

$$F(x^*) = F(x^{(0)}) \ + \ F'(x^{(0)})(x^* - x^{(0)}) \ +$$

$$+ \ \frac{1}{2}(x^* - x^{(0)})^\top : H_F(x^* + t(x^* - x^{(0)})) : (x^* - x^{(0)}),$$

where $t \in (0,1)$ and H_F is the Hessian matrix of F. From (1.9) we get

$$||x^{(1)} - x^*|| \leq ||(F'(x^{(0)}))^{-1}|| \ ||\frac{1}{2}(x^* - x^{(0)})^\top : H_F(x^* + t(x^* - x^{(0)})) : (x^* - x^{(0)})|| \leq$$

$$\leq 2C\frac{1}{2}||H_F(x^* + t(x^* - x^{(0)}))|| \ ||x^* - x^{(0)}||^2 \leq 2C\frac{L}{2}||x^* - x^{(0)}||^2,$$

as it follows from (1.6) that $||H_F(x)|| \leq L, \ \forall x \in B(x^*; R)$. Thus, (1.7) is proved for $k = 0$.

Because $x^{(0)} \in B(x^*, r)$, we get

$$||x^* - x^{(0)}|| \leq \frac{1}{2CL},$$

so that

$$||x^{(1)} - x^*|| \leq \frac{1}{2}||x^* - x^{(0)}||$$

and therefore $x^{(1)} \in B(x^*, r)$.

By the mathematical induction it is easy to prove that the relationship (1.7) holds for an arbitrary k and that $x^{(k)} \in B(x^*, r)$. The theorem is proved. □

Remark 1.5 If we omit the requirement of an existence and a continuity of the second derivative of F, the statement of Theorem 1.1 is still true. Since the theorem assums that $F \in C^1(D)$ and F' is locally Lipschitz continuous, we can use a generalized Hessian matrix in the Taylor expansion, where this requirement is not needed. For further detailes see [10].

Theorem 1.1 also shows disadvantages of the Newton method. For instance, to achieve the desired convergence of the method it is necessery to choose the initial iteration $x^{(0)}$ 'sufficiently close' to the solution. Secondly, in each step of

the Newton method we need to know the Jacobi matrix that, moreover, has to be nonsingular. In order to overcome these difficulties and to lower the computional cost, many modifications of the Newton method were developed, for instance, the Jacobi matrix is replaced by its approximation or for a given number of iterations is made fixed, its LU-factorization is used. Other methods are used to obtain the first iteration 'sufficiently close' to the solution. The system (1.5) is solved by various iterative methods. Modified Newton methods usually do not reach the second order, but a lower one.

1.3 The Semi-Smooth Newton Method

The classical Newton method assumes that F is differentiable. What are suitable analogies of the Newton method in the case when F is not differentiable?

Remark 1.6 Superlinear convergence of a 'semi-smooth Newton method' have been already proved by assuming that F is locally Lipschitz continuous, and by using the notions of the generalized Jacobian and the semismoothness. The main idea is based on the Rademacher theorem, that states: If $F : R^n \to R^m$ is a locally Lipschitz continuous function, then F is differentiable almost everywhere. Unfortunately, the Rademacher theorem does not hold in function spaces. That is why 'slanting' functions are introduced. Within the bounds of this paper we assume 'only' n-dimensional Euclidean spaces $\mathbb{R}^n$. In spite of that we will work with slanting functions and the theorem that proves the superlinear convergence of the 'semi-smooth Newton method' will be proved without using the Rademacher theorem, see [8].

Let us assume $L(\mathbb{R}^n, \mathbb{R}^n)$, $D \subseteq \mathbb{R}^n$ be an open subset, $F : D \to \mathbb{R}^n$ a function and $y, h \in \mathbb{R}^n$. To better understand a new notion of a particular generalization of the Jacobi matrix, we will recall the definition of the *strong* (*Fréchet*) derivative.

Remark 1.7 We assume the set $L(\mathbb{R}^n, \mathbb{R}^n)$. But it is without any problem to formulate the following theory also on a set $L(X, Y)$, where X and Y are Banach spaces.

Definition 1.2 *We say that the function F has at $y \in D$ strong or the Fréchet derivative if there exists a mapping $F' : D \to L(\mathbb{R}^n, \mathbb{R}^n)$ such that*

$$\lim_{h \to 0} \frac{1}{||h||}(F(y+h) - F(y) - F'(y)h) = 0. \tag{1.10}$$

Remark 1.8 If a function F has the Fréchet derivative at y, the operator F' from the last definition is uniquely defined and it is called the strong or the Fréchet derivative.

Definition 1.3

(i) The function F is called **slantly** *differentiable at $y \in D$ if there exists a mapping $F^\circ : D \to L(\mathbb{R}^n, \mathbb{R}^n)$ such that F° is uniformly bounded in an open neighbourhood of y and*

$$\lim_{h \to 0} \frac{1}{||h||}||F(y+h) - F(y) - F^\circ(y+h)h|| = 0. \tag{1.11}$$

The function F° is called a slanting function for F at y.

(ii) The function F is called slantly differentiable in D if there exists $F^\circ : D \to L(\mathbb{R}^n, \mathbb{R}^n)$ such that F° is a slanting function for F at every point $y \in D$. The function F° is called a slanting function for F in D.

Let us point out some properties of slanting functions:

- If F is continuously differentiable in D and we take $F^\circ(y) := F'(y)$ for all $y \in D$, then F° is a slanting function for F at every point of D.
- A slantly differentiable function F at y can have infinitely many slanting functions at y (See Example 1.2). In general, F° can take the values of F'

except at finite number of points of D, and take arbitrary values at these finite number of points. Then such F° is still a slanting function of F for all points of D. To conclude, if F is continuously differentiable function in D and F° is a slanting function for F in D, then F° coincides with F' except possibly on a set of measure zero.

- A convex combination of two slanting functions of F is also a slanting function of F.
- A slanting function is not continuous in general. See Example 1.2.
- A continuous function is not necessarily slantly differentiable (see Example 1.1). But it holds (see [8]) that a function is slantly differentiable at y if and only if F is Lipschitz continuous at y.

Example 1.1 Let $n = 1$ and

$$F(y) = \begin{cases} \sqrt{(}y) & \text{if } y \geq 0, \\ -\sqrt{(}-y) & \text{if } y < 0. \end{cases}$$

Since $F(h) - F(0) = h/\sqrt{|h|}$, and $\lim_{h\to 0} 1/\sqrt{|h|} \to \infty$, there is no uniformly bounded function F° such that fulfils the relationship (1.11) at $y = 0$.

Example 1.2 Let $n = 1$ and $F(y) = max\{0, y\}$. By inserting in (1.11) we get

$$F^\circ(y) = \begin{cases} 0 & y < 0, \\ 1 & y > 0, \\ \alpha \in \mathbb{R} & y = 0. \end{cases}$$

The first two cases are obvious. Let us have a detailed look at the last one ($y = 0$). For $h > 0$ we get (from the definition) $\frac{h-0-1\cdot h}{h} = 0$. For $h < 0$ we get $\frac{0-0-0\cdot h}{h} = 0$. We see that both cases are equal to zero for $F^\circ(0) = \alpha$, where $\alpha \in \mathbb{R}$ is arbitrary.

The **Semi-Smooth Newton method** is a sequence of values $\{y^{(k)}\}$ generated by the following rule

$$y^{(k+1)} = y^{(k)} - (F^\circ(y^{(k)}))^{-1} F(y^{(k)}), \quad k = 0, 1, 2, \ldots \tag{1.12}$$

Theorem 1.2 *Let $F : \mathbb{R}^n \mapsto \mathbb{R}^n$ be slantly differentiable in D with a slanting function F°. Suppose that $y^* \in D$ is a solution to the nonlinear equation $F(y) = 0$. Assume that $F^\circ(y)$ is nonsingular for all $y \in D$, and there exists a positive constant M such that*

$$||F^\circ(y)^{-1}|| \leq M, \quad \forall y \in D.$$

Then there exists $r > 0$ such that for any initial iteration $y^{(0)} \in B(y^*, r)$ the sequence of values $\{y^{(k)}\}$ generated by (1.12) is defined uniquely and converges to y^* superlinearly.

Proof: Firstly, we prove the following statement: if $\{y^{(k)}\}$ converges to y^*, then it converges superlinearly. We can assume without loss of generality that $y^{(k)} \neq y^*$ for all k, since the statement is trivially satisfied in the opposite situation. The iteration rule (1.12) gives

$$y^{(k+1)} - y^* = y^{(k)} - y^* - (F^\circ(y^{(k)}))^{-1} F(y^{(k)}).$$

Denoting $h^k = y^{(k)} - y^*$ we can write

$$y^{(k+1)} - y^* = (F^\circ(y^{(k)}))^{-1}(F^\circ(y^* + h^k)h^k - F(y^* + h^k) + F(y^*))$$

that implies

$$\frac{||y^{(k+1)} - y^*||}{||y^{(k)} - y^*||} \leq \frac{M||F(y^* + h^k) - F(y^*) - F^\circ(y^* + h^k)h^k||}{||h^k||}. \tag{1.13}$$

As the right-hand side tends to zero by the definition of the slanting function, we arrive at

$$\lim_{k\to\infty} \frac{||y^{(k+1)} - y^*||}{||y^{(k)} - y^*||} = 0.$$

We see that the convergence is indeed superlinear.

Now we prove that the Semi-Smooth Newton method converges for sufficiently small r. The definition of the slaning function gives

$$\forall \varepsilon > 0 \ \exists \delta > 0 \ \forall h \in \mathbb{R}^n : \ ||h|| < \delta:$$

$$\frac{||F(y^* + h) - F(y^*) - F^\circ(y^* + h)h||}{||h||} \leq \varepsilon.$$

Let us choose $\varepsilon := M^{-1}\eta$, where $\eta \in (0,1)$ is arbitrary but fixed. Consider $r \in (0,\delta)$ and $y^{(0)} \in B(y^*,r)$. Then (1.13) yields for $k=0$

$$\frac{||y^{(1)}-y^*||}{||y^{(0)}-y^*||} \leq MM^{-1}\eta = \eta.$$

Therefore $||y^{(1)}-y^*|| \leq \eta||y^{(0)}-y^*|| \leq \eta r$ so that $y^{(1)} \in B(y^*,r)$. By the induction we obtain $||y^{(k)}-y^*|| \leq \eta^k r$, that proves the convergence of $\{y^{(k)}\}$. The theorem is proved. $\square$

2 Inequality-Constrained Quadratic Problems

2.1 Formulation of the Problem

We shall be concerned with solving

$$\left.\begin{array}{ll} \text{minimize} & \frac{1}{2}x^\top Ax - x^\top b \\ \text{subject to} & x_{1,i} \geq l_i,\ x_{2,i}^2 + x_{3,i}^2 \leq g_i^2,\ i = 1, \ldots, m, \\ & x = (x_1^\top, x_2^\top, x_3^\top)^\top \in \mathbb{R}^n, \end{array}\right\} \tag{2.1}$$

where $x_1, x_2, x_3 \in \mathbb{R}^m$, $3m = n$. Here, $A \in \mathbb{R}^{n\times n}$ is a symmetric, positive definite matrix, $b \in \mathbb{R}^n$ and $l, g \in \mathbb{R}^m$.

Theorem 2.1 *There is a unique solution to the problem (2.1).*

Proof: The problem (2.1) is a program with a strictly quadratic objective on a convex set. For further detailes see [15] (chapters 12,16) or [14]. □

It is well known that the solution to (2.1) is fully determined by a system of equalities and inequalities called the *Karush-Kuhn-Tucker* (KKT) conditions. To this end introduce the *Lagrangian* $\mathcal{L} : \mathbb{R}^{n+2m} \to \mathbb{R}$ associated with (2.1) by

$$\mathcal{L}(x, \lambda, \mu) = \frac{1}{2}x^\top Ax - x^\top b + \lambda^\top(l - x_1) + \mu^\top(x_2^2 + x_3^2 - g^2),$$

where $\lambda \in \mathbb{R}^m$, $\mu \in \mathbb{R}^m$ are called *Lagrange multipliers.*

Theorem 2.2 *Let us denote by x^* the solution of (2.1). There exists $\lambda^* \in \mathbb{R}^m$ and $\mu^* \in \mathbb{R}^m$ so that the triplet (x^*, λ^*, μ^*) is a unique saddle-point to $\mathcal{L}$, i.e. it satisfies the following system of the KKT conditions*

$$\left.\begin{array}{l} \nabla_x\mathcal{L}(x, \lambda, \mu) = 0, \\ \nabla_\lambda\mathcal{L}(x, \lambda, \mu) \leq 0,\ \lambda \geq 0,\ \lambda^\top\nabla_\lambda\mathcal{L}(x, \lambda, \mu) = 0, \\ \nabla_\mu\mathcal{L}(x, \lambda, \mu) \leq 0,\ \mu \geq 0,\ \mu^\top\nabla_\mu\mathcal{L}(x, \lambda, \mu) = 0. \end{array}\right\} \tag{2.2}$$

Proof: For detailes see [15] (chapters 12,16) or [14]. □

For convenience, we divide A and b into blocks A_{ij}, b_i, where $i, j \in \{1, 2, 3\}$, consistently with the partition of x into x_1, x_2, x_3. Then (2.2) reads as follows

$$A_{11}x_1 + A_{12}x_2 + A_{13}x_3 - \lambda - b_1 = 0, \tag{2.3}$$

$$A_{21}x_1 + (A_{22} + 2M)x_2 + A_{23}x_3 - b_2 = 0, \tag{2.4}$$

$$A_{31}x_1 + A_{32}x_2 + (A_{33} + 2M)x_3 - b_3 = 0, \tag{2.5}$$

$$l - x_1 \leq 0,\ \lambda \geq 0,\ \lambda^\top(l - x_1) = 0, \tag{2.6}$$

$$(x_2^2 + x_3^2 - g^2) \leq 0,\ \mu \geq 0,\ \mu^\top(x_2^2 + x_3^2 - g^2) = 0, \tag{2.7}$$

where $M \in \mathbb{R}^{m\times m}$, $M = diag(\mu)$ and the second powers are considered componentwisely.

2.2 Application of the Semi-Smooth Newton Method

The idea how to use the Newton method for solving (2.1) is based on reformulating the KKT conditions as a system of equalities. The Newton iterations performed on this new system represent an unconstrained iterative process, however some equalities are typically described by non-differentiable functions. To overcome this difficulty we apply the Semi-Smooth Newton method.

Lemma 2.1

a) The condition (2.6) is (componentwisely) equivalent to

$$\lambda - \max\{0, \lambda + \rho(l - x_1)\} = 0, \quad \forall \rho > 0. \tag{2.8}$$

b) The condition (2.7) is (componentwisely) equivalent to

$$\mu - \max\{0, \mu + \rho(x_2^2 + x_3^2 - g^2)\} = 0, \quad \forall \rho > 0. \tag{2.9}$$

Proof:

a) We want to show that $l - x_1 \leq 0,\ \lambda \geq 0,\ \lambda^\top(l - x_1) = 0 \Leftrightarrow$
$\Leftrightarrow \lambda - \max\{0, \lambda + + \rho(l - x_1)\} = 0, \quad \forall \rho > 0.$

Let us start with the right implication '$\Rightarrow$'. There are two possibilities how to fulfil the complementarity condition $\lambda^\top(l - x_1) = 0$. Firstly, $(l - x_1) = 0$ and $\lambda \geq 0$, then undoubtedly $\lambda - \max\{0, \lambda + \rho(l - x_1)\} = 0$. Secondly, for $(l - x_1) \leq 0$ and $\lambda = 0$ we also get that $\lambda - \max\{0, \lambda + \rho(l - x_1)\} = 0$.

Now we prove the opposite implication '$\Leftarrow$'. If $\max\{0, \lambda + \rho(l - x_1)\} = 0$, then $\lambda = 0$ and $\rho(l - x_1) \leq 0$, so that $(l - x_1) \leq 0$.

If $\max\{0, \lambda + \rho(l - x_1)\} = \lambda + \rho(l - x_1)$, then $\rho(l - x_1) = 0$ and $\lambda \geq 0$, and because $\rho > 0$, then necessarily is $(l - x_1) = 0$.

Moreover $\lambda^\top(l - x_1) = 0$ in all cases.

b) The proof is analogous. □

Therefore the KKT conditions (2.3) - (2.7) are equivalent to the one (vector) equation

$$F(y) = 0, \tag{2.10}$$

with $y = (x^\top, \lambda^\top, \mu^\top)^\top$, where $F : \mathbb{R}^{n+2m} \mapsto \mathbb{R}^{n+2m}$ is determined by (2.3) - (2.5), (2.8) and (2.9) as follows:

$$F(y) = \begin{pmatrix} A_{11}x_1 + A_{12}x_2 + A_{13}x_3 - \lambda - b_1 \\ A_{21}x_1 + (A_{22} + 2M)x_2 + A_{23}x_3 - b_2 \\ \underline{A_{31}x_1 + A_{32}x_2 + (A_{33} + 2M)x_3 - b_3} \\ \lambda - D(\mathcal{A}_b)(\lambda + \rho(l - x_1)) \\ \mu - D(\mathcal{A}_q)(\mu + \rho(x_2^2 + x_3^2 - g^2)) \end{pmatrix}. \tag{2.11}$$

2.3 The Semi-Smooth Newton Method as an Active-Set Algorithm

On the equation (2.10) we now apply the Semi-Smooth Newton method, i.e.

$$F^\circ(y^{(k)})y^{(k+1)} = F^\circ(y^{(k)})y^{(k)} - F(y^{(k)}), \quad k = 0, 1, \ldots \tag{2.12}$$

Below, we present an implementation of (2.12) as an *active-set* algorithm. Let $\mathcal{M} := \{1, 2, \ldots, m\}$ and $\rho > 0$. We define *active-sets* at $y \in \mathbb{R}^{n+2m}$

$$\mathcal{A}_b(y) := \{i \in \mathcal{M} : \lambda_i + \rho(l_i - x_{1,i}) > 0\},$$

$$\mathcal{A}_q(y) := \{i \in \mathcal{M} : \mu_i + \rho(x_{2,i}^2 + x_{3,i}^2 - g_i^2) > 0\},$$

and *inactive-sets* as their complements: $\mathcal{I}_b(y) := \mathcal{M}\backslash\mathcal{A}_b(y)$ and $\mathcal{I}_q(y) := \mathcal{M}\backslash\mathcal{A}_q(y)$. With $\mathcal{S} \subseteq \mathcal{M}$, we associate the diagonal matrix $D(\mathcal{S})$ as follows

$$D(\mathcal{S}) := diag(s_1, \ldots, s_m), \quad s_i = \begin{cases} 1, & \text{if } i \in \mathcal{S}, \\ 0, & \text{if } i \notin \mathcal{S}. \end{cases}$$

In order to simplify the notation we denote $\mathcal{A}_b := \mathcal{A}_b(y)$, $\mathcal{A}_q := \mathcal{A}_q(y)$, $\mathcal{I}_b := \mathcal{I}_b(y)$ and $\mathcal{I}_q := \mathcal{I}_q(y)$.

The value of the slanting function $F^\circ(y)$ can be derived from the last formula by the standard differentiation rules

$$F^\circ(y) = \left(\begin{array}{ccc|cc} A_{11} & A_{12} & A_{13} & -I & 0 \\ A_{21} & A_{22} + 2M & A_{23} & 0 & 2X_2 \\ A_{31} & A_{32} & A_{33} + 2M & 0 & 2X_3 \\ \hline \rho D(\mathcal{A}_b) & 0 & 0 & D(\mathcal{I}_b) & 0 \\ 0 & -2\rho D(\mathcal{A}_q)X_2 & -2\rho D(\mathcal{A}_q)X_3 & 0 & D(\mathcal{I}_g) \end{array}\right), \tag{2.13}$$

where $X_2 = diag(x_2)$, $X_3 = diag(x_3)$, $X_2,\ X_3,\ I,\ 0 \in \mathbb{R}^{m\times m}$. Moreover, the direct computation gives

$$F^\circ(y)y - F(y) = \left(\begin{array}{l} b_1 \\ b_2 + 2X_2\mu \\ b_3 + 2X_3\mu \\ \hline \rho D(\mathcal{A}_b)l \\ -\rho D(\mathcal{A}_q)(x_2^2 + x_3^2 + g^2) \end{array}\right). \tag{2.14}$$

ALGORITHM: SEMI-SMOOTH NEWTON METHOD (SSNM)

Given $x^{(0)} \in \mathbb{R}^n$, $\lambda^{(0)} \in \mathbb{R}^m$, $\mu^{(0)} \in \mathbb{R}^m$ and $tol_x \geq 0$.

STEP 0 Set $k := 0$ and choose $\rho > 0$. (Typically $\rho = 1$.)

STEP 1 Define the active and the inactive sets: $\mathcal{A}_b := \mathcal{A}_b(y^{(k)})$, $\mathcal{A}_q := \mathcal{A}_q(y^{(k)})$, $\mathcal{I}_b := \mathcal{I}_b(y^{(k)})$ and $\mathcal{I}_q := \mathcal{I}_q(y^{(k)})$.

STEP 2 Compute $y^{(k+1)} := ((x^{(k+1)})^\top, (\lambda^{(k+1)})^\top, (\mu^{(k+1)})^\top)^\top$ as the solution to the linear system given by the matrix (2.13) and the right-hand-side vector (2.14), in which $\mu := \mu^{(k)}$, $M = diag(\mu)$, $X_2 = diag(x_2^{(k)})$ and $X_3 = diag(x_3^{(k)})$.

STEP 3 Set $err^{(k)} := ||x^{(k+1)} - x^{(k)}||/(||x^{(k+1)}|| + 1)$. If $err^{(k)} \leq tol_x$, return $x := x^{(k+1)}$, $\lambda := \lambda^{(k+1)}$ and $\mu := \mu^{(k+1)}$.

STEP 4 Set $k := k + 1$ and go to STEP 1 .

---

3 Implementation

For a reasonable implementation of the algorithm (SSNM) it is necessary to think about all possible problems and make particular adjustments. The skeleton of the algorithm will stay untouched. There are plenty of ways how to do it.

Firstly, we will express the main ideas. Secondly, we will describe the most important things from that ideas in detail (solving inner problems).

The matrix performing in (2.13), i.e.

$$F^{\circ}(y) = \left(\begin{array}{ccc|cc} A_{11} & A_{12} & A_{13} & -I & 0 \\ A_{21} & A_{22}+2M & A_{23} & 0 & 2X_2 \\ A_{31} & A_{32} & A_{33}+2M & 0 & 2X_3 \\ \hline \rho D(\mathcal{A}_b) & 0 & 0 & D(\mathcal{I}_b) & 0 \\ 0 & -2\rho D(\mathcal{A}_q)X_2 & -2\rho D(\mathcal{A}_q)X_3 & 0 & D(\mathcal{I}_g) \end{array}\right),$$

will be divided into indicated parts and a new notation will be introduced as

$$\begin{pmatrix} K & B_1^{\top} \\ B_2 & -D \end{pmatrix}.$$

We refer to this matrix as the *generalized saddle-point matrix*.

The right-hand-side vector performing in (2.14), i.e.

$$F^{\circ}(y)y - F(y) = \left(\begin{array}{c} b_1 \\ b_2 + 2X_2\mu \\ b_3 + 2X_3\mu \\ \hline \rho D(\mathcal{A}_b)l \\ -\rho D(\mathcal{A}_q)(x_2^2 + x_3^2 + g^2) \end{array}\right)$$

will again be divided into indicated parts and a following notation will be introduced:

$$\begin{pmatrix} f \\ h \end{pmatrix}.$$

Also the vector of unknowns $y = (x^{\top}, \lambda^{\top}, \mu^{\top})^{\top}$ will be re-marked as $(x^{\top}, z^{\top})^{\top}$, where $x^{\top} = x^{\top}$ and $z^{\top} = (\lambda^{\top}, \mu^{\top})$. Finally, we get the system

$$\begin{pmatrix} K & B_1^{\top} \\ B_2 & -D \end{pmatrix} \begin{pmatrix} x \\ z \end{pmatrix} = \begin{pmatrix} f \\ h \end{pmatrix}. \tag{3.1}$$

We refer to this system as the *generalized saddle-point system* in our case with generally nonsymmetric and nonsingular matrix, where

$$K = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} + 2M & A_{23} \\ A_{31} & A_{32} & A_{33} + 2M \end{pmatrix},$$

$$B_1 = \begin{pmatrix} -I & 0 & 0 \\ 0 & 2X_2 & 2X_3 \end{pmatrix},$$

$$B_2 = \begin{pmatrix} -\rho D(\mathcal{A}_b) & 0 & 0 \\ 0 & 2\rho D(\mathcal{A}_q)X_2 & 2\rho D(\mathcal{A}_q)X_3 \end{pmatrix},$$

$$D = \begin{pmatrix} D(\mathcal{I}_b) & 0 \\ 0 & D(\mathcal{I}_g) \end{pmatrix}$$

and

$$f = \begin{pmatrix} b_1 \\ b_2 + 2X_2\mu \\ b_3 + 2X_3\mu \end{pmatrix},$$

$$h = \begin{pmatrix} -\rho D(\mathcal{A}_b)l \\ \rho D(\mathcal{A}_q)(x_2^2 + x_3^2 + g^2) \end{pmatrix}.$$

Let us modify the algorithm on particular places to make the computional cost as low as possible. At the same time, we require the final tests to be sufficiently exact, i.e. comparable with the tolerance. One of the important characteristics of the efficiency of the algorithm is the number of matrix K^{-1} multilpications, because this is here absolutely the most demanding computional operation. As we can mention, the matrix K is a slight modification of the Hessian matrix A from our problem (2.1). That is why, the number that is taking down (matrix K^{-1} multiplications) is usually called Hessian multiplications. Another characteristic of the efficiency is the number of iterations needed for the convergence of the method.

First and foremost we will modify STEP 2, the computation of $y^{(k+1)} := ((x^{(k+1)})^\top, (\lambda^{(k+1)})^\top, (\mu^{(k+1)})^\top)^\top$ (i.e. the solution to the linear system given by the matrix (2.13) and the right-hand-side vector (2.14)).

3.1 Nonsymmetric Case

As we can see, the matrix $F^{\circ}(y)$ is nonsymmetric. But its structure and the nonsingularity enables us to use the Schur complement reduction. This approach will be described in detail in Subsection 3.3.1. The main benefit of this acces is that we split one huge problem into two smaller ones. If we follow the Schur complement path we get to a system of linear equations with nonsymmetric nonsingular square matrix (inner problem). To solve it effectively an iterative method of bi-conjugate gradient (BiCGSTAB) is used (see Subsection 3.3.2). Furthermore, the LU-factorization of the matrix K is used (see Subsection 3.3.3).

The computation of the LU-factorization is unfortunately quite expensive and it has to be done during every iteration (because there is always another matrix). Another idea how to solve the system from Step 2 is to avoid using the LU-factorization. Moreover, to have the computation still effective we do not use the Schur complement. The idea is to solve the problem directly by using BiCGSTAB algorithm.

3.2 Symmetric Case

BiCGSTAB is slower comparing to the conjugate gradient method (CGM). Whay not to use CGM then?

For using CGM it is necessary to have a symmetric, positive definite matrix. If we remind the notation and have a detailed look at our system with the matrix

$$F^{\circ}(y) = \left(\begin{array}{ccc|cc} A_{11} & A_{12} & A_{13} & -I & 0 \\ A_{21} & A_{22}+2M & A_{23} & 0 & 2X_2 \\ A_{31} & A_{32} & A_{33}+2M & 0 & 2X_3 \\ \hline \rho D(\mathcal{A}_b) & 0 & 0 & D(\mathcal{I}_b) & 0 \\ 0 & -2\rho D(\mathcal{A}_q)X_2 & -2\rho D(\mathcal{A}_q)X_3 & 0 & D(\mathcal{I}_g) \end{array}\right)$$

and the right-hand side vector

$$F^{\circ}(y)y - F(y) = \left(\begin{array}{c} b_1 \\ b_2 + 2X_2\mu \\ b_3 + 2X_3\mu \\ \hline \rho D(\mathcal{A}_b)l \\ -\rho D(\mathcal{A}_q)(x_2^2 + x_3^2 + g^2) \end{array}\right),$$

we can see that the matrix could somehow be made symmetric through adjusting B_1 and B_2. Our changes will influence only the vector z (it means λ and μ).

Firstly, ρ can be omitted. Secondly, the active and the inactive sets, in particular $D(\mathcal{A}_b), D(\mathcal{A}_q), D(\mathcal{I}_b), D(\mathcal{I}_q)$, are complementary, i.e. for $j = b, q$ we have $D(\mathcal{A}_j)D(\mathcal{I}_j) = 0$ and $\forall \alpha \in \mathbb{R}^m$ can be written as $\alpha = D(\mathcal{A}_j)\alpha + D(\mathcal{I}_j)\alpha$. There is no problem to solve the system using only the active sets. For instance, from the equation

$$\rho D(\mathcal{A}_b)x_1 + D(\mathcal{I}_b)\lambda = \rho D(\mathcal{A}_b)l,$$

we get that $D(\mathcal{I}_b)\lambda = 0$. The 'inactive part' of the z–solution must be equal to zero. This is the way how to make the matrix symmetric. Instead of the system (3.1) we get a system

$$\begin{pmatrix} K & \bar{B}^\top \\ \bar{B} & 0 \end{pmatrix} \begin{pmatrix} x \\ \bar{z} \end{pmatrix} = \begin{pmatrix} f \\ \bar{h} \end{pmatrix}. \tag{3.2}$$

The sizes of the matrixes are changed acording to the sizes of active sets. It is essential to do the backward adjustments to the z–solution when interpreting the solution to the problem.

After getting the symmetrized problem, the LU-factorization and the Schur complement reduction (that is why there is x and not $\bar{x}$ in (3.2)) will be used. The arising linear system will be solved by CGM (see Subsection 3.3.4).

To sum up, we have just described three variants how to implement the algorithm (SSNM). We will shortly refer to them as A, B and C, respectively.

3.3 Solving Inner Problems

3.3.1 The Schur Complement

To solve the generalized saddle-point system (3.1), i.e.

$$\begin{pmatrix} K & B_1^\top \\ B_2 & -D \end{pmatrix} \begin{pmatrix} x \\ z \end{pmatrix} = \begin{pmatrix} f \\ h \end{pmatrix},$$

we will show a method based on the *Schur complement* reduction.

We can perceive the system as two equations. From the first equation we express x, insert it in the second equation and express z. So

$$Kx + B_1^\top z = f$$

and therefore

$$x = K^{-1}(f - B_1^\top z). \tag{3.3}$$

We get

$$B_2 x - Dz = h \qquad \Rightarrow \qquad B_2 K^{-1} f - B_2 K^{-1} B_1^\top z - Dz = h,$$

and finally

$$(D + B_2 K^{-1} B_1^\top) z = B_2 K^{-1} f - h.$$

Let us denote

$$S := D + B_2 K^{-1} B_1^\top.$$

We write

$$Sz = B_2 K^{-1} f - h. \tag{3.4}$$

S is called the *Schur complement*. We arrived at a block upper triangular system

$$\begin{pmatrix} K & B_1^\top \\ 0 & S \end{pmatrix} \begin{pmatrix} x \\ z \end{pmatrix} = \begin{pmatrix} f \\ B_2 K^{-1} f - h \end{pmatrix}.$$

Because the matrix K and the saddle-point matrix are nonsingular, we can write the block triangular factorization

$$\begin{pmatrix} K & B_1^\top \\ B_2 & -D \end{pmatrix} = \begin{pmatrix} I & 0 \\ B_2 K^{-1} & I \end{pmatrix} \begin{pmatrix} K & 0 \\ 0 & -S \end{pmatrix} \begin{pmatrix} I & K^{-1} B_1^\top \\ 0 & I \end{pmatrix}.$$

Moreover, we can see that S is also nonsingular.

To conclude, the vector of unknowns $y = (x^\top, z^\top)^\top$ is not computed at once, but as a solution to two systems of smaller sizes. Firstly, we compute z from (3.4) and after that x from (3.3).

3.3.2 BiCGSTAB

For a computation of z from the relation (3.4), i.e. $Sz = B_2K^{-1}f - h$, we use an iterative bi-conjugate gradient method (BiCGSTAB) according to Van der Vorst (1992) see [18], which obeys the following:

$$\text{find } z \in \mathbb{R}^n \text{ such that } \quad Fz = d, \quad \text{where } d \in \mathbb{R}^n \text{ and } F \in \mathbb{R}^{n\times n},$$

where F is (generally) nonsymmetric and nonsingular.

ALGORITHM: BiGCSTAB

Given $z^{(0)} \in \mathbb{R}^n$, $tol_v \geq 0$.

STEP 0 Set $r^{(0)} := d - Fz^{(0)}$, $p^{(0)} := r^{(0)}$, $\tilde{r}^{(0)}$ arbitrary (usually we set $\tilde{r}^{(0)} := r^{(0)}$) and $k := 0$.

STEP 1 If $||r^{(k)}|| > tol_v$, compute:

a) $\tilde{p}^{(k)} := Fp^{(k)}$

b) $\alpha_k := (r^{(k)})^\top \tilde{r}^{(0)} / (\tilde{p}^{(k)})^\top \tilde{r}^{(0)}$

c) $s^{(k)} := r^{(k)} - \alpha_k \tilde{p}^{(k)}$

d) $\tilde{s}^{(k)} := Fs^{(k)}$

e) $\omega_k := (\tilde{s}^{(k)})^\top s^{(k)} / (\tilde{s}^{(k)})^\top \tilde{s}^{(k)}$

f) $z^{(k+1)} := z^{(k)} + \alpha_k p^{(k)} + \omega_k s^{(k)}$

g) $r^{(k+1)} := s^{(k)} - \omega_k \tilde{s}^{(k)}$

h) $\beta_{k+1} := (\alpha_k/\omega_k)(r^{(k+1)})^\top \tilde{r}^{(0)} / (r^{(k)})^\top \tilde{r}^{(0)}$

i) $p^{(k+1)} := r^{(k+1)} + \beta_{k+1}(p^{(k)} - \omega_k \tilde{p}^{(k)})$

j) $k := k + 1$

STEP 2 Otherwise, i.e. if $||r^{(k)}|| \leq tol_v$, end.

Remark 3.1 At the beginning of this algorithm it is necessary to choose an initial iteration $z^{(0)}$. Because it is an inner computation, it will be needed to choose this initial value during each iteration. We can choose it, to make it simple, always as a zero vector, or, to make it more effective, as a zero vector only at the first outer step and then we will always take the computed z from the previous computation.

Remark 3.2 If we compute z according to the algorithm, we do not need to know the Schur complement S explicitely. We are only interested in the result of Schur complement-vector products, which can be computed stepwise. So $S = D + B_2K^{-1}B_1^\top$ and we are interested in the result of Sp. Proceed as follows: let us denote by $y = Dp$ and $p_1 = B_2(K^{-1}(B_1^\top p))$. Then $Sp = y + p_1$.

3.3.3 The LU-factorization

As another computional-cost-saving modification an LU-factorization of the matrix K will be used, i.e. $K = LU$. Due to the symmetry of the matrix K we can use a special case of an LU-factorization, the so called $LDL^\top$-factorization, where the lower triangular matrix L agrees with the matrix L from the classical LU-factorization and D is a diagonal matrix, that has on its diagonal diagonal elements of the upper triangular matrix U from the classical LU-factorization.

During the computation with the matrix K we need to store only the matrix L and the diagonal of the matrix D, which we denote by dD. Let us get back to the computation of x from the formula (3.3), i.e.

$$x = K^{-1}f - K^{-1}B_1^\top z.$$

The expression $K^{-1}f$ will be computed as follows:

$$K^{-1}f = (LDL^\top)^{-1}f = (L^\top)^{-1}D^{-1}L^{-1}f = ((L^\top)^{-1}(D^{-1}(L^{-1}f))).$$

In other words, we are solving a system $Ka = f$. Then

$$\underbrace{L\,D\,\underbrace{L^\top a}_{z}}_{y} = f.$$

And so

$$Ly = f,$$
$$Dz = y,$$
$$L^\top a = z.$$

The Matlab command is:

```
a = (((L\f)./dD)'/L)'.
```

Analogically, we will also compute $K^{-1}B_1^\top z$.

If we return to the Remark 3.2, we can notice, that in this situation it can also be used what was just mentioned. Having $Sp = y + p_1$, where $S = D + B_2K^{-1}B_1^\top$, $y = Dp$ and $p_1 = B_2(K^{-1}(B_1^\top p))$. The Matlab command for p_1 is:

```
p1 = B2*((L\(p'*B1)'./dD)'/L)'.
```

Because $B_1^\top p = (p^\top B_1)^\top$ (less demanding is a transpozition of a vector).

3.3.4 CGM

We shall be considered with a problem

$$\text{find } x \in \mathbb{R}^n \text{ such that } \quad Ax = b, \quad \text{where } b \in \mathbb{R}^n \text{ and } A \in \mathbb{R}^{n\times n} \tag{3.5}$$

is symmetric and positive definite. Below, we will present a so called practical form of the conjugate gradient method (CGM), see [3], chapter 10.

ALGORITHM: CGM

Given $x^{(0)} \in \mathbb{R}^n$, $tol_v \geq 0$.

STEP 0 Set $r^{(0)} := b - Ax^{(0)}$, $p^{(0)} := r^{(0)}$, $\rho^{(0)} := (r^{(0)})^\top r^{(0)}$ and $k := 0$.

STEP 1 If $||r^{(k)}|| > tol_v$, compute:

a) $w^{(k)} := Ap^{(k)}$

b) $\alpha_k := \rho^{(k)}/(p^{(k)})^\top w^{(k)}$

c) $x^{(k+1)} := x^{(k)} + \alpha_k p^{(k)}$

d) $r^{(k+1)} := r^{(k)} - \alpha_k w$

e) $\rho^{(k+1)} := (r^{(k+1)})^\top r^{(k+1)}$

f) $\beta_{k+1} := \rho^{(k+1)}/\rho^{(k)}$

g) $p^{(k+1)} := r^{(k+1)} + \beta_{k+1} p^{(k)}$

j) $k := k + 1$

STEP 2 Otherwise, i.e. if $||r^{(k)}|| \leq tol_v$, end.

Theorem 3.1 *For any initial iteration $x^{(0)} \in \mathbb{R}^n$ the sequence $\{x^{(k)}\}$ generated by CGM converges to the solution x^* of the linear system (3.5) in at most n steps.*

Theorem 3.2 *If A has only r distinct eigenvalues, then the CGM iteration will terminate at the solution in at most r iterations.*

For the proofs of these theorems see [15], chapter 5.

3.4 Adaptive Precision Control

The inner precision tol_v, performing in solving of our inner problems, can be fixed, or can be appropriately changed during each step. This is called *adaptive precision control*. For instance, choose two parametres r_{tol} and c_{fact}, where

$0 < r_{tol} < 1$ and $0 < c_{fact} < 1$ (usually $r_{tol} = 0.01$ and $c_{fact} = 0.9$). Then

$$tol_v^{(k)} := min(r_{tol} \cdot err^{(k-1)}, c_{fact} \cdot tol_v^{(k-1)}),$$

where $err^{(k)} := ||x^{(k+1)} - x^{(k)}||/(||x^{(k+1)}|| + 1)$ (see algorithm SSNM) and $tol_v^{(-1)} = r_{tol}/c_{fact}$.

Note that the inner problems will be solved 'inexactly'. These inexact solutions will influence the convergence of the outer iterations in a negative way (slow down). But in general, the whole problem will be solved more quickly.

4 Numerical Experiments

All our experiments are carried out in Matlab. For any further detailes see m-files on the attached CD.

4.1 Model Problem in 1D

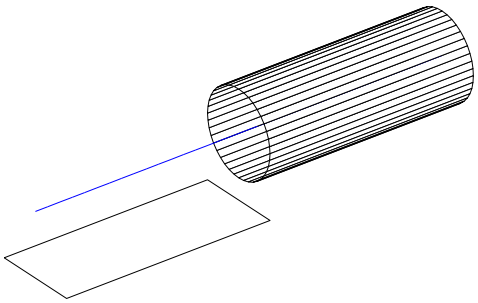
The model problem that is used in our experiments describes a loaded wire that is partially above a plane far off the distance l and partially (in the second half) inside the cylindrical tube of the radius g (see Figure 4.1).

A finite element discretization on a regular grid leads to the algebraic problem

$$\left.\begin{array}{ll} \text{minimize} & \frac{1}{2}x^\top Ax - x^\top b \\ \text{subject to} & x_{1,i} \geq l_i,\ x_{2,i}^2 + x_{3,i}^2 \leq g_i^2,\ i = 1, \ldots, m, \\ & x = (x_1^\top, x_2^\top, x_3^\top, x_4^\top)^\top \in \mathbb{R}^n, \end{array}\right\} \tag{4.1}$$

where $x_1, x_2, x_3, x_4 \in \mathbb{R}^m$, $4m = n$. Here, $A \in \mathbb{R}^{n\times n}$ is symmetric, positive definite matrix, $b \in \mathbb{R}^n$ and $l, g \in \mathbb{R}^m$. We refer to constraints $x_{1,i} \geq l_i$ as *simple bounds*, and to constraints $x_{2,i}^2 + x_{3,i}^2 \leq g_i^2$ as *circular consraints*. The unknown x_4 is not under any constraints. This problem is a slight modification of the problem (2.1).

Figure 4.1 Geometry of the wire.



On the problem (4.1) we apply the Semi-Smooth Newton method. Firstly, we have implemented the algorithm SSNM as it is, without any adjustments. It simply means - using a backslash to solve the linear system from Step 2. Let us have a look at some results of this version.

If it is not stated otherwise, the input data $x^{(0)}$, $\lambda^{(0)}$ and $\mu^{(0)}$ have been chosen as zero vectors, the tolerance tol_x has been set as 10^{-6}, $\rho = 1$ and the final tests are sufficiently exact.

Remark 4.1 Tables 4.1 - 4.4 demonstrates the behaviour of the Semi-Smooth Newton method when solving problem (4.1) with various set constraints l and g. The number n indicates the size of the problem, i.e. the size of the matrix A. Taken down are the ratios of active and inactive sets, i.e. $\mathcal{A}_b : \mathcal{I}_b / \mathcal{A}_q : \mathcal{I}_q$, below them are the numbers of iterations needed for finding the solution with desired tolerance.

Table 4.1
$g = 2$, only simple bounds are active in the solution.

n	$l = -1.5$	$l = -1$	$l = -0.8$	$l = -0.5$	$l = -0.1$	$l = 0$
	0 : 8/0 : 8	1 : 7/0 : 8	1 : 7/0 : 8	3 : 5/0 : 8	4 : 4/0 : 8	5 : 3/0 : 8
32	2	3	5	5	6	6
	0 : 16/0 : 16	1 : 15/0 : 16	1 : 15/0 : 16	4 : 12/0 : 16	6 : 10/0 : 16	9 : 7/0 : 16
64	2	3	6	7	9	10
	0 : 32/0 : 32	1 : 31/0 : 32	3 : 29/0 : 32	6 : 26/0 : 32	12 : 20/0 : 32	19 : 13/0 : 32
128	2	3	8	13	16	16
	0 : 64/0 : 64	1 : 63/0 : 64	4 : 60/0 : 64	10 : 54/0 : 64	23 : 41/0 : 64	37 : 27/0 : 64
256	2	3	15	22	29	30
	0 : 128/0 : 128	1 : 127/0 : 128	8 : 120/0 : 128	21 : 107/0 : 128	45 : 83/0 : 128	73 : 55/0 : 128
512	2	3	27	41	55	58
	0 : 256/0 : 256	1 : 255/0 : 256	16 : 240/0 : 256	40 : 216/0 : 256	89 : 167/0 : 256	146 : 110/0 : 256
1024	2	3	51	79	107	113
	0 : 512/0 : 512	1 : 511/0 : 512	30 : 482/0 : 512	79 : 433/0 : 512	178 : 334/0 : 512	292 : 220/0 : 512
2048	2	3	101	156	211	223

If we choose $l = 0$, $g = 0.3$ and the size of the problem is equal to the number 64 (see Table 4.2), it seems that the method does not converge, in other words, even in 1000 iterations fails to find the solution. If we continue to make higher n or lower g, the situation repeats. All these cases are taken down in Table 4.2 and denoted by symbol $\times$. The difficulty here is probably in the initial iteration, which in not sufficiently close to the solution.

Table 4.2
$l = 0$, both simple bounds and circular constraints are active in the solution.

n	$g = 1.4$	$g = 1$	$g = 0.5$	$g = 0.3$	$g = 0.1$	$g = 0.01$
	5 : 3/2 : 6	6 : 2/4 : 4	7 : 1/5 : 3	7 : 1/5 : 3	8 : 0/8 : 0	8 : 0/8 : 0
32	7	7	8	9	10	14
	10 : 6/2 : 14	11 : 5/5 : 11	13 : 3/6 : 10			16 : 0/16 : 0
64	10	8	9	×	×	62
	20 : 12/4 : 28	22 : 10/5 : 27	26 : 6/10 : 22	29 : 3/16 : 16	31 : 1/22 : 10	
128	16	13	9	16	64	×
	39 : 25/4 : 60	45 : 19/8 : 56	52 : 12/18 : 46			
256	29	22	58	×	×	×
	77 : 51/4 : 124	89 : 39/12 : 116	104 : 24/33 : 95			
512	54	42	27	×	×	×
	155 : 101/6 : 250	177 : 79/22 : 234	208 : 48/60 : 196	228 : 28/95 : 161		
1024	104	82	51	186	×	×
	309 : 203/11 : 501	354 : 158/39 : 473	416 : 96/120 : 392			
2048	206	161	99	×	×	×

For this reason we will find another initial iteration by Polyak-type-algoritm (program QPC, see [12]), which will definitely be sufficiently close to the solution. Moreover, in relation to this new initial iteration $x^{(0)}$ we appropriately modify also other input arguments $\lambda^{(0)}$ and $\mu^{(0)}$.

Remark 4.2 Table 4.3 takes down the behaviour of the Semi-Smooth Newton method with input values modified by QPC. The numbers in round brackets write down the number of iterations that are needed for finding a new initial iteration $x^{(0)}$ by QPC with a tolerance set to 10^{-1}. In some cases, however, it is necessary to compute with a higher tolerance to achieve the convergence of the Semi-Smooth Newton method. In angular brackets are in these cases written down numbers y, where the lowest such a precision is 10^{-y}.

Table 4.4 compares for $l = 0$ and $g = 1$ the Semi-Smooth Newton method with null input data $x^{(0)}$, $\lambda^{(0)}$ and $\mu^{(0)}$, with input data modified by QPC.

4.1.1 Algorithm with Adjustments

In the subsection 3.4 we have familiarized ourselves with the adaptive precision control tol_v. In the mentioned theory there are two optional parametres r_{tol} and

Table 4.3 QPC
$l = 0$, both simple bounds and circular constraints are active in the solution.

n	$g = 0.3$	$g = 0.1$	$g = 0.01$	$g = 0.001$
	7 : 1/5 : 3	8 : 0/8 : 0	8 : 0/8 : 0	8 : 0/8 : 0
32	8(3)	7(3)	19(2)	20(2)
	14 : 2/9 : 7	16 : 0/13 : 3	16 : 0/16 : 0	16 : 0/16 : 0
64	5(6)	4(4)	20(2)	29(3)
	29 : 3/16 : 16	31 : 1/22 : 10	32 : 0/31 : 1	32 : 0/32 : 0
128	3(69)[2]	7(9)	4(5)	20(4)
	57 : 7/26 : 38	61 : 3/40 : 24	64 : 0/58 : 6	64 : 0/64 : 0
256	6(31)	3(99)[2]	5(10)	2(10)
	114 : 14/49 : 79	122 : 6/77 : 51	127 : 1/111 : 17	128 : 0/126 : 2
512	15(114)	9(289)[2]	2(418)[4]	9(11)
	228 : 28/95 : 161	244 : 12/149 : 107	253 : 3/219 : 37	256 : 0/249 : 7
1024	9(373)	14(379)	6(730)[2]	11(774)[5]
	456 : 56/186 : 326	488 : 24/296 : 216	506 : 6/434 : 78	511 : 1/496 : 16
2048	34(809)	15(1026)	10(1558)	2(1897)[6]

c_{fact}. The total efficiency of the computation will be examined through these two parametres. The aim is to find their 'optimal' values. It is obvious, that it is not possible to find all-purpose optimal parametres, i.e. one value of parametres that is optimal for any case. It is necessary to gain particular experience how to handle these parametres. If the size of the problem is n and the number of Hessian multiplications will be n or even smaller, then it is a success.

The conjugate gradient method for solving a linear system of equations certainly converges after n iteration. In our case we deal with a nonlinear problem, so if we get the number of iterations comparable with n, it is a success.

In the 1D example we have firstly implemented our first idea with the Schur complement (A) and we want to explore the adaptive precision control. To verify that it makes sense, we will carry out particular comparative tests. We will still consider the same problem, with $l = 0$ and $g = 0.5$.

In Table 4.5, we monitor the numbers of Hessian multiplications and the number of iterations (needed for the convergence of the method). Two following cases are compared.

a) the inner precision tol_v will be fixed (10^{-6}), the vector $z^{(0)}$ will always be

Table 4.4 QPC
$l = 0$, both simple bounds and circular constraints are active in the solution.

n	$g = 1$	$g = 1(QPC)$
	6 : 2/4 : 4	6 : 2/4 : 4
32	7	8(1)
	11 : 5/5 : 11	11 : 5/5 : 11
64	8	6(1)
	22 : 10/5 : 27	22 : 10/5 : 27
128	13	11(1)
	45 : 19/8 : 56	45 : 19/8 : 56
256	22	4(532)[2]
	89 : 39/12 : 116	89 : 39/12 : 116
512	42	40(1)
	177 : 79/22 : 234	177 : 79/22 : 234
1024	82	80(1)
	354 : 158/39 : 473	354 : 158/39 : 473
2048	161	159(1)

a null vector and $\rho = 1$.

b) the adaptive precision control tol_v will be used with parameters $r_{tol} = 0.01$ and $c_{fact} = 0.9$, the vector $z^{(0)}$ will be a null vector only at the first time and then modified (see Remark 3.1) and $\rho = 0.5$.

Table 4.5
Hessian multiplications/iterations.

n	a)	b)
32	162/8	94/10
64	243/9	128/10
128	391/9	170/10
256	5253/65	341/13
512	2997/29	756/20
1024	7969/49	1317/23
2048	21846/84	10576/68

From the table we can easily read, that the number of Hessian multiplications has been significantly set lower by our modifications. With the growing n the numbers of iterations, are also lower.

In the paper see [13] it is stated, that it is covenient to choose $r_{tol} = 0.01$ and $c_{fact} = 0.9$. For example for $n = 256$ we get 1077/31 (Hessian multiplications/iterations). But if we choose $r_{tol} = 0.8$ and $c_{fact} = 0.82$ we get 81/14. This just proves the fact that it is very tricky to find the optimal values of the parametres. Loads of experiments for this purpose have been done. Have a look at some of them (Table 4.6) to have an idea how to choose the optional parametres r_{tol} and c_{fact}.

Table 4.6
Hessian multiplications/iterations, $r_{tol} = 0.8$.

c_{fact}	$n = 256$	$n = 512$	$n = 1024$
0.9	262/20	188/18	397/17
0.8	146/22	269/17	778/22
0.82	82/14	249/19	350/18
0.7	145/15	476/30	3945/47
0.6	167/19	383/23	7745/53
0.5	135/17	419/19	1723/25
0.4	213/17	259/17	539/17
0.3	296/14	301/13	6278/32
0.2	934/18	1627/19	4654/22
0.1	451/11	1649/15	12926/28
$r_{tol} = 0.01$, $c_{fact} = 0.9$	1077/31	$\times$	2044/30

We have also noticed that there is not a small influence on the results when choosing ρ (from the 'maximum conditions', Lemma 2.1). Have a look at the next table (Table 4.7).

Moreover, we have implemented for the 1D example the second idea (B) - using BiCGSTAB. Next table (Table 4.8) compares variant A with B.

Problems in 3D are definitely (in our case) much more interesting than problems in 1D. Let us move to 3D.

4.2 Model Problem in 3D

As a three-dimensional model problem we will deal with a one-body contact problem with Tresca friction. Let us consider the elastic body represented by the

Table 4.7
Hessian multiplications/iterations, $c_{fact} = 0.9$, $n = 512$.

r_{tol}	$\rho = 1$	$\rho = 0.9$	$\rho = 0.5$	$\rho = 0.1$
0.9	370/22	371/21	314/24	223/17
0.8	188/18	414/16	230/16	184/18
0.7	1391/31	201/17	271/19	192/16
0.6	198/16	337/19	254/14	169/15
0.5	372/22	355/21	271/19	211/19
0.4	49/9[1]	558/15	151/13	281/15
0.3	438/16	396/20	248/14	251/19
0.2	319/13	297/15	248/18	302/18
0.1	409/21	329/17	301/15	218/14
0.05	430/18	497/21	313/19	309/15
0.01	×	686/18	756/20	518/14
0.001	1709/23	1784/24	2218/30	1492/22

Table 4.8
Hessian multiplications/iterations, $c_{fact} = 0.9$.

n	A, $r_{tol} = 0.01$	B, $r_{tol} = 0.01$	B, $r_{tol} = 0.1$
32	94/10	415/11	74/6
64	137/11	910/12	425/13
128	195/11	1064/10	786/12
256	1077/31	3457/19	1529/17
512	×	11534/30	2841/25
1024	2044/30	32062/50	8410/42
2048	5421/49	×	101668/92

prism

$$\Omega = (0,3) \times (0,1) \times (0,1).$$

The boundary $\partial\Omega$ consists of three disjoint parts

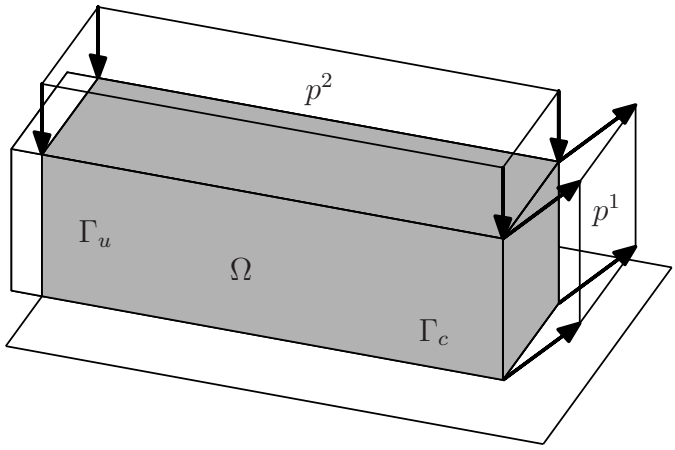
$$\Gamma_u = \{0\} \times (0,1) \times (0,1), \quad \Gamma_c = (0,3) \times (0,1) \times \{0\},$$

$$\Gamma_p = \partial\Omega \backslash (\overline{\Gamma_u \cup \Gamma_c}).$$

On Γ_u we prescribe zero displacements while surface tractions p^j $(j = 1, 2)$ act on Γ_p. On Γ_c we consider the non-penetration condition with respect to the perfectly rigid foundation (an initial gap is equal to zero) and the effect of (isotropic) Tresca friction (see Figure 4.2). Finally we assume that the volume forces are vanishing. The elastic properties of Ω are described by the Lamè equations with material

parameters the Young modulus $E = 2.119 \cdot 10^5$ [MPA] and the Poisson constant $\nu = 0.277$ (steel).

Figure 4.2 Geometry of the body.



After finite element approximation we arrive at the following algebraic minimization problem:

$$\left.\begin{array}{ll} \text{minimize} & \frac{1}{2}u^\top Ku - u^\top f + \sum\limits_{i=1}^{m} g_i||(T_{1,i}u, T_{2,i}u)^\top||_2 \\ \text{subject to} & Nu \leq d, \end{array}\right\} \tag{4.2}$$

where K denotes a symmetric positive definite stiffness matrix, N, T_1 and T_2 ($T_{j,i}$ denotes a i-th row for $j = 1, 2$ and $i = 1, 2, \ldots, m$) are matrices projecting displacements in contact nodes to the normal and tangential directions, respectively, f is the load vector, $g = (g_1, g_2, \ldots, g_m)^\top$ is the vector of slip bound values and the vector d collects distances between Ω and the rigid foundation of the contact nodes. Here, m is the number of contact nodes on Γ_c and $||\cdot||_2$ indicates a norm in $\mathbb{R}^2$.

Due to the fact that the functional $\sum\limits_{i=1}^{m} g_i||(T_{1,i}u, T_{2,i}u)^\top||_2$ is not differentiable and the non-penetration condition is quite complicated, our solution method is based on the dual formulation of (4.2). We use two types of Lagrange multipliers: $\lambda_N \in \mathbb{R}^m$ is associated with the non-penetration condition while $\lambda_{T_1},\ \lambda_{T_2} \in \mathbb{R}^m$ regularize the non-differentiability in the objective function of (4.2). To simplify

the notation we denote

$$\lambda = \begin{pmatrix} \lambda_N \\ \lambda_{T_1} \\ \lambda_{T_2} \end{pmatrix}, \; B = \begin{pmatrix} N \\ T_1 \\ T_2 \end{pmatrix}, \; c = \begin{pmatrix} d \\ 0 \\ 0 \end{pmatrix}.$$

The Lagrangian associated with the problem (4.2) reads as follows

$$\mathcal{L}(u, \lambda) = \frac{1}{2} u^\top K u - u^\top f + \lambda^\top (Bu - c),$$

where u is unconstrained and $\lambda \in \Lambda(g)$ for the set of Lagrange multipliers given as

$$\Lambda(g) = \{\lambda \in \mathbb{R}^{3m} : \; \lambda_{N,i} \geq 0, \; ||(\lambda_{T_1,i}, \lambda_{T_2,i})^\top||_2 \leq g_i, \; i = 1, 2, \ldots, m\}.$$

Then the problem (4.2) is equivalent to the saddle-point problem

$$\min_u \max_{\lambda \in \Lambda(g)} \mathcal{L}(u, \lambda). \tag{4.3}$$

The first unknown may be eliminated from (4.3) by

$$u = K^{-1}(f - B^\top \lambda). \tag{4.4}$$

Substituting (4.4) into (4.3) we obtain the dual problem

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\lambda^\top B K^{-1} B^\top \lambda - \lambda^\top (B K^{-1} f - c) \\ \text{subject to} & \lambda \in \Lambda(g). \end{array}$$

Note that this problem corresponds with (2.1), so that we can solve it by the SSNM. After computing λ we can evaluate u from (4.4).

We took some advantage of exploring the 1D example. Now, we are at most interested to compare the efficiency of our three variants (A, B and C) that have been implemented.

Firstly, we were trying to find the 'optimal' parametres r_{tol} and c_{fact} for each variant. We have concluded that

- for A it is $r_{tol} = 0.2$ and $c_{fact} = 0.4$,

- for B it is $r_{tol} = 0.1$ and $c_{fact} = 0.99$,
- for C it is $r_{tol} = 0.1$ and $c_{fact} = 0.2$.

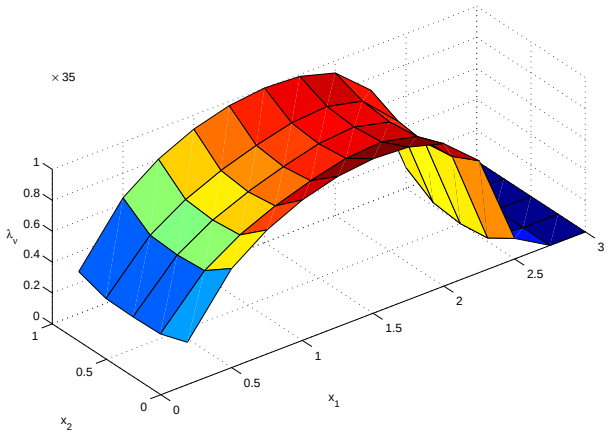
With these choices we have carried out next comparisons, see Table 4.9.

Table 4.9
Hessian multiplications/iterations and times (sec.) above.

n	$nx/ny/nz$	A	B	C
		6.5938	38	3.3125
270	15/5/5	796/16	182/14	802/10
		18.9531	98.4219	10.3906
378	18/6/6	1216/16	1480/26	1308/10
		41.8438	211.9688	22.5156
504	21/7/7	1536/16	1729/25	1596/10
		84.2813	505.9688	49.7344
648	24/8/8	1888/16	2199/21	2246/10
		158.9063	915.2656	107.9219
810	27/9/9	2257/17	NaN	3191/11
		312.0156	1389.8906	176.0156
990	30/10/10	3061/17	NaN	3508/10
		538.1875	2011.1875	331.4531
1188	33/11/11	3679/17	2102/18	4615/11
			5970.4531	
1404	36/12/12	out of memory	4184/20	out of memory

Remark 4.3 The 'NaN' means that there was a dividing by zero in BiCGSTAB, due to rounding errors and we did not get the solution. The 'out of memory' expresses that an ordinary computer is not able to compute the solution because of an unsufficient memory. The $nx,\ ny,\ nz$ in the table describe into how many parts is the body divided in the corresponding directions of axes $x,\ y,\ z$, respectively. The number n indicates the size of the problem.

Figure 4.3 Normal contact stress, C.



In spite of the fact that A needs more iterations to converge than C, the variant A is the most effective. This is because the characteristic Hessian multiplications is more important than iterations when concerning effectivity. (Here, this is caused by the fact that the inner problems for C are solved more exactly than for A). The variant B seems to be good for large problems. But the so called *break down* occurs - this is a problem connected with the rounding errors in BiCGSTAB when using it without the Schur complement reduction and it does not compute the solution.

Appendix

Definition 4.1 *The matrix norm $||\cdot||$ is said to be consistent with the vector norm $||\cdot||$, if*

$$||Ax|| \leq ||A||\ ||x|| \quad \forall x \in \mathbb{R}^n,\ A \in R^{m\times n}.$$

Lemma 4.1 Let $A \in \mathbb{R}^{m\times n}$ and $I \in \mathbb{R}^{m\times n}$ be the identity matrix. If $||A|| < 1$, then $(I - A)^{-1}$ exists and it holds that

$$||(I - A)^{-1}|| \leq \frac{1}{1 - ||A||}.$$

Definition 4.2 *The Hessian matrix of F, i.e. $H_F : \mathbb{R}^n \to \mathbb{R}^{n\times n\times n}$*

$$(H_F(x))_{ijk} = \left(\frac{\partial^2 F_i}{\partial x_j \partial x_k}\right)(x) \quad i, j, k = 1, 2, \ldots, n,$$

for $x = (x_1, \ldots, x_n)^\top$ and $F = (F_1, \ldots, F_n)^\top$.

Remark 4.4 Because H_F is a three-index matrix it is necessary to use a generalized case of the dot product for matrices with different number of indexes, which is denoted by ':'.

Definition 4.3 *The generalization of the definition of the dot product. The case of matrixes with a different number of indexes. If*

$$A = (a_{i_1,\ldots,i_m,j_1,\ldots,j_n})_{i_1\leq k_1,\ldots,i_m\leq k_m,j_1\leq l_1,\ldots,j_n\leq l_n}$$

and

$$B = (b_{j_1,\ldots,j_n})_{j_1\leq l_1,\ldots,j_n\leq l_n},$$

then

$$A : B = (c_{i_1,\ldots,i_m})_{i_1\leq k_1,\ldots,i_m\leq k_m},$$

where

$$c_{i_1,\ldots,i_m} = \sum_{j_1\leq l_1,\ldots,j_n\leq l_n} a_{i_1,\ldots,i_m,j_1,\ldots,j_n} b_{j_1,\ldots,j_n}.$$

Conclusion

To conclude, the goal of this work – the completion of the effective solver for the contact problem with friction – was sucesfully accomplished. The A variant (the nonsymmetric case with the Schur complement reduction) proved to be the most effective variant of the implementation.

After describing the Newton method and its disadvantages, we introduced the slanting functions and the Semi–Smooth Newton method as a possibility how to deal with these disadvantages. Moreover, we have showed how to solve the minimization problem of the quadratic functional with separable constraints. An important part of this work was the chapter devoted to the implementation, where we have presented three variants of the implementation of the algorithm of the Semi–Smooth Newton method. The 1D model example helped us to make our program as effective as possible. Finally, the 3D model problem fulfiled all our expectations we were heading towards during whole work.

Due to a dealing with a vast subject that includes the finite elemnet method, nonlinear programming, contact problems etc., I have found it difficult to decide how much information give to the reader about these subjects that are not really the main topic of this work. Finally, I have decided to give the sufficient and sometimes just basic information about these topics.

Because I have been absorbed in this subject, I would like to occupy myself with this topic even more closely in the future.

Hlavní cíl diplomové práce – sestrojení efektivního řešiče pro kontaktní úlohu se třením, byl uspěšně dosažen. Jako nejefektivnější varianta implementace se ukázala varianta A - nesymetrický případ s použitím rozkladu na Schurův komplement.

V práci jsme se blíže seznámili s Newtonovou metodou a jejími nevýhodami. Jako jedno z možných východisek z těchto problémů byla představena nehladká Newtonova netoda, která využívá slanting funkcí. Ukázali jsme, jak je možné pomocí nehladké Newtonovy metody řešit minimalizační úlohy s kvadratickým funkcionálem a separovatelnými kvadratickými omezeními. Důležitou částí této práce byla kapitola o implementaci, která obsáhla tři implementační varianty algoritmu nehladké Newtonovy metody. Modelový příklad v 1D posloužil jako velmi dobrý pomocník při ověřování teoretických úvah a odlad'ování programu. Závěrečný modelový příklad kontaktní úlohy završil vše, k čemu jsme celou dobu směřovali.

Jelikož tato práce zahrnuje rozsáhlé téma, které v sobě obsahuje několik disciplín (metody konečných prvků, nelineární programování, kontaktní úlohy atd.), bylo poměrně těžké určit, do jaké míry se o problematice přímo nesouvisející s tématem rozepsat. V těchto případech jsem se snažila poskytnout nejpodstatnější informace a nenarušovat tak hlavní tok práce.

Daná problematika mě zaujala natolik, že bych se jí ráda věnovala i v budoucnu. Výzvou je pro mě hlubší pochopení problematiky jako celku. Další velmi lákavou motivací jsou dynamické kontaktní úlohy.

Přestaň myslet: To je mé
a to není mé.
Pak mi pověz, kdo jsi ty?
A také mi řekni,
jak vypadala tvoje tvář,
než se narodili tví rodiče?
(zenová moudrost)

Annexe

Here is some information about the attached CD with m-files. CD includes two files:

1. 1D - 1D model problem

    - no_adjust - algorithm with no adjustments and QPC is added
    - with_adjust - algorithm with adjustments

2. 3D - 3D model problem

    - mysolvers - solvers of the problem (SSNM)
    - definition_of_the_problem
    - data - plotting and results

1. the running m-file is compute.m, the main solver is SSNM.m
2. the running m-file is myTEST.m, the main solver is SSNM.m

Bibliography

[1] Benzi, M., Golub, G. H., Liesen, J.: *Numerical solution of saddle point problems*. Acta Numerica, (2005), pp. 1-137.

[2] Felcman, J.: *Numerická matematika*. KNM Press, Praha, 2005.

[3] Golub, G. H., Van Loan, C. F.: *Matrix Computations*. The Johns Hopkins University Press, Baltimore; Maryland, 1996.

[4] Haslinger, J., Kučera, R.: *Scalable T-FETI based algorithm for 3D contact problems with orthotropic friction*. Submitted to Lecture Notes in Applied and Computional Mechanics, **16** (2009).

[5] Haslinger, J., Zoubek, T., Kučera, R., Peichl, G.: *Projected Schur complement method for solving non-symmetric systems arising from a smooth fictitious domain approach*. Numer. Linear Algebra Appl., **14** (2007), pp. 713-739.

[6] Hintermüller, M., Ito, K., Kunisch, K.: *The primal-dual active set strategy as a semismooth Newton method*. SIAM J. Optim., **13** (2003), pp. 865-888.

[7] Hüeber, S.: *Discretization techniques and efficient algorithms for contact problems*. Institut für Angewandte Analysis und Numerische Simulation Universität Stuttgart, Thesis (2008).

[8] Chen, X., Nashed, Z., Qi, L.: *Smoothing methods and semismooth methods for nondifferentiable operator equations*. SIAM J. Numer. Anal., **38** (2000), pp. 1200-1216.

[9] Ito, K., Kunish, K.: *Semi-smooth Newton methods for variational inequalities of the first kind*. Mathematical Modelling and Numerical Analysis, **37** (2002), pp. 41-62.

[10] Jeyakumar, V., Luc, D. T.: *Approximate Jacobian matrices for nonsmoooth continuous maps and C^1-optimization*. SIAM J. Optim., **36** (1998), pp. 1815-1832.

[11] Kučera, R., Machalová, J., Ženčák, P.: *Newton-like algorithms for 3D contact problems*. Proceedings PANM14, Horní Maxov (2008), pp. 111 - 117.

[12] Kučera, R.: *Convergence rate of an optimization algorithm for minimizing quadratic functions with separable convex constraints*. SIAM J. Optim., **19** (2008), pp. 846-862.

[13] Ligurský, T., Haslinger, J., Kučera, R.: *Approximation and numerical realization of 3D contact problems with Coulomb friction and a solution-dependent coefficient of friction*. Submitted to International Journal for Numerical Methods in Engineering (2009).

[14] Míka, S.: *Matematická optimalizace*, Západočeská univerzita, Plzeň, 1997.

[15] Nocedal, J., Wright, S. J.: *Numerical Optimization*, Series in Operations Research, Springer, New York; Berlin; Heildeberg, 2006.

[16] Quarteroni, A., Sacco, R., Saleri, F.: *Numerical Mathematics*. Springer, New York; Berlin; Heildeberg, 2007.

[17] Sikorski, R.: *Diferenciální a integrální počet, Funkce více proměnných*. Academia, Praha, 1973.

[18] Van der Vorst (1992), H. A.: *BiCGSTAB: a fast and smoothly converging variant of BiCG for solution of nonsymmetric linear systems*. SIAM J. Sci. Statist. Comput., **13** (1992), pp. 631–644.

[19] Cílek, V.: *Mrtvá kočka*. Dokořán, 2002.