Operations Research Letters 42 (2014) 213–216

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

Performance analysis of linear and nonlinear techniques for automatic scaling of discretized control problems

Marco Sagliano *

Guidance, Navigation and Control Department, Deutsches Zentrum für Luft- und Raumfahrt, Robert Hooke Straße 7, Bremen, 28359, Germany

ARTICLE INFO

Article history:
Received 13 January 2014
Received in revised form
17 March 2014
Accepted 17 March 2014
Available online 21 March 2014

Keywords:
Automatic scaling
Optimal control problem
Nonlinear programming problem

ABSTRACT

This paper introduces three (one linear and two nonlinear) automatic scaling techniques for NLPs with states and constraints spread over several orders of magnitude, without requiring complex off-the-shelf external tools. All of these methods have been compared to standard techniques and applied to three problems using SNOPT and IPOPT. The results confirm that the proposed techniques significantly improve the NLP conditioning, yielding more reliable and in some cases, faster NLP solutions.

1. Introduction

Optimal control problems which are too complex to be solved analytically can often be solved numerically once the original problem is converted to a nonlinear programming problem (NLP). This conversion is carried out by using one of the many transcription methods, which transform the original continuous problem into an approximate discretized version that can be numerically solved with one of the well-known NLP solvers such as SNOPT [7] or IPOPT [12]. Unfortunately, having a good transcription method is not sufficient to ensure the quality of the solution, since poor scaling can make it difficult to compute the minimizer accurately or to even compute the minimizer at all. To overcome this problem, historically the first approach has always been the time-consuming process of manually scaling the problem [2,10], although several automatic scaling methods can also be found in the literature, c.f. the isoscaling (IS) and Jacobian rows normalization (JRN) methods described in [10,1,9]. The evolution of these automatic scaling techniques leads to the development of the projected Jacobian rows normalization (PJRN) which we propose here, and by extension, to the nonlinear scaling methodologies. For completeness, it is important to stress that convex optimization [4] or pre-optimization [8] could also be used to automatically scale the NLP, although this is out of the scope of this work, since we are interested in simple techniques which do not require additional off-the-shelf software. This paper is organized as follows. Section 2 characterizes the NLP problem to be scaled, and Section 3 briefly reports the standard scaling technique for the states, which is independent of the NLP scaling that we discuss here. Sections 4 and 5 present the proposed linear and nonlinear automatic scaling techniques. Finally, Section 6 reports the improvements in the condition numbers obtained with the presented techniques. Moreover, the CPU times required to scale and solve the NLP problems are reported, along with the minimal cost obtained by solving the scaled NLP for the three different problems analyzed here.

2. Characterization of the NLP problem

A detailed description of NLPs arising from the transcription of optimal control problems can be found in [3,5]. In the general case, we can formulate the NLP as having a cost function $J(\mathbf{X})$, a set of algebraic constraints $\mathbf{F}(\mathbf{X})$ representing the differential equations, and in some cases, a set of path constraints $\mathbf{G}(\mathbf{X})$. Hence the NLPs we consider are of the form (1).

$$\begin{aligned} &\min J = J(\mathbf{X}),\\ &\mathbf{F}(\mathbf{X}) = 0\\ &\mathbf{g}_L \leq \mathbf{G}(\mathbf{X}) \leq \mathbf{g}_U. \end{aligned} \tag{1}$$

A measure of the quality of a scaling method is the condition number of the Jacobian of the NLP (1), which in the general case is a rectangular matrix given by (2).

$$\mathbf{Jac} = \begin{Bmatrix} \nabla J \\ \nabla \mathbf{F} \\ \nabla \mathbf{G} \end{Bmatrix}. \tag{2}$$

* Tel.: +49 17659645804.
E-mail address: Marco.Sagliano@dlr.de.

http://dx.doi.org/10.1016/j.orl.2014.03.003
0167-6377/

We use the condition number of the Jacobian as a measure of a scaling method's quality because the Jacobian defines the search direction during the iterative process, and therefore a well-conditioned Jacobian is essential for solving (1) without excessive rounding errors.

In the following sections, we will show how the Jacobian of the differential equations $\mathbf{F}(\mathbf{X})$ and the path constraints $\mathbf{G}(\mathbf{X})$ can be treated with different techniques.

3. Scaling of NLP states

The states $\mathbf{X}$ of the NLP problem are scaled using the standard linear transformation given in [1], regardless of the NLP scaling method (IS, JRN, etc.) that we use. Specifically, the scaled state $\tilde{\mathbf{X}}$ is given by (3).

$$\tilde{\mathbf{X}} = \mathbf{K_x} \cdot \mathbf{X} + \mathbf{b_x}. \tag{3}$$

$\mathbf{K_x}$ is a diagonal matrix, and $\mathbf{b_x}$ is a vector having the same dimensions as $\mathbf{X}$. Since we always deal with bounded states and control, the diagonal elements of the matrices $\mathbf{K_x}$ and $\mathbf{b_x}$ are defined to be

$$\mathbf{K_{x_{ii}}} = \frac{1}{\mathbf{X_{U_i}} - \mathbf{X_{L_i}}}, \qquad \mathbf{b_{x_i}} = -\frac{\mathbf{X_{L_i}}}{\mathbf{X_{U_i}} - \mathbf{X_{L_i}}}. \tag{4}$$

Note that the transformation (4) yields scaled states $\tilde{\mathbf{X}}$ which always lie in the interval $[0, 1]$. In case of unbounded states, artificial upper and lower boundaries are usually introduced [1].

4. Linear techniques

Linear scaling techniques use a scaling of the form (5).

$$\tilde{\mathbf{F}} = \mathbf{K_f} \cdot \mathbf{F}, \qquad \tilde{\mathbf{G}} = \mathbf{K_g} \cdot \mathbf{G}. \tag{5}$$

$\mathbf{K_f}$ and $\mathbf{K_g}$ are diagonal matrices. The isoscaling (IS) method is one such technique whereby the constraints $\mathbf{F}$ are scaled exactly like the states, that is,

$$\mathbf{K_f} = \mathbf{K_x},$$

where $\mathbf{K_x}$ is given by (4), see [1,9]. Note that isoscaling does not help in scaling the constraints $\mathbf{G}$. A possible refinement of this approach has been suggested by Rao [10], who uses randomly sampled points around the vector $\mathbf{X}$, and computes the mean of the norms of the Jacobian rows of $\mathbf{F}$ and $\mathbf{G}$ instead of the norm of the Jacobian rows. Unfortunately, this technique significantly increases the CPU time needed to compute the scaling coefficients, since the Jacobian matrix must be evaluated many more times. Next, we introduce a simple linear scaling technique which does not require additional Jacobian evaluations, and hence is much less computationally expensive.

4.1. Projected Jacobian rows normalization

Isoscaling bases the scaling of the constraints solely on the scaling of the states. In other words, it does not take into account the relationship between the states and the constraints, which is represented in linearized form by the Jacobian. Conversely, the Jacobian rows normalization (JRN) only considers this relationship, without involving the states' normalization in the process. Specifically, in the JRN technique, the diagonal elements of $\mathbf{K_f}$ and $\mathbf{K_g}$ are given by (6).

$$\mathbf{K_{f_{ii}}} = \frac{1}{|\nabla \mathbf{F}|_i}, \qquad \mathbf{K_{g_{ii}}} = \frac{1}{|\nabla \mathbf{G}|_i}. \tag{6}$$

The projected Jacobian rows normalization (PJRN) technique which we propose considers both of these factors. Specifically, in the PJRN, the diagonal elements of $\mathbf{K_f}$ and $\mathbf{K_g}$ are given by (7).

$$\mathbf{K_{f_{ii}}} = \frac{1}{\left|\nabla \mathbf{F} \cdot \mathbf{K_x^{-1}}\right|_i}, \qquad \mathbf{K_{g_{ii}}} = \frac{1}{\left|\nabla \mathbf{G} \cdot \mathbf{K_x^{-1}}\right|_i}. \tag{7}$$

As we will show in Section 6, this scaling generally leads to a better-conditioned Jacobian matrix, and to a more uniformly distributed singular values.

The Jacobian of the PJRN-scaled NLP is given by (8).

$$\tilde{\mathbf{Jac}} = \begin{Bmatrix} \tilde{\nabla}\tilde{J} \\ \tilde{\nabla}\tilde{\mathbf{F}} \\ \tilde{\nabla}\tilde{\mathbf{G}} \end{Bmatrix} = \begin{Bmatrix} K_J \cdot \nabla J \cdot \mathbf{K_x^{-1}} \\ \mathbf{K_F} \cdot \nabla \mathbf{F} \cdot \mathbf{K_x^{-1}} \\ \mathbf{K_G} \cdot \nabla \mathbf{G} \cdot \mathbf{K_x^{-1}} \end{Bmatrix}. \tag{8}$$

K_J is a parameter which normalizes the cost function J, $\mathbf{K_x}$ is given by (4), and $\mathbf{K_f}$ and $\mathbf{K_g}$ are given by (6).

5. Nonlinear techniques

Nonlinear scaling techniques generalize the second relationship reported in (5). Note that it is also possible to scale $\mathbf{F}$ with a nonlinear scaling technique, however we will take advantage of the boundedness of $\mathbf{G}$, and hence we only consider the nonlinear scaling of $\mathbf{G}$ in this paper. Specifically, we propose using the logarithm and the inverse-power.

5.1. Logarithmic scaling

The first nonlinear scaling technique we propose is the natural logarithm, in which case the scaled constraint function is given by (9).

$$\tilde{\mathbf{G}} = \log\left(\mathbf{G} + \mathbf{C}\right). \tag{9}$$

The constant vector $\mathbf{C}$ ensures that the argument of the logarithm is always greater than or equal to 1. Specifically, since the constraint function is bounded from below by $\mathbf{g_L}$ (1), we choose $\mathbf{C}$ to be given by (10).

$$\mathbf{C} = -\mathbf{g_L} + \mathbf{1}. \tag{10}$$

When the constraints are intrinsically positive (e.g. when we consider the dynamic pressure or the heat rate, as in the case of the Space Shuttle Entry Problem), this simply reduces to

$$\mathbf{C} = \mathbf{1} \tag{11}$$

where $\mathbf{1}$ is a vector of ones with the same dimensions as $\mathbf{G}$.

The Jacobian of a logarithmically scaled NLP is of the form (8), where $\nabla\tilde{\mathbf{G}}$ is given by (12).

$$\tilde{\nabla}\tilde{\mathbf{G}} = \frac{\mathbf{1}}{\mathbf{G} + \mathbf{C}} \cdot \nabla \mathbf{G} \cdot \mathbf{K_x^{-1}}. \tag{12}$$

Note that since the logarithmic scaling only affects $\mathbf{G}$, $\mathbf{K_f}$ must be chosen using a linear scaling technique such as IS, JRN, or PJRN.

5.2. Inverse-power scaling

The second nonlinear scaling technique we propose is the inverse-power scaling technique, in which case the scaled constraint function is given by (13).

$$\tilde{\mathbf{G}} = \left(\mathbf{G} + \mathbf{C}\right)^{\frac{1}{\mathbf{n}}} \tag{13}$$

$\mathbf{n}$ is a positive integer, and $\mathbf{C}$ is chosen so that $\mathbf{G}+\mathbf{C}$ is always greater than or equal to 1. Specifically, in the present paper, we always