Science & Technology Facilities Council

Technical Report

RAL-TR-2011-007

Updating the regularization parameter in the adaptive cubic regularization algorithm

NIM Gould, M Porcelli, PL Toint

February 2011

Enquiries about copyright, reproduction and requests for additional copies of this report should be addressed to:

RAL Library
STFC Rutherford Appleton Laboratory
R61
Harwell Oxford
Didcot
OX11 0QX

Tel: +44(0)1235 445384
Fax: +44(0)1235 446403
email: libraryral@stfc.ac.uk

Science and Technology Facilities Council reports are available online at: http://epubs.stfc.ac.uk

ISSN 1358- 6254

Updating the regularization parameter in the adaptive cubic regularization algorithm

Nicholas I. M. Gould[1,2], Margherita Porcelli[3] and Philippe L. Toint[3]

ABSTRACT
The adaptive cubic regularization method (Cartis, Gould & Toint, Math. Programming, DOI: 10.1007/s10107-009-0286-5 & 10.1007/s10107-009-0337-y) has been recently proposed for solving unconstrained minimization problems. At each iteration of this method, the objective function is replaced by a cubic approximation which comprises an adaptive regularization parameter whose role is related to the local Lipschitz constant of the objective's Hessian. We present new updating strategies for this parameter based on interpolation techniques, which improve the overall numerical performance of the algorithm. Numerical experiments on large nonlinear least-squares problems are provided.

Keywords: unconstrained optimization, cubic regularization, numerical performance.
AMS classification: 49J52, 49M37, 65F22, 65K05, 90C26, 90C30, 90C55.

---

[1] Computational Science and Engineering Department, Rutherford Appleton Laboratory, Chilton, Oxfordshire, OX11 0QX, England, EU. Email: nick.gould@stfc.ac.uk .
Current reports available from "http://www.numerical.rl.ac.uk/reports/reports.shtml".

[2] This work was supported by the EPSRC grant EP/E053351/1.

[3] Department of Mathematics, Facultés Universitaires ND de la Paix, 61, rue de Bruxelles, B-5000 Namur, Belgium, EU.
Email : margherita.porcelli@fundp.ac.be, philippe.toint@fundp.ac.be .
Current reports available from "http://www.fundp.ac.be/∼phtoint/pht/publications.html".

Computational Science and Engineering Department
Atlas Centre
Rutherford Appleton Laboratory
Oxfordshire OX11 0QX

February 25, 2011

1 Introduction

We consider the unconstrained minimization problem

$$\min_{x \in \mathbb{R}^n} f(x), \tag{1.1}$$

where f is a twice continuously differentiable function of the variables $x \in \mathbb{R}^n$. A simplistic method for solving this problem is to compute an improving step s_k by minimizing a quadratic Taylor-series model of the objective function around the current iterate x_k. Unfortunately, it is well-known that an iteration based on this simple idea may not always be well-defined (when the Taylor model is nonconvex), nor converge globally. These drawbacks may be overcome by restricting the model minimization to a trust region containing x_k [8]. Clearly, trust-region strategies may be considered as regularization techniques because they control the difference between two consecutive iterates by explicitly imposing a restriction on the stepsize.

The main motivation for this paper is a series of recent papers where alternative regularization strategies are introduced [2, 3, 7, 17, 20, 24]. These procedures are based on the minimization of quadratic or cubic models for the objective function in a neighbourhood implicitly defined by a regularization term that penalizes the step length. In particular, the adaptive cubic regularization (ARC) algorithm is proposed in [3] for solving problem (1.1). At each iteration, the objective function is locally replaced by a cubic approximation, in which third- and higher-order Taylor-series terms are replaced by a cubic regularization term, and an adaptive estimation of the local Lipschitz constant of the objective function's Hessian is employed. The method has been shown to have excellent global and local convergence properties and numerical experiments indicate that the new procedure may be competitive with the trust region approach when solving small-scale problems [3]. Additionally, and of theoretical interest, ARC possesses a better worst-case evaluation-complexity bound than its trust-region competitor [5].

The purpose of this paper is twofold. Firstly, we propose alternative updating rules for the regularization parameter of the ARC algorithm which are based on interpolation techniques. In particular, in the trust-region case, the restriction on the stepsize is explicitly imposed by the trust-region constraint. By contrast, in the cubic regularization case the control on the stepsize is nonlinear and is defined implicitly. This suggests a need to design an efficient updating rule for the regularization parameter that is able to control the stepsize in a flexible way.

Secondly, we shall apply these ideas and report on extensive numerical experiments on the solution of large nonlinear least-squares problems, that is problems of the form

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{2}\|h(x)\|_2^2, \tag{1.2}$$

where $h : \mathbb{R}^n \to \mathbb{R}^m$ is a given continuously differentiable mapping. By limiting our discussion to this problem, we may specialize the models employed in both the ARC and trust-region algorithms to those that are suited to solving nonlinear least-squares problems, specifically using regularized Gauss-Newton-based models, and consequently to take advantage of the ideas and implementations details proposed in [7] for the solution of large regularized linear least-squares problems. Since we are primarily interested in large problems for which matrix factorization often has prohibitive computational cost, we shall focus on iterative algorithms for the subproblems, particularly on those implemented as part of version 2.4 of the GALAHAD optimization library [16]. Such procedures are based on the minimization of the local model of the objective function over a sequence of (nested) subspaces associated with the Lanczos procedure. As a result, they are especially suited to the large-scale setting and allow us to test the methods on large problems from the CUTEr test collection [15]. In particular, the new updating rules for the regularization parameter of the ARC algorithm are experimentally validated and a comparison with the trust-region algorithm is performed on problem (1.2).

The paper is organized as follows. In Section 2 we review the standard trust-region algorithm and the ARC algorithm for the solution of problem (1.1). New updating rules for the regularization parameter in the ARC algorithm are introduced in Section 3. Section 4 is dedicated to numerical experiments and, finally, in Section 5 we draw some conclusions.

Throughout the paper we use the following notation. The Euclidean (ℓ_2) norm is denoted by $\|\cdot\|$, and I represents the identity matrix. Given a sequence of vectors $\{x_k\}$, for any generic function h we let $h_k = h(x_k)$. Let $g(x) = \nabla f(x)$ where f is the objective function in (1.1) and let $J(x)$ denote the Jacobian matrix of the residual function $h(x)$ in (1.2). Finally, $\epsilon_m \approx 10^{-16}$ denotes the relative machine (double) precision.

2 The algorithms

In this section, we describe the kth iteration of two globally convergent algorithms for the solution of problem (1.1): the standard trust-region algorithm (e.g. [8]) and the ARC algorithm ([3]).

In the trust-region framework, a quadratic model of $f(x)$ around x_k is constructed by defining the model of the objective function to be

$$q_k(s) = f_k + g_k^T s + \frac{1}{2} s^T H_k s, \tag{2.1}$$

where H_k is a symmetric approximation to the local Hessian $\nabla_{xx} f_k$. Then, a trial step s_k is computed by solving (possibly only approximately) the subproblem

$$\min_{s \in \mathbb{R}^n} \{q_k(s) : \quad \|s\| \leq \Delta_k\}, \tag{2.2}$$

where $\Delta_k > 0$ is the so-called trust-region radius.

By contrast, assuming that the objective's Hessian $\nabla_{xx} f$ is globally Lipschitz continuous on $\mathbb{R}^n$ with Lipschitz constant L, the cubic model used in the ARC algorithm is based on the bound

$$\begin{aligned} f(x_k+s) &= f_k + s^T g_k + \frac{1}{2} s^T \nabla_{xx} f_k s + \int_0^1 (1-\tau) s^T [\nabla_{xx} f(x_k+\tau s) - \nabla_{xx} f_k] s \, d\tau \\ &\le f_k + s^T g_k + \frac{1}{2} s^T \nabla_{xx} f_k s + \frac{1}{6} L \|s\|^3 \stackrel{\rm def}{=} l_k(s), \end{aligned} \tag{2.3}$$

which holds for for all $s \in \mathbb{R}^n$. Thus, so long as $l_k(s_k) < l_k(0) = f_k$, the new iterate $x_{k+1} = x_k + s_k$ improves $f(x)$. In [3], a dynamic positive parameter σ_k replaces the Lipschitz constant $L/2$ and a symmetric approximation H_k to the local Hessian $\nabla_{xx} f_k$ is allowed. At each iteration, the cubic model

$$c_k(s) = f_k + s^T g_k + \frac{1}{2} s^T H_k s + \frac{1}{3} \sigma_k \|s\|^3, \tag{2.4}$$

is employed as an approximation to the objective f and the subproblem

$$\min_{s \in \mathbb{R}^n} c_k(s) \tag{2.5}$$

is solved. The parameter σ_k plays a crucial role in the description of the ARC algorithm as it measures the discrepancy between the objective function and its second order Taylor expansion and of the difference between the exact and the approximate Hessian [3].

It is important to note that the restriction on stepsize is explicitly imposed by the trust-region constraint in the trust-region case, while stepsize control is defined implicitly, indeed nonlinearly, in the cubic case. In fact, a step s_k derived by reducing (2.5) is always bounded [3, Lem.2.2] by

$$\|s_k\| \le 3 \max \left[\frac{\|H_k\|}{\sigma_k}, \sqrt{\frac{\|g_k\|}{\sigma_k}} \right].$$

Such a bound suggests that the regularization parameter σ_k for the ARC algorithm may loosely be interpreted as the reciprocal of the trust-region radius Δ_k. This observation in turn suggests choosing updating rule for the parameter σ_k by analogy with the trust-region case. In a standard trust-region scheme, the trust-region radius may be enlarged if there is a sufficient decrease in $f(x)$, computed by some measure of the relative objective changes, and it is reduced otherwise. In the regularization case, the parameter σ_k is decreased if there is a sufficient agreement between the objective function and the model, but increased or left unchanged otherwise.

In both algorithms, the agreement between the model and the objective function is given by the standard ratio of the achieved to the predicted reduction, and the size of this ratio is used to decide whether or not to accept the trial step and to change the regularization parameter. This ratio takes the form

$$\rho_q(s_k) = \frac{f_k - f(x_k+s_k)}{q_k(0) - q_k(s_k)}, \tag{2.6}$$

in the trust-region case, and

$$\rho_c(s_k) = \frac{f_k - f(x_k + s_k)}{c_k(0) - c_k(s_k)}, \tag{2.7}$$

in the the cubic regularization case, where the models q_k and c_k are defined in (2.1) and (2.4) respectively. Without ambiguity, let $\rho(s)$ represent both $\rho_c(s)$ and $\rho_q(s)$, and let η_1, η_2 be constants such that $0 < \eta_1 < \eta_2 < 1$. We say that the iteration k is *very successful* if $\rho(s_k) \geq \eta_2$, *successful* if $\rho(s_k) \in [\eta_1, \eta_2)$, *unsuccessful* otherwise. When it is useful to distinguish the case $\rho(s_k) < 0$ within the unsuccessful case, we refer to a *very unsuccessful* iteration.

The general framework of the methods described so far is presented in Algorithm 2.1. The string `METHOD` denotes the name of the method, i.e. it is either `‘TRUST-REGION’` or `‘ARC’`. Sections 2.1 and 3 give further insight into Steps 1 and 4.

2.1 Computing a trial step

Step 1 of Algorithm 2.1 leaves substantial implementation freedom, which may be used according to context. The focus of this paper is on the case where matrix factorizations of the Hessian matrix are not feasible, implying that iterative methods for computing a trial step are needed. We consider the class of subspace minimization methods, i.e. methods that find an approximate solution by solving a sequence of minimization problems with the additional constraint that s is contained in a subspace. This class may be divided into two subclasses depending on the construction of the sequence of subspaces. The first consists of expanding subspaces methods. The Conjugate Gradient (CG) method belongs to this subclass as it may be viewed as a subspace minimization method for finding an unconstrained minimizer of a strictly convex quadratic function, where, at each successive iteration, the quadratic function is minimized by restricting the variable to a sequence of nested Krylov subspaces. In [3, 14], methods based on this approach have been proposed for solving the regularized cubic problem (2.5) and the trust-region problem (2.2), respectively. The second subclass comprises low-dimensional subspace methods, i.e. methods that always generate subspaces of low-dimension. Such methods have been proposed in literature only for solving problem (2.2) and differ in the choice of the subspaces [11, 12, 18, 19]. In order to apply the same subspace approach to both the trust-region and the cubic case, we consider the former subclass of methods to perform Step 1.

Consider the nonlinear least-squares problem (1.2). At the current iterate x_k, the exact Hessian of the objective function f has the form

$$\nabla_{xx} f_k = J_k^T J_k + S_k,$$

where S_k contains the second-order information on the residual. If S_k is small, it is reasonable to consider the first order approximation $H_k = J_k^T J_k$. This is the case, for instance, in

Algorithm 2.1: Generic trust-region/cubic regularization method

An initial point x_0 as well as constants $0 < \eta_1 < \eta_2 < 1$ and $\gamma > 1$ are given.
If `METHOD = 'TRUST-REGION'`, set the initial radius $\Delta_0 > 0$ and the constants τ_1, τ_2 such that $0 \leq \tau_1 \leq \tau_2 \leq 1$. Else set the intial regularization parameter $\sigma_0 > 0$ and the constants ν_1, ν_2 such that $1 < \nu_1 \leq \nu_2$.

For $k = 0, 1, \ldots,$ until convergence,

Step 1: Trial step computation. If `METHOD = 'TRUST-REGION'`, compute s_k as an (approximate) solution of problem (2.2). Else, compute s_k as an (approximate) solution of problem (2.5).

Step 2: Step acceptance. If `METHOD = 'TRUST-REGION'`, compute $\rho(s_k) = \rho_q(s_k)$ as in (2.6). Else, compute $\rho(s_k) = \rho_c(s_k)$ as in (2.7).

If $\rho(s_k) \geq \eta_1$, let $x_{k+1} = x_k + s_k$; otherwise let $x_{k+1} = x_k$.

Step 4: Regularization parameter update. If `METHOD = 'TRUST-REGION'` set

$$\Delta_{k+1} \in \begin{cases} [\Delta_k, \infty) & \text{if } \rho(s_k) \geq \eta_2 & \text{[very successful iteration]} \\ [\tau_2\Delta_k, \Delta_k] & \text{if } \rho(s_k) \in [\eta_1, \eta_2) & \text{[successful iteration]} \\ [\tau_1\Delta_k,\ \tau_2\Delta_k] & \text{otherwise} & \text{[unsuccessful iteration]} \end{cases}. \tag{2.8}$$

Else set

$$\sigma_{k+1} \in \begin{cases} (0,\ \sigma_k] & \text{if } \rho(s_k) \geq \eta_2 & \text{[very successful iteration]} \\ [\sigma_k,\ \nu_1\sigma_k] & \text{if } \rho(s_k) \in [\eta_1, \eta_2) & \text{[successful iteration]} \\ [\nu_1\sigma_k,\ \nu_2\sigma_k] & \text{otherwise} & \text{[unsuccessful iteration]} \end{cases}. \tag{2.9}$$

a neighborhood of a zero residual solution of problem (1.2), [10]. Using the approximation $H_k = J_k^T J_k$, the quadratic model in (2.1) takes the form

$$q_k(s) = \frac{1}{2}\|J_k s + h_k\|^2, \tag{2.10}$$

which is the Gauss-Newton model for f, and the cubic model in (2.4) becomes

$$c_k(s) = \frac{1}{2}\|J_k s + h_k\|^2 + \frac{\sigma_k}{3}\|s\|^3, \tag{2.11}$$

yielding a Gauss-Newton model regularized by a cubic term.

Procedures have been proposed in [7] to solve the subproblems (2.2) and (2.5) in the special case where the models are given in (2.10) and (2.11) respectively. The core component of these procedures is the Golub and Kahan bi-diagonalization process [13] that generates orthonormal basis of a sequence of expanding subspaces $\{\mathcal{V}_j\}_{j\geq 1}$. Let $V_j \in \mathbb{R}^{n\times j}$ be the orthonormal matrix whose columns span $\mathcal{V}_j$. The solutions of problems (2.2) and (2.5) are found by computing the sequence of minimizers y_j of the reduced problems

$$\min_{y\in\mathbb{R}^j} \{q_k(V_j y) : \quad \|y\| \leq \Delta_k\}, \tag{2.12}$$

and

$$\min_{y\in\mathbb{R}^j} c_k(V_j y), \tag{2.13}$$

respectively, increasing the dimension j of the subspaces until $s_j = V_j y_j$ is sufficiently accurate. At that point, the step s_k in the full space is taken as the last computed s_j [7].

It is interesting to note, that if the LSQR algorithm [21] is used to solve the unconstrained problem $\min_s q_k(s)$, a basis of the Krylov subspaces

$$\mathcal{K}_j = \left\{(J_k^T J_k)^i J_k^T h_k\right\}_{i=0}^{j-1},$$

is given by the columns of V_j. Due to the equivalence between the LSQR and CG methods, the sequence s_j generated by LSQR has the favorable property to be monotonically increasing in norm [23]. Thus, either LSQR finds a solution in the interior of the trust-region, or finds an iterate s_j s.t. $\|s_{j-1}\| \leq \Delta_k < \|s_j\|$ and in this case we may conclude that the solution of the problem (2.2) lies on the boundary of the trust-region. When this happens two alternative strategies can be followed: either the so-called Steihaug-Toint point [8, §7.5.1] is computed or a solution on the boundary is computed to any prescribed accuracy. The Steihaug-Toint strategy interpolates the last interior iterate s_{j-1} with the newly discovered exterior one s_j to find the boundary point between them. The resulting step has the favorable property that the optimal decrease of q_k at the exact solution of the trust-region problem (2.2), is no more than twice that achieved at the Steihaug-Toint point (see [25] or [8, Thm.7.5.9]). On the negative side however, it makes no attempt to find a constrained solution with prescribed accuracy. A more refined strategy solves a sequence of constrained reduced problems (2.12) increasing j until s_j is sufficiently accurate [7]. Note that this strategy specializes to problem (2.12) the GLTR method [14] for the general trust-region problem (2.2) in which the CG method is used as long as the iterates are in the interior of the trust-region and the expanding subspaces are defined by the Lanczos vectors.

3 Updating rules for the regularization parameters

Because of its central role, the definition of a procedure to update the regularization parameters at Step 4 of Algorithm 2.1 may have a crucial influence on its overall performance. In this section, we first review two established updating strategies for the trust-region radius Δ_k and then propose new strategies for the parameter σ_k for the ARC algorithm.

Clearly, the rule (2.8) in Algorithm 2.1 leaves considerable flexibility. A simple and reasonable choice is to select

$$\Delta_{k+1} = \begin{cases} \max\{\gamma_2\|s_k\|, \Delta_k\} & \text{if } \rho_q(s_k) \geq \eta_2 \quad \text{[very successful iteration]}, \\ \Delta_k & \text{if } \rho_q(s_k) \in [\eta_1, \eta_2) \quad \text{[successful iteration], and} \\ \gamma_1\|s_k\| & \text{otherwise} \quad \text{[unsuccessful iteration]}, \end{cases} \tag{3.1}$$

where γ_1 and γ_2 are constants such that $0 < \gamma_1 < 1 \leq \gamma_2$, but further refinements are possible using interpolation techniques in the unsuccessful case. If $\rho_q(s_k)$ is negative, the agreement between the model and the objective function is extremely poor and some drastic action might be warranted. In this case, we presume for simplicity that s_{k+1} will be aligned with s_k and we compute a trust-region radius small enough to ensure that the new step gives at least a successful iteration [8, Chapter 17]. To compute such a radius, we consider a step of the form αs_k with $\alpha > 0$ and we set $\Delta_{k+1} = \alpha_\eta^{bad}\Delta_k$ where α_η^{bad} solves $\rho_q(\alpha s_k) = \eta$, which is equivalent to the scalar nonlinear equation

$$f_k - f(x_k + \alpha s_k) = \eta(q_k(0) - q_k(\alpha s_k)), \tag{3.2}$$

with $\eta \in [\eta_1, 1)$ and η_1 as given in Algorithm 2.1. To avoid the expense of computing the extra function value $f(x_k + \alpha s_k)$ and to simplify the solution of (3.2), the scalar function $\hat{f}(\alpha) = f(x_k + \alpha s_k)$, $\alpha > 0$ is replaced by a quadratic interpolating polynomial for $\hat{f}$. The polynomial $t_f(\alpha)$ such that t_f and t_f' agree with $\hat{f}$ and $\hat{f}'$ at 0, and $t_f(1) = \hat{f}(1) = f(x_k+s_k)$, is given by

$$t_f(\alpha) = f_k + g_k^T s_k \alpha + \left(f(x_k + s_k) - f_k - g_k^T s_k\right)\alpha^2.$$

Substituting this value for $f(x_k + \alpha s_k)$ into (3.2) and solving for α, yields the value of α_η^{bad} given by

$$\alpha_\eta^{bad} = \frac{(1-\eta)g_k^T s_k}{(1-\eta)(f_k + g_k^T s_k) + \eta\, q_k(s_k) - f(x_k + s_k)}. \tag{3.3}$$

We may therefore modify (3.1) to use this information and obtain the more sophisticated rule

$$\Delta_{k+1} = \begin{cases} \max\{\gamma_2\|s_k\|, \Delta_k\} & \text{if } \rho_q(s_k) \geq \eta_2 \quad \text{[very successful iteration]}, \\ \Delta_k & \text{if } \rho_q(s_k) \in [\eta_1, \eta_2) \quad \text{[successful iteration]}, \\ \gamma_1\|s_k\| & \text{if } \rho_q(s_k) \in [0, \eta_1) \quad \text{[unsuccessful iteration], and} \\ \min\{\gamma_1\|s_k\|, \max\{\gamma_3, \alpha_\eta^{bad}\}\Delta_k\} & \text{otherwise} \quad \text{[very unsuccessful iteration]}, \end{cases} \tag{3.4}$$

where α_η^{bad} is given by (3.3) and the constants $\gamma_1, \gamma_2, \gamma_3$ are such that $0 < \gamma_3 < \gamma_1 < 1 \leq \gamma_2$ [8].

Let us now consider the ARC framework with this in mind. The updating rule proposed in [3] aims to try to reduce the model rapidly to match the Newton model once convergence sets in, while maintaining some regularization before the asymptotic behaviour. The rule used in the reported experiments was

$$\sigma_{k+1} = \begin{cases} \max\{\min\{\sigma_k, \|g_k\|\}, \epsilon_m\} & \text{if } \rho_c(s_k) \geq \eta_2 \quad \text{[very successful iteration]}, \\ \sigma_k & \text{if } \rho_c(s_k) \in [\eta_1, \eta_2) \quad \text{[successful iteration], and} \\ \gamma\sigma_k & \text{otherwise} \quad \text{[unsuccessful iteration]}, \end{cases} \tag{3.5}$$

with $\gamma \geq 1$. Clearly, the relationship between the step length and the regularization parameter in (3.5) is not as simple as in the updating rules (3.1) for the trust-region case and the control of the first by the second is performed implicitly.

To relate the step size and the parameter σ_k in a more direct way, we now present an alternative strategy for updating σ_k in the spirit of the interpolation procedures used with the trust-region scheme. Specifically, we try to ensure, in the very unsuccessful case, that the next iterate gives at least a successful iteration. In the very successful case we may also exploit the overestimation property (2.3) measuring at each iteration the gap between the current objective function value $f(x_k + s_k)$ and the current model value $c_k(s_k)$ and reduce σ_k in order to decrease this gap (cf. [17, 24]). In particular, given the current x_k, σ_k and s_k, we presume, as above, that s_{k+1} is of the form $\alpha s_k, \alpha > 0$ and compute the value σ_{k+1} to ensure suitable conditions on αs_k.

As in the trust-region case, we avoid the need to compute the value of $f(x_k + \alpha s_k)$ by using instead a suitable interpolating approximation. The interpolating cubic function $p_f(\alpha)$, $\alpha \geq 0$ we use here is built by requiring that $p_f(0) = f_k$, $p_f'(0) = g_k^T s_k$, $p''(0) = s_k^T H_k s_k$ and $p_f(1) = f(x_k + s_k)$, and hence takes the form

$$p_f(\alpha) = f_k + g_k^T s_k \alpha + \frac{1}{2} s_k^T H_k s_k \alpha^2 + p_{f3}\alpha^3, \tag{3.6}$$

where

$$p_{f3} = f(x_k + s_k) - q_k(s_k). \tag{3.7}$$

The quadratic model (2.1) along the direction s_k may be written as

$$q(\alpha) = f_k + g_k^T s_k \alpha + \frac{1}{2} s_k^T H_k s_k \alpha^2, \tag{3.8}$$

while its regularized cubic counterpart (2.4) is

$$c(\alpha, \sigma) = q(\alpha) + \frac{\sigma \|s_k\|^3}{3} \alpha^3. \tag{3.9}$$

We now define the current overestimation gap χ_k^f to be

$$\chi_k^f = c_k(s_k) - f(x_k + s_k). \tag{3.10}$$

Note that the model c_k at s_k overestimates $f(x_k + s_k)$, i.e. $\chi_k^f \geq 0$, if and only if $\rho_c(s_k) \geq 1$.

Consider the very successful ($\chi_k^f > 0$) case first, in which case the regularization parameter should be decreased. If the current gap χ_k^f is large enough, we aim at reducing it by a factor $\beta \in (0, 1)$. Assume first that $f(x_k + s_k) \geq q_k(s_k)$. Remembering that the next step should minimize the cubic model (in particular along s_k), we thus search for α and σ such that

$$c(\alpha, \sigma) - p_f(\alpha) = \beta\, \chi_k^f \quad \text{and} \tag{3.11}$$

$$\frac{d}{d\alpha} c(\alpha, \sigma) = 0, \tag{3.12}$$

$c(\alpha,\sigma)$ and $p_f(\alpha)$ given in (3.9) and (3.6). It follows from (3.11) that

$$\sigma = 3\frac{\beta\chi_k^f + p_{f3}\alpha^3}{\alpha^3\|s_k\|^3} \equiv \sigma_k + 3\frac{\chi_k^f}{\|s_k\|^3}\left(\frac{\beta-\alpha^3}{\alpha^3}\right), \tag{3.13}$$

and substituting (3.13) into (3.12), we find that the required α satisfies the cubic scalar equation

$$3\beta\chi_k^f + g_k^T s_k\alpha + s_k^T H_k s_k\alpha^2 + 3p_{f3}\alpha^3 = 0. \tag{3.14}$$

Thus, we determine the root α of (3.14) which exceeds $\sqrt[3]{\beta}$ by the least (if there is such a root) and recover $\sigma^*_{k,\beta}$ from (3.13). If there is no such α, or if α is too large, we simply reduce σ_k by a factor $\delta_1 \in (0,1)$.

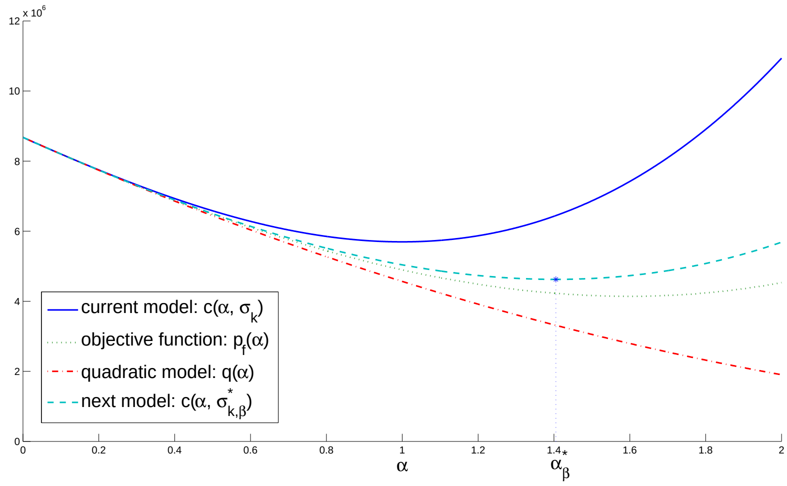

Figure 3.1: Very successful iteration and $f(x_k+s_k) \geq q_k(s_k)$.

In Figures 3.1–3.3 the current cubic model $c(\alpha,\sigma_k)$, the approximated objective function $p_f(\alpha)$, the quadratic model $q(\alpha)$ and the next cubic model $c(\alpha,\sigma^*_{k,\beta})$ are plotted. Figure 3.1 represents an example where the k-th iterate is very successful and $f(x_k+s_k) \geq q_k(s_k)$. In this example, $\beta = 0.5$ and equation (3.14) has two positive roots. The largest one (α^*_β in the figure) is larger than $\sqrt[3]{\beta} \approx 0.7937$ and gives $\sigma^*_{k,\beta}$ such that $\sigma^*_{k,\beta} < \sigma_k$.

Consider now the case where $f(x_k+s_k) < q_k(s_k)$. If we attempt to solve the system (3.11)–(3.12), i.e. try to reduce the quantity $c_k(s_k) - f(x_k+s_k)$ by a factor β, we might reduce this gap too much, leading to undesirable value of the new σ. Figure 3.2-(a) illustrates the typical situation: in this example ($\beta = 0.5$), equation (3.14) only has one positive solution ($\alpha^{\chi_k^f} \approx 0.745$ in the figure), but it is smaller than $\sqrt[3]{\beta}$ so that the corresponding $\sigma^*_{k,\beta}$ computed by (3.13) is larger than the current σ_k. To avoid this undesirable situation, we instead attempt to reduce the following gap

$$\chi_k^q = c_k(s_k) - q_k(s_k), \tag{3.15}$$

and search for α and σ such that

$$c(\alpha,\sigma) - q(\alpha) = \beta\,\chi_k^q \quad \text{and} \tag{3.16}$$

$$\frac{d}{d\alpha}c(\alpha,\sigma) = 0, \tag{3.17}$$

with $c(\alpha,\sigma)$ and $q(\alpha)$ given in (3.9). Computing σ from (3.16), we then find that

$$\sigma = 3\frac{\beta\chi_k^q}{\alpha^3\|s_k\|^3} \equiv \frac{\beta}{\alpha^3}\sigma_k, \tag{3.18}$$

and substituting (3.18) in (3.17) yields that α solves the quadratic scalar equation

$$3\beta\chi_k^q + g_k^T s_k\alpha + s_k^T H_k s_k\alpha^2 = 0. \tag{3.19}$$

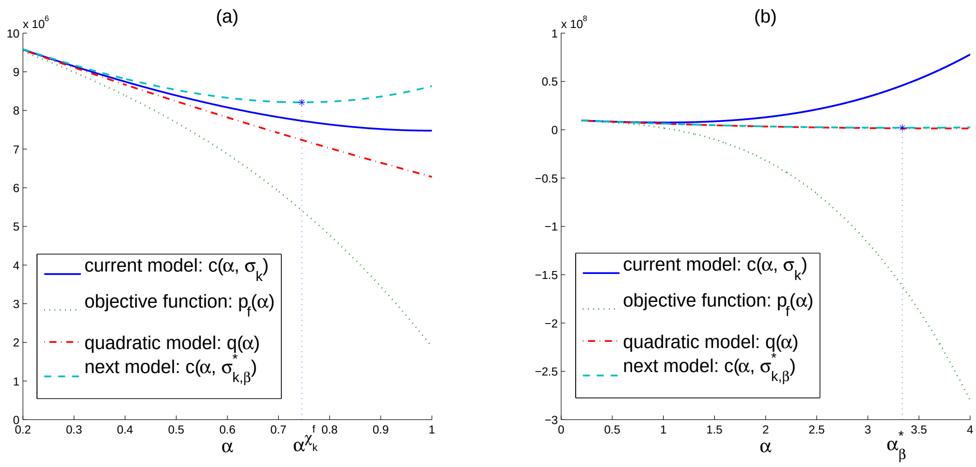

Figure 3.2: Very successful iteration and $f(x_k + s_k) < q_k(s_k)$.

As in the previous case, we compute the root of (3.19) which exceeds $\sqrt[3]{\beta}$ by the least (if such a root exists) and compute the corresponding value $\sigma_{k,\beta}^*$ using (3.18). Once again, if there is no such α, or if α is too large, we simply reduce σ_k by a factor $\delta_1 \in (0,1)$. Figure 3.2-(b) illustrates the same example as in Figure 3.2-(a) but now solving the system (3.16)–(3.17): equation (3.19) has 2 positive roots and one (α_β^* in the figure) is larger than $\sqrt[3]{\beta}$, so that the corresponding $\sigma_{k,\beta}^*$ is smaller than σ_k.

Let us now turn to the very unsuccessful case, $\rho_c(s_k) < 0$, where we wish to increase the regularization parameter. We proceed as in the trust-region framework simply requiring that αs_k produces at least a successful iterate. We thus search for α and σ such that

$$f_k - p_f(\alpha) = \eta(f_k - c(\alpha,\sigma)), \quad \text{and} \tag{3.20}$$

$$\frac{d}{d\alpha}c(\alpha,\sigma) = 0, \tag{3.21}$$

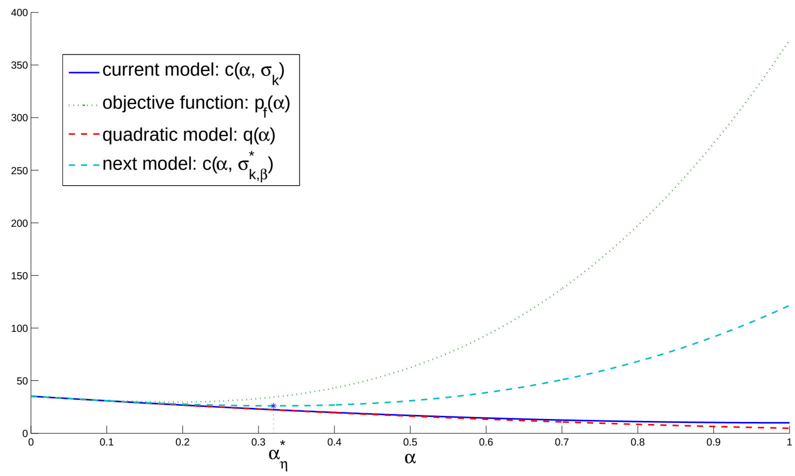

Figure 3.3: Very unsuccessful iteration.

for some $\eta \in [\eta_1, 1)$. Computing σ from (3.21) we obtain that

$$\sigma = \frac{-g_k^T s_k - s_k^T H_k s_k \alpha}{\alpha^2 \|s_k\|^3}, \tag{3.22}$$

and substituting this expression in (3.20), we find that α must be a root of the quadratic scalar equation

$$2(3 - 2\eta) g_k^T s_k + (3 - \eta) s_k^T H_k s_k \alpha + 6 p_{f3} \alpha^2 = 0, \tag{3.23}$$

where p_{f3} is positive since $\rho_c(s_k) < 0$. The discriminant of the above equation is given by

$$(3 - \eta)^2 (s_k^T H_k s_k)^2 - 48(3 - 2\eta) g_k^T s_k p_{f3},$$

and as $\eta < 3/2$, it is always positive. In this case, the above equation as two roots of opposite sign. If α_η^* is the positive one, we then compute $\sigma_{k,\eta}^*$ from (3.22) with $\alpha = \alpha_\eta^*$. Figure 3.3 shows an example of this case.

Combining these different cases together, we are now able to state the complete rule for updating the current regularization parameter σ_k: it is described as Algorithm 3.1 on page 14. This algorithm also safeguards against the case where equations (3.14) and (3.19) do not admit a solution larger than $\sqrt[3]{\beta}$, or where such a solution exists but may be very much larger than this value, resulting in a tiny corresponding $\sigma_{k,\beta}^*$. In all these cases, we simply choose a fraction of the current σ_k. On the other hand, note that, by definition, the values of $\sigma_{k,\beta}^*$ computed in (3.24) and (3.25) are positive and smaller than the current σ_k. Figure 3.4 shows the value of σ_{k+1} computed by Algorithm 3.1 as a function of the objective function value $f(x_k + s_k)$. This curve for σ_{k+1} is a piecewise linear function where the sloping pieces correspond to values of σ_{k+1} computed by the interpolation rules (3.24)–(3.26).

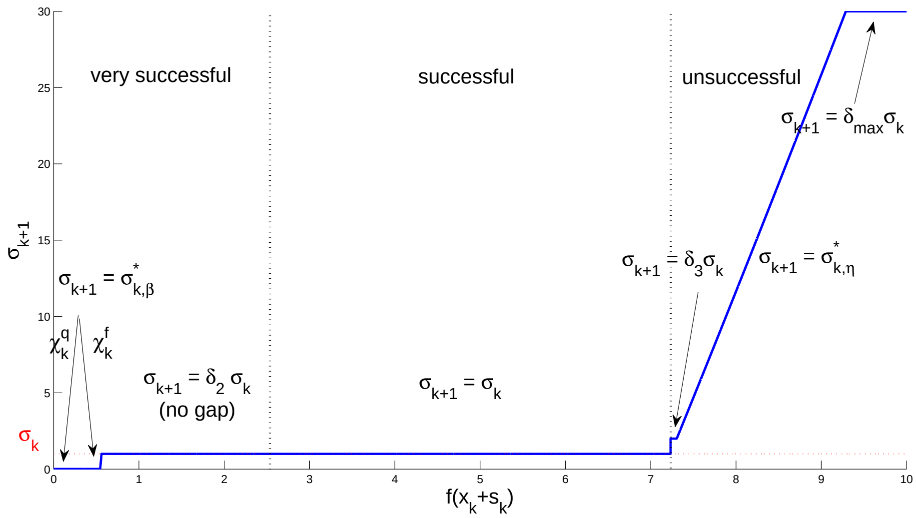

Figure 3.4: Plot of σ_{k+1}, computed by Algorithm 3.1 with parameters $\beta = 0.01,\ \alpha_{max} = 2,\ \epsilon_\chi = 10^{-8},\ \delta_1 = 0.1,\ \delta_2 = 1,\ \eta = \eta_1,\ \delta_3 = 2,\ \delta_{max} = 30$, as a function of $f(x_k + s_k)$.

4 Numerical experiments

We now present numerical experiments on nonlinear least-squares problems (1.2), where we study the numerical behaviour of the trust-region and the ARC algorithms employing the different updating rules presented in Section 3 in a first stage, and, in a second stage, compare the two algorithms using the best performing rules.

To compare the overall computational effort of the algorithms we use the performance profiles proposed by Dolan and Moré [9] for a given set of test problems and a given selection of algorithms. For each problem P in our testing set and each Algorithm A, we let $\texttt{fe}_{P,A}$ denote the number of function evaluations required to solve problem P using Algorithm A and $\texttt{fe}_P$ be number of function evaluations required by the best algorithm to solve problem P, i.e. the algorithm which uses the fewest function evaluations. The performance profile is defined for the algorithm A as

$$\pi_A(\tau) = \frac{\text{number of problems s.t. } \texttt{fe}_{P,A} \le \tau\, \texttt{fe}_P}{\text{number of problems}}, \qquad \tau \ge 1. \tag{4.1}$$

In what follows and in order to improve readability of the performance profile graphs, we limit the plot $\pi_A(\tau)$ to the interval $[1, 4]$ and report the number of failures in the legend.

4.1 The problem set

Numerical results are given for problems from the CUTEr test collection [15]. The test examples we consider are constructed using the CUTEr interactive select tool in order to locate the problems with no objective function and with constraints that are systems of nonlinear equations. We exclude the problems CHEMRCTA, CHEMRCTB, DRCAVTY3,

Algorithm 3.1: Regularization parameter update

Given the current x_k, s_k, σ_k, let the constants η_1 and η_2 be fixed by Algorithm 2.1. Let the positive threshold ϵ_χ and the constants $\delta_1, \delta_2, \delta_3, \delta_{max}, \beta, \eta$ be chosen such that

$$0 < \delta_1 < \delta_2 \leq 1 \leq \delta_3 \ll \delta_{max}, \quad 0 < \beta < 1, \quad 0 < \eta < 3/2, \quad 1 < \alpha_{max}.$$

Compute $\rho_c(s_k)$ by (2.7) and

$$\chi_k = c_k(s_k) - \max\left\{f(x_k + s_k), q_k(s_k)\right\}.$$

- If $\rho_c(s_k) \geq 1$ and $\chi_k \geq \epsilon_\chi$, then
  - If $f(x_k + s_k) \geq q_k(s_k)$, solve equation (3.14) with $\chi_k^f = \chi_k$. Let $\mathcal{A}^* = \{\alpha \mid \alpha \text{ is a root of (3.14) and } \alpha \geq \sqrt[3]{\beta}\}$.
    - $*$ If $\mathcal{A}^* = \emptyset$, set $\sigma_{k+1} = \max\{\delta_1\sigma_k, \epsilon_m\}$.
    - $*$ If $\mathcal{A}^* \neq \emptyset$, let $\alpha_\beta^* = \text{argmin}\{(\alpha - \sqrt[3]{\beta}) \mid \alpha \in \mathcal{A}^*\}$. If $\alpha_\beta^* \leq \alpha_{max}$, compute
$$\sigma_{k,\beta}^* = \sigma_k + 3\frac{\chi_k}{\|s_k\|^3}\left(\frac{\beta - \alpha_\beta^{*3}}{\alpha_\beta^{*3}}\right), \tag{3.24}$$
      and set $\sigma_{k+1} = \max\{\sigma_{k,\beta}^*, \epsilon_m\}$;
      If $\alpha_\beta^* > \alpha_{max}$, set $\sigma_{k+1} = \max\{\delta_1\sigma_k, \epsilon_m\}$.
  - Else if $f(x_k + s_k) < q_k(s_k)$, solve equation (3.19) with $\chi_k^q = \chi_k$. Let $\mathcal{A}^* = \{\alpha \mid \alpha \text{ is a root of (3.19) and } \alpha \geq \sqrt[3]{\beta}\}$.
    - $*$ If $\mathcal{A}^* = \emptyset$, set $\sigma_{k+1} = \max\{\delta_1\sigma_k, \epsilon_m\}$.
    - $*$ If $\mathcal{A}^* \neq \emptyset$, let $\alpha_\beta^* = \text{argmin}\{(\alpha - \sqrt[3]{\beta}) \mid \alpha \in \mathcal{A}^*\}$. If $\alpha_\beta^* \leq \alpha_{max}$, compute
$$\sigma_{k,\beta}^* = \frac{\beta}{\alpha_\beta^{*3}}\sigma_k, \tag{3.25}$$
      and set $\sigma_{k+1} = \max\{\sigma_{k,\beta}^*, \epsilon_m\}$;
      If $\alpha_\beta^* > \alpha_{max}$, set $\sigma_{k+1} = \max\{\delta_1\sigma_k, \epsilon_m\}$.
- Else if $\rho_c(s_k) \geq 1$ and $\chi_k < \epsilon_\chi$, set $\sigma_{k+1} = \max\{\delta_2\sigma_k, \epsilon_m\}$.
- Else if $\rho_c(s_k) \in [\eta_2, 1)$, set $\sigma_{k+1} = \max\{\delta_2\sigma_k, \epsilon_m\}$.
- Else if $\rho_c(s_k) \in [\eta_1, \eta_2)$, set $\sigma_{k+1} = \sigma_k$.
- Else if $\rho_c(s_k) \in [0, \eta_1)$, set $\sigma_{k+1} = \delta_3\sigma_k$.
- Else $(\rho_c(s_k) < 0)$, compute the positive root α_η^* of equation (3.23) and compute
$$\sigma_{k,\eta}^* = \frac{-g_k^T s_k - s_k^T H_k s_k \alpha_\eta^*}{\alpha_\eta^{*2}\|s_k\|^3}. \tag{3.26}$$
  Set $\sigma_{k+1} = \min\{\max\{\sigma_{k,\eta}^*, \delta_3\sigma_k\}, \delta_{max}\sigma_k\}$.

FLOSP2HH, FLOSP2HL, FLOSP2HM, FLOSP2TH, FLOSP2TL, FLOSP2TM, HYDCAR20, SEMICON2 and SEMICN2U as no algorithm succeeded in solving these problems for any tested parameter choice. For some CUTEr problems, we considered variants that differ in the dimensions (denoted with the superscript ^{2},^{3}). The resulting testing set consists of 95 problems of the form (1.2) whose names and dimensions are reported in Table 5.1 of Appendix. The problems ARGLALE, ARGBLE, GROWTH, HIMMELBD and OSCIPANE are large residual problems, i.e. the objective function value at the computed solution is much greater than one, the remaining are small or zero residual problems. Moreover, for 9 problems $m > n$, for 28 problems $m < n$, the remaining 58 ones being square.

4.2 Implementation issues

We implemented Algorithm 2.1 in Fortran 95, using the procedures presented in Section 2.1 to solve the subproblem at Step 1. We consider two implementations of the trust-region algorithm (TR-ST and TR-bST) which use the GALAHAD's package [16] LSTR and differ in the computation of the boundary trust-region solution: TR-ST computes the Steihaug-Toint point, TR-bST computes a more accurate solution as described in Section 2.1. The tested version of the ARC algorithm for solving problem (1.2) has been implemented using the GALAHAD's packages LSRT and it is denoted by ARC-LS.

In Algorithm 2.1, we set the specific algorithmic constants

$$\eta_1 = 0.01, \;\; \eta_2 = 0.95, \tag{4.2}$$

and the initial regularization parameters Δ_0 and σ_0 are chosen equal to one. The algorithm is terminated as soon as either

$$\|J_k^T h_k\| \le \max\{\epsilon_{ga},\, \epsilon_{gr}\, \|J_0^T h_0\|\} \quad \text{or} \quad \|h_k\| \le \max\{\epsilon_{fa},\, \epsilon_{fr}\, \|h_0\|\}, \tag{4.3}$$

where $\epsilon_{fa}, \epsilon_{ga}, \epsilon_{fr}, \epsilon_{gr} > 0$ are tolerances chosen as $\epsilon_{fa} = \epsilon_{ga} = 10^{-6}$, $\epsilon_{fr} = \epsilon_{gr} = 10^{-12}$. Moreover, we require that the trial step s_k computed at Step 1 of Algorithm 2.1 satisfies the inexact stopping criterion given by

$$\|\nabla m_k(s_k)\| \le \min\{\epsilon_{in},\, \|\nabla m_k(0)\|^{1/2}\}\|\nabla m_k(0)\|, \tag{4.4}$$

where m_k represents the models c_k in (2.11) and q_k in (2.10) and $\epsilon_{in} = 10^{-1}$ is fixed. If the problem dimension n is lower than 50, we allow for the generation of the full space in the Krylov sequence in order to compute a very accurate solution of the subproblems (2.12) and (2.13). Furthermore, any run exceeding 2 hours of CPU time, performing more than 5000 outer iterations or if the magnitude of computed search direction is lower than $10\epsilon_m$, is considered a failure. All other parameters in the GALAHAD's packages are set at their default values.

All our tests were performed on an Intel Xeon (TM) 3.4 Ghz, 1GB of RAM; the codes are all double precision, and compiled under g95 without optimization (default).

4.3 Numerical results

We consider first the trust-region algorithm and the trust-region radius updating rules described in Section 3. In particular we compare the updating rules (3.1) and (3.4) where we used parameter values given by

$$\gamma_1 = 1/2, \ \ \gamma_2 = 2, \ \ \gamma_3 = 0.0625, \tag{4.5}$$

and tried the values η_1 and η_2 given in (4.2) for the parameter η in (3.3).

In Figure 4.1, the function evaluation performance profiles show that both TR-ST and TR-bST are slightly more efficient using the updating rule (3.4) with $\eta = \eta_1$. Moreover, TR-bST is also a little more robust with this choice. The performance profile of Figure 4.2 summarizes the comparison between the two trust-region implementations using the best performing rule with the best parameter choice. As one might hope, the figure suggests that the extra effort required to solve the subproblem more accurately appears to offer some overall benefit.

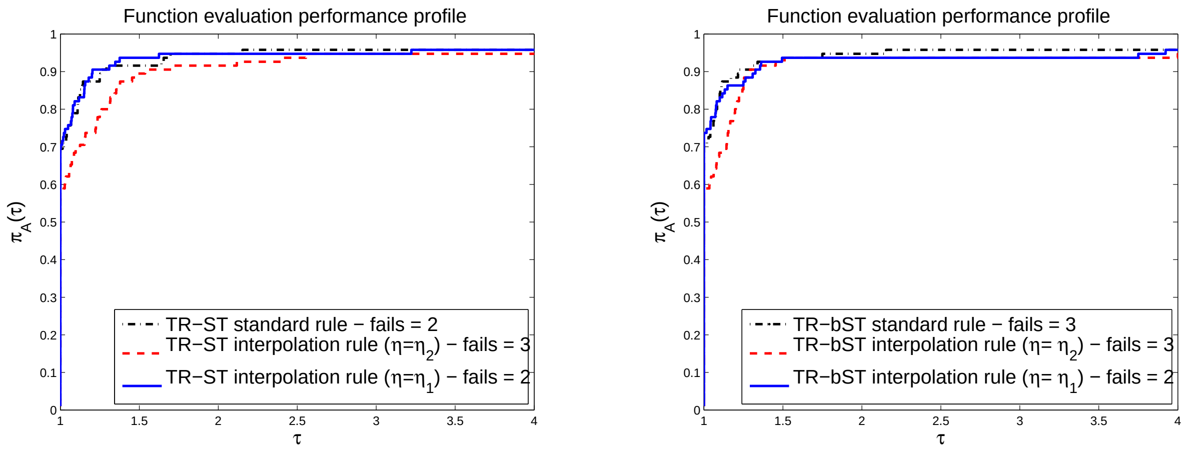

Figure 4.1: The function evaluation performance profile: TR-ST (left) and TR-bST (right) with (3.1) ("standard rule") and (3.4) using $\eta = \eta_1, \eta_2$ ("interpolation rule").

We now examine the sensitivity in number of function evaluations for the parameter choices of the new updating rule for σ_k for the ARC algorithm. To this purpose, we performed a small parametric study starting from the following reasonable values for the parameters in Algorithm 3.1

$$\beta = 1/100, \ \alpha_{max} = 2, \ \epsilon_\chi = 10^{-8}, \ \delta_1 = 1/10, \ \delta_2 = 1, \ \eta = \eta_1, \ \delta_3 = 2, \ \delta_{max} = 100, \tag{4.6}$$

and varying one parameter at the time in some set to find the best performing value.

More precisely, let all the parameters be ordered as β, α_{max}, ϵ_χ, δ_1, δ_2, η, δ_3, δ_{max} and be fixed as in (4.6). Let p be a parameter to be analyzed. Moreover, let $I_p = \{p_1, \ldots, p_q\}$ be a set of trial values for p, A_{p_i} be the ARC-LS algorithm run with $p = p_i$ and let $\pi_{A_{p_i}}(\tau)$ be the performance measure defined in (4.1) comparing the algorithms A_{p_i}, $p_i \in I_p$. To

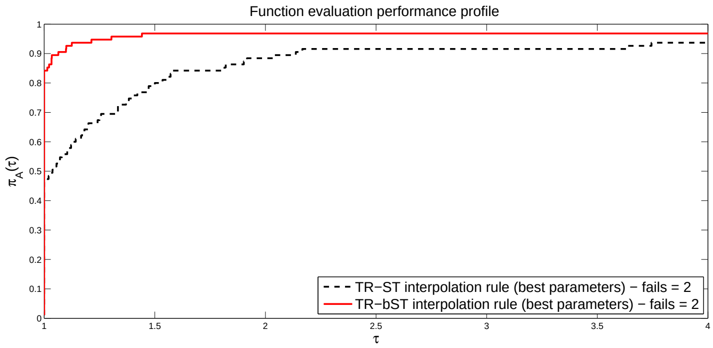

Figure 4.2: The function evaluation performance profile: `TR-ST` and `TR-bST` with the interpolation rule (3.4) and the best parameter choice ($\eta = \eta_1$).

estimate the efficiency of these algorithms, we compute the percentage of problems ($\%\texttt{pb}_{\hat\tau}$) for which $\pi_{A_{p_i}}(\tau) \le \hat\pi$ with $\hat\pi \gtrsim 1$ and to evaluate their robustness, we compute the number of failures. Taking into account these performance measures, we fix the "best" value for the parameter $p \in I_p$ and we proceed with the analysis of the subsequent parameter in the list. In Table 4.1, we report the sets I_p for all the parameters in Algorithm 3.1, the efficiency measure ($\%\texttt{pb}_{\hat\tau}$) for $\hat\tau = 1, 1.15, 1.25, 1.5, 2$ and the number of failures (`#fails`). We note that a more sophisticated choice, in which the globally optimal parameters for our test set is determined [1], is possible but has not been performed.

For each set I_p, it is quite easy to find the best performing parameter choice. It results from Table 4.1 that the new updating rule is not very sensitive to the parameter choice and that `ARC-LS` performs slightly better with the following parameter assignment:

$$\beta = 1/100,\ \alpha_{max} = 2,\ \epsilon_\chi = 10^{-10},\ \delta_1 = 1/10,\ \delta_2 = 1,\ \eta = \eta_1,\ \delta_3 = 2,\ \delta_{max} = 100. \quad (4.7)$$

We remark that in the experiments, a solution α^*_β of equations (3.14) and (3.19) was always found and that only in few cases this values was larger than α_{max}. Moreover, the value $\sigma^*_{k,\eta}$ computed by (3.26) was very often positive and lower than the current σ_k. Consequently, the regularization parameter was in fact updated by using the proposed interpolation techniques most of the time.

In Figure 4.3, `ARC-LS` using Algorithm 3.1 and the parameters in (4.7) is compared with `ARC-LS` using the old rule (3.5) and $\gamma = 2$ employed in [3]. The new rule clearly outperforms the old one. A possible explanation of the relatively poor behaviour of `ARC-LS` with the old rule may be found in what follows. In the experiments, we noticed that the norm of the gradient oscillates considerably for some problems, resulting in high oscillations in the updated σ_k through the iterations. Furthermore, we observed that, using (3.5), σ_k was updated in several runs using a small $\|g_k\|$ and hence was considerably reduced; the next iterate was then unsuccessful and doubling σ_k to recover an acceptable σ_k gave rise to many unsuccessful iterations.

p	I_p	`#fails`	$\%\mathtt{pb}_{\hat{\tau}}, \hat{\tau} = 1$	$\%\mathtt{pb}_{\hat{\tau}}, \hat{\tau} = 1.15$	$\%\mathtt{pb}_{\hat{\tau}}, \hat{\tau} = 1.25$	$\%\mathtt{pb}_{\hat{\tau}}, \hat{\tau} = 1.5$	$\%\mathtt{pb}_{\hat{\tau}}, \hat{\tau} = 2$
	0.001	3	58.95	90.53	92.63	93.68	95.79
	0.005	4	55.79	86.32	92.63	93.68	95.79
β	$\boxed{0.01}$	3	68.42	91.58	95.79	95.79	96.84
	0.05	3	51.58	82.11	89.47	93.68	95.79
	0.1	4	51.58	81.05	89.47	95.79	95.79
	1	4	42.11	69.47	75.79	91.58	94.74
	$\boxed{2}$	3	60.00	88.42	92.63	94.74	94.74
	3.5	3	55.79	85.26	90.53	92.63	92.63
α_{max}	5	3	61.05	84.21	90.53	92.63	92.63
	10	3	63.16	86.32	90.53	91.58	92.63
	50	4	61.05	84.21	89.47	90.53	91.58
	10^{-12}	3	81.05	92.63	93.68	95.79	95.79
	10^{-11}	3	83.16	93.68	95.79	95.79	96.84
	$\boxed{10^{-10}}$	2	80.00	95.79	96.84	96.84	97.89
ϵ_χ	10^{-9}	3	76.84	92.63	93.68	94.74	96.84
	10^{-8}	3	73.68	92.63	92.63	94.74	94.74
	10^{-6}	4	65.26	78.95	82.11	86.32	90.53
	0.01	6	69.47	85.26	88.42	91.58	92.63
	0.05	3	65.26	92.63	94.74	95.79	95.79
δ_1	$\boxed{0.1}$	2	71.58	94.74	97.89	97.89	97.89
	0.25	3	62.11	87.37	93.68	95.79	95.79
	0.5	3	58.95	83.16	92.63	95.79	95.79
	0.25	3	61.05	74.74	85.26	91.58	94.74
	0.5	3	53.68	78.95	85.26	90.53	96.84
δ_2	0.75	4	57.89	86.32	91.58	94.74	95.79
	0.9	4	57.89	85.26	90.53	93.68	95.79
	$\boxed{1}$	2	58.95	89.47	95.79	97.89	97.89
	$\boxed{\eta_1}$	2	72.63	94.74	95.79	96.84	97.89
	$(\eta_2 - \eta_1)/2$	4	68.42	88.42	91.58	94.74	95.79
η	η_2	5	62.11	81.05	88.42	91.58	94.74
	1.25	3	63.16	78.95	87.37	89.47	90.53
	1.50	4	69.47	89.47	93.68	94.74	94.74
	$\boxed{2}$	2	72.63	93.68	96.84	97.89	97.89
δ_3	2.5	4	66.32	89.47	91.58	94.74	95.79
	3	4	65.26	88.42	94.74	95.79	95.79
	4	3	66.32	83.16	92.63	94.74	96.84
	10	4	66.32	86.32	90.53	91.58	93.68
	50	4	70.53	89.47	93.68	95.79	95.79
δ_{max}	$\boxed{100}$	2	74.74	95.79	97.89	97.89	97.89
	500	4	71.58	91.58	93.68	93.68	94.74
	1000	4	72.63	88.42	93.68	93.68	94.74

Table 4.1: Parametric study.

Finally, we compare `TR-ST`, `TR-bST` and `ARC-LS` using the best performing updating rules for the regularization parameters, i.e. for the trust-region radius Δ_k the rule (3.4) with the parameters in (4.5) and $\eta = \eta_1$ and for the regularization parameter σ_k, the rule presented in Algorithm 3.1 with the parameter choice (4.7). The corresponding function evaluation performance profiles are plotted in Figures 4.4 and 4.5. `ARC-LS` fails on problems `ARWHDNE`, `DRCAVITY2`, `TR-bST` on problems `DRCAVITY2`, `POROUS2` and `TR-ST` on problems `QR3D`[2], `POROUS2`. Evidently, `ARC-LS` is much more efficient than `TR-ST`. Compared to `TR-bST`, it is better 68.42% of the runs and `TR-bST` is within a factor 2 of `ARC-LS` for the 88.10% of the runs.

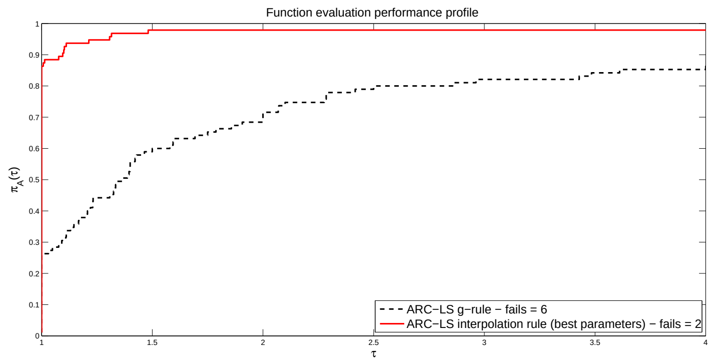

Figure 4.3: The function evaluation performance profile: `ARC-LS` with (3.5) ("g-rule") and `ARC-LS` with Algorithm 3.1 and parameters (4.7) ("interpolation rule (best parameters)").

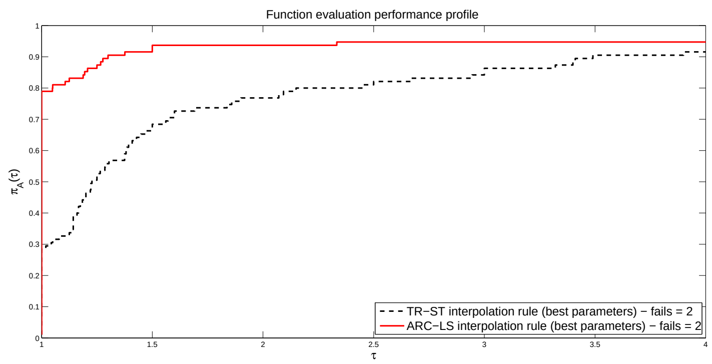

Figure 4.4: The function evaluation performance profile: `TR-ST` rule (3.4) with $\eta = \eta_1$ ("interpolation rule (best parameters)") and `ARC-LS` with Algorithm 3.1 and parameters (4.7) ("interpolation rule (best parameters)").

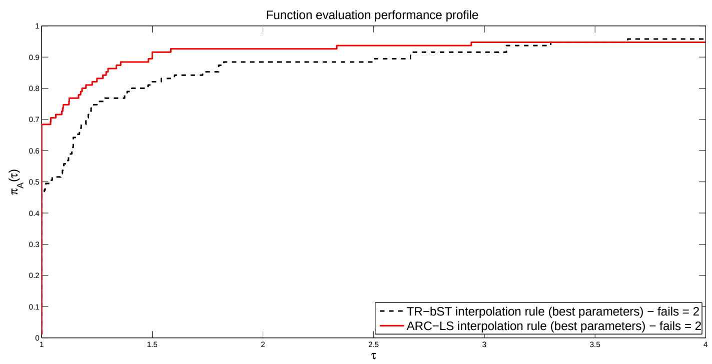

Figure 4.5: The function evaluation performance profile: `TR-bST` rule (3.4) with $\eta = \eta_1$ ("interpolation rule (best parameters)") and `ARC-LS` with Algorithm 3.1 and parameters (4.7) ("interpolation rule (best parameters)").

We report in the Appendix the complete set of results of the experiments described in this section.

We also considered strategies for choosing the initial regularization parameter σ_0 along the lines of the strategy proposed in [22] for automatically computing the initial trust-region radius. In particular, we tested a strategy in which one solves a one-dimensional minimization problem (along the steepest descent direction) in the hope of estimating a better value of σ_0 for starting the minimization in the full space. However, these experiments (not reported here) produced disappointing results in that it turned out to be generally better to start minimization in the full-space from the start and not "waste" additional function evaluations for estimating σ_0. This is not entirely unexpected in our context where we assume the cost of function evaluation to dominate the inner linear algebra calculations. But it is also clear that any *a priori* user estimation of the Hessian Lipschitz constant can be usefully exploited by selecting σ_0 appropriately.

5 Conclusion

In this paper we propose a new reliable strategy to update the regularization parameter in the cubic regularization algorithm (ARC). This strategy is based on analyzing the adequacy between the objective function and its cubic model, and exploits its overestimation property. Moreover, it has the favorable feature of not requiring extra function values. We report numerical tests which show that the new rule considerably improves the numerical performance of the ARC algorithm. We also provide a numerical comparison between the ARC and trust-region frameworks on a set of large nonlinear least-squares CUTEr problems. These suggest a numerical advantage of the former on our set of test problems.

Acknowledgements

The second author wishes to thank Stefania Bellavia and Benedetta Morini for several helpful discussions and for their continued encouragement and support.

References

[1] C. Audet, D. Orban, *Finding optimal algorithmic parameters using derivative-free optimization*, SIAM Journal on Optimization, 17(3):642–664, 2006.

[2] S. Bellavia, C. Cartis, N. I. M. Gould, B. Morini, Ph. L. Toint, *Convergence of a Regularized Euclidean Residual Algorithm for Nonlinear Least-Squares*, SIAM Journal on Numerical Analysis, 48:1–29, 2010.

[3] C. Cartis, N.I.M. Gould, Ph.L. Toint, *Adaptive cubic overestimation methods for unconstrained optimization. Part I: motivation, convergence and numerical results*, Mathematical Programming, Ser. A, (2009), DOI: 10.1007/s10107-009-0286-5.

[4] C. Cartis, N.I.M. Gould, Ph.L. Toint, *Adaptive cubic overestimation methods for unconstrained optimization. Part II: Worst-case function- and derivative-evaluation complexity*, Mathematical Programming, Ser. A, (2010a), DOI: 10.1007/s10107-009-0337-y.

[5] C. Cartis, N.I.M. Gould, Ph.L. Toint, *Complexity bounds for second-order optimality in unconstrained optimization*, Technical Report NAXYS-11-2010, Department of Mathematics, FUNDP - University of Namur, Namur, Belgium, 2010.

[6] C. Cartis, N.I.M. Gould, Ph.L. Toint, *On the complexity of steepest descent, Newton's and regularized Newton's methods for nonconvex unconstrained optimization*, SIAM Journal on Optimization, 20(6):2833–2852, 2010.

[7] C. Cartis, N.I.M. Gould, Ph.L. Toint, *Trust-region and other regularisations of linear least-squares problems*, BIT, 49(1):21-53, 2009.

[8] A. R. Conn, N. I. M. Gould and Ph. L. Toint, *Trust-Region Methods*, SIAM, Philadelphia, USA, 2000.

[9] E.D. Dolan, J.J. Moré, *Benchmarking optimization software with performance profiles*, Mathematical Programming, 91:201–213, 2002.

[10] J.E. Dennis, R.B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*, Prentice Hall, Englewood Cliffs, NJ, 1983.

[11] J.B. Erway, P.E. Gill, *A Subspace Minimization Method for the Trust-Region Step*, SIAM Journal on Optimization, 20:1439–1461, 2009.

[12] J. B. Erway, P. E. Gill, J. D. Griffin, *Iterative Methods for Finding a Trust-Region Step*, SIAM Journal on Optimization, 20:1110–1131, 2009.

[13] G. H. Golub, W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, SIAM Journal on Numerical Analysis, 2(2):205–224, 1965.

[14] N.I.M. Gould, S. Lucidi, M. Roma, Ph.L. Toint *Solving the trust-region subproblem using the Lanczos method*, SIAM Journal on Optimization, 9(2):504–525, 1999.

[15] N.I.M. Gould, D. Orban, Ph.L. Toint, *CUTEr, a constrained and unconstrained testing environment, revisited*, ACM Transactions on Mathematical Software, 29(4):373–394, 2003.

[16] N.I.M. Gould, D. Orban, Ph.L. Toint, GALAHAD—*a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization*, ACM Transactions on Mathematical Software, 29(4):353–372, 2003.

[17] A. Griewank, *The modification of Newton's method for unconstrained optimization by bounding cubic terms*, Technical Report NA/12 (1981), Department of Applied Mathematics and Theoretical Physics, University of Cambridge, United Kingdom, 1981.

[18] W.W. Hager, S.C. Park, *Global convergence of SSM for minimizing a quadratic over a sphere*, Mathematics of Computation, 74:1413–1423, 2005.

[19] W.W. Hager, *Minimizing a quadratic over a sphere*, SIAM Journal on Optimization, 12:188-208, 2001.

[20] Yu. Nesterov, B.T. Polyak, *Cubic regularization of Newton's method and its global performance*, Mathematical Programming, 108(1):177–205, 2006.

[21] C.C. Paige, M.A. Saunders, *ALGORITHM 583: LSQR: an algorithm for sparse linear equations and sparse least squares*, ACM Transactions on Mathematical Software, 8(2):195–209, 1982.

[22] A. Sartenaer, *Automatic determination of an initial trust region in nonlinear programming*, SIAM Journal on Scientific Computing, 18(6):1788–1803, 1997.

[23] T. Steihaug, *The conjugate gradient method and trust regions in large scale optimization*, SIAM Journal on Numerical Analysis, 20:626–637, 1983.

[24] M. Weiser, P. Deuflhard, B. Erdmann, *Affine conjugate adaptive Newton methods for nonlinear elastomechanics*, Optimization Methods and Software, 22(3):413–431, 2007.

[25] Y. Yuan, *On the Truncated Conjugate-Gradient Method*, Mathematical Programming Ser. A, 87(3):561–573, 1999.

Appendix

Table 5.1 contains the problem set information (name and dimensions). Tables 5.2-5.6 collect all the results of the experiments described in Section 4: we reported the total number of function evaluation for each method and algorithmic option tested and we used the following symbols for the failures: '$*$' for the time exceeding runs, '>' for the runs exceeding the maximum number of iteration allowed, 'ss' if the norm of the search step is below the fixed treshold.

Name	n	m	Name	n	m	Name	n	m
AIRCRFTA	8	5	DECONVNE	61	41	OSCIPANE	500	500
ARGAUSS	3	15	DRCAVTY1	196	100	PFIT1	2	2
ARGLALE	200	400	DRCAVTY2	4489	3969	PFIT2	2	2
ARGLBLE	200	400	EIGENA	110	110	PFIT3	2	2
ARGTRIG	200	200	EIGENA^2	2550	2550	PFIT4	2	2
ARTIF	502	500	EIGENA^3	4970	4970	POROUS1	1024	900
ARTIF^2	5002	5000	EIGENB	110	110	POROUS1^2	5184	4900
ARWDHNE	500	998	EIGENB^2	2550	2550	POROUS1^3	22500	21904
BDVALUES	102	100	EIGENC	462	462	POROUS2	1024	900
BDVALUS^2	5002	5000	EIGENC^2	2652	2652	POROUS2^2	5184	4900
BOOTH	2	2	GOTTFR	2	2	POROUS2^3	22500	21904
BRATU2D	484	400	GROWTH	3	12	POROUS2^4	62500	61504
BRATU2D^2	5184	4900	HATFLDF	3	3	POWELLBS	2	2
BRATU2DT	484	400	HATFLDG	25	25	POWELLSQ	2	2
BRATU2DT^2	5184	4900	HEART6	6	6	QR3D	610	610
BRATU3D	1000	512	HEART8	8	8	QR3D^2	2420	2420
BRATU3D^2	4913	3375	HIMMELBA	2	2	QR3DBD	457	610
BROWNALE	200	200	HIMMELBC	2	2	QR3DBD^2	1717	2420
BROWNALE^2	1000	1000	HIMMELBD	2	2	RECIPE	3	3
BROYDN3D	1000	1000	HIMMELBE	3	3	SINVALNE	2	2
BROYDN3D^2	10000	10000	HS8	2	2	SPMSQRT	10000	16664
BROYDNBD	1000	1000	HYDCAR6	29	29	TRIGGER	7	6
BROYDNBD^2	10000	10000	HYPCIR	2	2	WOODSNE	10000	7501
CBRATU2D	3200	2888	INTEGREQ	102	100	YATP1SQ	2600	2600
CBRATU3D	3456	2000	INTEGREQ^2	502	500	YATP1SQ^2	40400	40400
CHANDHEQ	100	100	METHANB8	31	31	YATP1SQ^3	63000	63000
CHANNEL	2400	2398	METHANL8	31	31	YATP2SQ	2600	2600
CHANNEL^2	9600	9598	MSQRTA	4900	4900	YATP2SQ^2	40400	40400
CHNRSBNE	50	98	MSQRTA^2	5625	5625	YATP2SQ^3	63000	63000
CLUSTER	2	2	MSQRTB	4900	4900	YFITNE	3	17
COOLHANS	9	9	MSQRTB^2	5625	5625	ZANGWIL3	3	3
CUBENE	2	2	NYSTROM5	18	20			

Table 5.1: The problem set.

Name	TR-ST standard	TR-ST interpolation η_2	TR-ST interpolation η_1	TR-bST standard	TR-bST interpolation η_2	TR-bST interpolation η_1
AIRCRFTA	4	4	4	4	4	4
ARGAUSS	2	2	2	2	2	2
ARGLALE	6	6	6	6	6	6
ARGLBLE	5	5	5	5	5	5
ARGTRIG	9	9	9	9	9	9
ARTIF	20	20	20	22	20	22
ARTIF^2	25	25	25	17	17	17
ARWDHNE	398	ss	ss	ss	ss	172
BDVALUES	43	43	43	62	62	62
BDVALUES^2	391	391	391	416	416	416
BOOTH	4	4	4	4	4	4
BRATU2D	7	7	7	5	5	5
BRATU2DT^2	12	12	12	10	10	10
BRATU2DT	24	19	22	14	13	14
BRATU2D^2	9	9	9	6	6	6
BRATU3D	8	8	8	7	7	7
BRATU3D^2	10	10	10	8	8	8
BROWNALE	6	6	6	6	6	6
BROWNALE^2	8	8	8	8	8	8
BROYDN3D	9	9	9	9	9	9
BROYDN3D^2	11	11	11	11	11	11
BROYDNBD	19	19	19	18	18	18
BROYDNBD^2	27	29	27	19	19	19
CBRATU2D	8	8	8	6	6	6
CBRATU3D	9	9	9	8	8	8
CHANDHEQ	15	15	15	14	14	14
CHANNEL	213	451	293	154	185	154
CHANNEL^2	270	381	159	106	103	115
CHNRSBNE	61	75	64	48	60	48
CLUSTER	8	8	8	8	8	8
COOLHANS	890	827	538	604	777	653
CUBENE	6	6	6	6	6	6
DECONVNE	16	18	16	12	12	12
DRCAVTY1	41	44	41	35	40	35
DRCAVTY2	*	*	490	*	*	*
EIGENA	21	21	21	20	20	20
EIGENA^2	108	109	106	72	79	72
EIGENA^3	166	171	171	84	87	84
EIGENB	131	176	141	133	154	143
EIGENB^2	1047	1283	1050	687	862	936
EIGENC	93	102	104	70	79	66
EIGENC^2	834	751	904	245	263	249
GOTTFR	11	16	11	11	16	11
GROWTH	54	71	54	11	71	54
HATFLDF	9	23	29	8	32	30
HATFLDG	8	8	8	9	11	9
HEART6	484	558	528	617	687	580
HEART8	46	49	53	38	46	48

Name	TR-ST standard	TR-ST interpolation η_2	TR-ST interpolation η_1	TR-bST standard	TR-bST interpolation η_2	TR-bST interpolation η_1
HIMMELBA	4	4	4	4	4	4
HIMMELBC	6	6	6	6	6	6
HIMMELBD	41	25	30	42	24	30
HIMMELBE	4	4	4	4	4	4
HS8	7	7	7	6	6	6
HYDCAR6	458	530	600	425	381	438
HYPCIR	5	5	5	5	5	5
INTEGREQ	5	5	5	5	5	5
INTEGREQ^2	5	5	5	5	5	5
METHANB8	94	94	94	94	94	94
METHANL8	237	189	204	266	178	266
MSQRTA	46	44	45	44	46	46
MSQRTA^2	54	57	55	54	61	53
MSQRTB	47	44	47	44	44	44
MSQRTB^2	52	51	50	47	54	49
NYSTROM5	223	268	198	140	149	129
OSCIPANE	8	8	8	8	8	8
PFIT1	13	105	109	13	105	51
PFIT2	28	13	13	28	13	13
PFIT3	10	11	9	11	11	9
PFIT4	14	169	14	9	9	9
POROUS1	47	51	44	41	39	35
POROUS1^2	190	251	147	90	86	81
POROUS1^3	822	1016	888	168	183	154
POROUS2	2174	149	1788	>	>	>
POROUS2^2	195	291	230	106	137	120
POROUS2^3	1045	1204	914	200	281	186
POROUS2^4	2749	3124	2478	299	364	271
POWELLBS	87	113	82	69	79	74
POWELLSQ	98	19	18	109	19	108
QR3D	621	497	573	186	165	153
QR3D^2	>	>	>	915	700	916
QR3DBD	342	307	353	74	92	74
QR3DBD^2	1252	1234	1249	678	506	686
RECIPE	24	33	39	18	21	18
SINVALNE	27	30	24	25	30	24
SPMSQRT	15	15	15	15	15	15
TRIGGER	8	8	8	8	8	8
WOODSNE	42	39	39	35	33	33
YATP1SQ	41	47	36	29	30	29
YATP1SQ^2	28	29	30	26	27	25
YATP1SQ^3	28	27	27	26	24	25
YATP2SQ	30	30	30	31	31	31
YATP2SQ^2	34	34	34	33	33	33
YATP2SQ^3	30	30	30	31	31	31
YFITNE	46	50	62	46	50	61
ZANGWIL3	8	8	8	8	8	8

Table 5.2: Results for `TR-ST` and `TR-bST`.

ARC-LS

Name	β 0.001	0.005	0.01	0.5	0.1	α_{max} 1	3.5	5	10	50	ϵ_χ 10^{-12}	10^{-11}	10^{-10}	10^{-9}	10^{-6}	δ_1 0.01	0.05	0.25	0.5
AIRCRFTA	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
ARGAUSS	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
ARGLALE	6	6	6	6	6	6	5	5	4	4	6	6	6	6	6	5	5	6	7
ARGLBLE	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
ARGTRIG	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
ARTIF	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	20	20	23	24
ARTIF^2	23	20	20	20	23	23	19	19	18	18	20	20	20	20	20	19	19	19	21
ARWDHNE	ss	ss	ss	ss	ss	ss	ss	ss	ss	ss	ss	ss	ss	ss	ss	ss	ss	ss	ss
BDVALUES	39	63	47	49	44	39	47	47	47	47	34	34	34	34	47	34	34	34	34
BDVALUES^2	*	*	*	*	*	*	*	*	*	270	*	*	270	*	*	*	*	*	*
BOOTH	4	4	4	4	5	5	4	4	4	4	4	4	4	4	4	4	4	4	4
BRATU2D	6	6	6	7	7	8	6	6	6	6	6	6	6	6	6	6	6	7	7
BRATU2D^2	6	6	6	9	9	10	6	6	6	6	6	6	6	6	6	6	6	6	6
BRATU2DT	14	15	14	15	15	19	14	14	14	14	14	14	14	14	14	14	14	14	14
BRATU2DT^2	10	10	10	11	13	13	10	10	10	10	10	10	10	10	10	10	10	10	10
BRATU3D	7	7	7	7	7	8	7	7	7	7	7	7	7	7	7	7	7	7	7
BRATU3D^2	7	7	7	7	8	8	7	7	7	7	7	7	7	7	7	7	7	7	7
BROWNALE	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
BROWNALE^2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
BROYDN3D	8	8	8	8	8	7	8	8	8	8	8	8	8	8	8	8	8	8	8
BROYDN3D^2	9	9	9	9	9	8	9	9	9	9	9	9	9	9	9	9	9	9	9
BROYDNBD	10	10	10	10	10	9	10	10	10	10	10	10	10	10	10	10	10	10	10
BROYDNBD^2	11	11	11	11	11	10	11	11	11	11	11	11	11	11	11	11	11	11	11
CBRATU2D	6	7	7	7	9	9	7	7	7	7	7	7	7	7	7	7	7	7	7
CBRATU3D	7	7	7	7	8	8	7	7	7	7	7	7	7	7	7	7	7	7	7
CHANDHEQ	19	19	19	19	19	15	19	19	19	19	19	19	19	19	20	17	18	19	19
CHANNEL	142	147	140	146	136	147	145	145	142	142	139	139	140	140	206	143	147	148	148
CHANNEL^2	99	97	94	95	98	102	101	101	101	101	101	101	94	94	94	106	101	98	95
CHNRSBNE	40	40	41	41	41	40	41	41	41	41	41	41	41	41	41	41	41	41	41
CLUSTER	10	10	10	10	10	9	10	10	10	10	10	10	10	10	11	10	10	10	10
COOLHANS	381	462	467	530	544	560	467	21	21	21	431	431	441	467	>	577	434	455	571
CUBENE	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14
DECONVNE	38	36	36	38	39	13	36	36	36	36	17	17	19	33	36	19	19	19	19
DRCAVTY1	39	38	39	37	38	42	39	39	39	39	43	43	43	43	36	43	43	43	43
DRCAVTY2	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
EIGENA	23	21	20	21	22	20	20	20	20	20	20	20	20	20	20	20	20	20	20
EIGENA^2	75	73	72	74	73	70	77	77	77	77	72	72	72	72	73	73	71	71	68
EIGENA^3	92	92	92	91	91	86	92	92	92	92	92	92	92	92	92	93	92	91	93
EIGENB	171	140	143	150	135	200	147	147	147	147	179	172	156	149	141	151	152	151	161
EIGENB^2	679	770	721	1085	681	1043	709	709	709	709	773	765	760	739	1233	1074	1006	960	788
EIGENC	65	68	65	63	66	64	67	67	67	67	65	65	65	65	64	74	68	66	64
EIGENC^2	245	275	256	253	234	287	275	275	275	275	260	260	259	257	274	298	245	230	232
GOTTFR	8	8	8	9	8	10	8	8	8	8	8	8	8	8	8	8	8	8	8
GROWTH	58	56	53	57	59	56	53	53	53	53	53	53	53	53	53	56	55	55	57
HATFLDF	28	27	26	25	25	39	26	26	26	26	25	25	25	25	35	25	25	25	25
HATFLDG	11	9	7	9	8	8	7	7	7	7	7	7	7	7	7	7	7	7	7
HEART6	164	164	159	160	163	275	159	159	163	163	159	159	159	159	164	162	162	162	162
HEART8	18	18	18	18	18	27	18	18	18	18	18	18	18	18	18	18	18	18	18

Table 5.3: Results for ACO-LS: parametric study for the interpolation rule (Algorithm 3.1).

ARC-LS

	β					α_{max}					ϵ_χ					δ_1			
Name	0.001	0.005	0.01	0.5	0.1	1	3.5	5	10	50	10^{-12}	10^{-11}	10^{-10}	10^{-9}	10^{-6}	0.01	0.05	0.25	0.5
HIMMELBA	4	4	4	5	5	5	4	4	4	4	4	4	4	4	4	4	4	4	4
HIMMELBC	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
HIMMELBD	39	39	39	39	39	39	39	39	39	39	39	39	39	39	39	39	39	39	39
HIMMELBE	6	6	6	5	5	6	5	5	5	5	6	6	6	6	6	5	5	6	6
HS8	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
HYDCAR6	460	461	437	490	470	493	448	448	448	448	434	472	466	449	455	399	408	448	462
HYPCIR	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
INTEGREQ	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
INTEGREQ^2	5	5	5	5	5	6	5	5	5	5	5	5	5	5	5	5	5	5	5
METHANB8	149	148	78	142	101	104	78	78	78	78	163	152	120	112	159	120	120	120	120
METHANL8	235	218	229	213	205	287	260	260	207	207	213	170	189	189	303	189	189	189	189
MSQRTA	40	40	40	40	40	35	40	35	35	35	40	40	38	40	44	39	39	39	37
MSQRTA^2	50	50	46	50	50	47	46	46	46	46	45	45	45	47	52	48	51	49	46
MSQRTB	38	38	38	39	39	35	38	34	34	34	38	38	40	38	42	38	38	39	35
MSQRTB^2	46	46	46	46	46	43	46	43	43	43	44	44	44	43	50	44	45	45	44
NYSTROM5	175	15	13	14	15	13	12	12	12	12	13	13	13	13	22	12	13	13	14
OSCIPANE	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
PFIT1	160	155	150	176	168	152	2510	2510	2510	2510	150	150	150	150	1212	164	168	154	137
PFIT2	207	206	217	223	223	221	217	198	198	198	216	216	217	217	475	225	224	3349	221
PFIT3	272	295	280	287	272	276	280	4001	4001	4001	279	279	279	279	280	3880	284	318	297
PFIT4	379	381	406	367	395	367	373	373	411	386	403	403	403	404	412	412	385	380	383
POROUS1	36	35	32	34	37	42	32	32	32	32	32	32	32	32	32	32	32	32	32
POROUS1^2	78	76	80	83	93	82	80	80	80	80	80	80	80	80	80	80	80	81	81
POROUS1^3	175	164	175	169	168	161	186	186	186	186	175	175	173	175	175	181	175	173	175
POROUS2	48	>	74	108	>	>	74	78	146	146	74	74	74	74	74	>	1901	70	580
POROUS2^2	113	122	107	111	110	108	107	106	106	106	107	107	107	107	107	109	108	103	104
POROUS2^3	216	216	220	203	234	221	198	198	202	202	220	220	220	220	220	200	213	220	211
POROUS2^4	322	335	350	329	314	318	346	373	373	372	350	350	350	350	349	335	358	346	351
POWELLBS	169	192	181	180	160	168	181	181	181	181	90	88	99	133	528	99	99	99	99
POWELLSQ	13	13	13	13	13	13	417	417	417	417	13	13	13	13	13	13	13	13	13
QR3D	154	165	168	158	206	221	171	171	171	171	168	168	168	168	178	172	166	171	167
QR3D^2	1027	1019	1005	1022	1013	1041	1041	1041	1041	1041	956	954	954	969	1197	957	949	953	951
QR3DBD	64	71	65	63	70	70	65	65	65	65	64	64	64	64	65	64	64	64	64
QR3DBD^2	739	645	604	700	621	761	717	717	717	717	608	608	603	602	619	699	725	599	734
RECIPE	10	10	10	10	10	12	10	10	10	10	10	10	10	10	10	10	10	10	10
SINVALNE	23	23	23	23	23	21	23	23	23	23	23	23	23	23	23	23	23	23	23
SPMSQRT	12	12	12	12	12	12	12	12	12	14	12	12	12	12	12	12	12	13	14
TRIGGER	9	9	9	10	10	10	9	9	9	9	9	9	9	9	14	9	9	9	9
WOODSNE	28	28	28	28	28	29	28	28	28	28	28	28	28	28	28	28	28	28	28
YATP1SQ	47	47	43	48	41	46	43	43	43	43	43	43	43	43	43	43	43	43	43
YATP1SQ^2	23	23	23	23	23	24	23	23	23	23	23	23	23	23	23	23	23	23	23
YATP1SQ^3	23	22	22	20	20	20	22	22	22	22	22	22	22	22	22	22	22	22	22
YATP2SQ	10	10	10	10	10	10	10	10	10	>	10	10	10	10	10	>	11	12	11
YATP2SQ^2	10	10	10	10	10	9	8	8	8	8	10	10	10	10	10	20	10	15	12
YATP2SQ^3	10	10	10	10	10	10	10	10	10	>	10	10	10	10	10	>	11	12	11
YFITNE	44	44	44	44	44	42	41	41	41	41	44	44	44	44	44	42	43	46	52
ZANGWIL3	5	5	5	6	6	7	5	5	4	4	5	5	5	5	5	5	5	6	7

Table 5.4: Results for ACO-LS: parametric study for the interpolation rule (Algorithm 3.1).

`ARC-LS`

Name	δ_2 0.25	δ_2 0.5	δ_2 0.75	δ_2 0.9	η $(\eta_2-\eta_1)/2$	η η_2	η 1.25	δ_3 1.5	δ_3 2.5	δ_3 3	δ_3 4	δ_{max} 10	δ_{max} 50	δ_{max} 500	δ_{max} 1000	g-rule
AIRCRFTA	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
ARGAUSS	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
ARGLALE	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	15
ARGLBLE	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
ARGTRIG	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
ARTIF	20	21	22	21	20	19	19	22	18	22	23	29	21	21	21	30
ARTIF^2	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	42
ARWDHNE	ss	ss	ss	ss	ss	ss	ss	ss	ss	ss	ss	258	ss	ss	ss	795
BDVALUES	38	40	40	39	34	34	34	34	34	34	34	34	34	34	34	461
BDVALUES^2	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
BOOTH	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	7
BRATU2D	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	12
BRATU2D^2	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	24
BRATU2DT	14	14	14	14	14	14	14	15	14	14	14	14	14	14	14	29
BRATU2DT^2	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	54
BRATU3D	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	13
BRATU3D^2	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	24
BROWNALE	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
BROWNALE^2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
BROYDN3D	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
BROYDN3D^2	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	10
BROYDNBD	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
BROYDNBD^2	10	11	11	11	11	11	11	11	11	11	11	11	11	11	11	12
CBRATU2D	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	20
CBRATU3D	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	16
CHANDHEQ	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	26
CHANNEL	159	141	133	129	142	151	164	146	141	148	146	146	145	139	139	237
CHANNEL^2	111	98	91	89	105	102	107	99	98	99	102	101	96	94	97	214
CHNRSBNE	46	43	39	39	40	60	48	41	40	41	43	41	41	41	41	43
CLUSTER	9	9	10	10	10	10	10	10	10	10	10	10	10	10	10	9
COOLHANS	416	600	493	684	441	441	441	441	441	441	441	491	435	934	934	3403
CUBENE	14	14	14	14	14	14	14	14	14	14	14	14	13	14	14	13
DECONVNE	15	22	16	19	19	18	15	19	19	19	16	19	19	19	19	30
DRCAVTY1	42	43	42	47	45	42	49	43	43	43	42	61	47	40	40	53
DRCAVTY2	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
EIGENA	22	21	20	19	20	20	20	20	21	20	20	20	20	20	20	32
EIGENA^2	75	65	62	71	72	75	77	71	74	73	74	75	72	72	72	260
EIGENA^3	96	87	80	86	93	92	95	91	92	92	93	92	92	92	92	425
EIGENB	192	174	173	155	149	224	213	151	160	164	172	173	166	153	150	173
EIGENB^2	1034	1318	998	806	868	818	2309	757	792	762	822	758	954	878	1149	*
EIGENC	73	78	64	68	73	107	76	67	95	73	65	68	80	65	62	67
EIGENC^2	325	309	294	273	258	272	568	276	273	269	272	248	250	230	224	463
GOTTFR	12	9	8	9	9	12	12	8	8	9	10	8	8	8	8	16
GROWTH	53	55	55	56	53	93	186	60	54	52	86	53	53	53	53	157
HATFLDF	26	22	24	25	25	26	27	25	25	25	25	25	25	25	25	19
HATFLDG	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	16
HEART6	165	144	151	158	158	171	171	164	164	168	169	132	159	159	159	554
HEART8	28	28	23	24	18	18	18	18	18	18	18	18	18	18	18	21

Table 5.5: Results for `ACO-LS`: parametric study for the interpolation rule (Algorithm 3.1) and g-rule (3.5) (last column).

ARC-LS

	δ_2				η			δ_3				δ_{max}				g-rule
Name	0.25	0.5	0.75	0.9	$(\eta_2-\eta_1)/2$	η_2	1.25	1.5	2.5	3	4	10	50	500	1000	
HIMMELBA	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	8
HIMMELBC	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
HIMMELBD	38	38	40	40	39	29	23	57	33	29	26	40	39	40	40	51
HIMMELBE	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	8
HS8	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
HYDCAR6	569	450	443	512	424	488	484	461	460	447	431	494	480	485	486	534
HYPCIR	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
INTEGREQ	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
INTEGREQ^2	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	6
METHANB8	81	156	124	129	120	120	120	120	120	120	120	120	120	120	120	81
METHANL8	263	241	152	224	188	181	203	191	186	194	283	307	202	212	200	391
MSQRTA	38	38	38	38	40	40	40	40	40	40	40	40	40	40	40	53
MSQRTA^2	53	49	46	44	51	50	50	45	45	45	45	45	45	45	45	64
MSQRTB	36	36	37	36	38	38	38	38	38	38	38	38	38	38	38	53
MSQRTB^2	44	40	41	42	45	46	46	44	44	44	44	45	43	44	44	59
NYSTROM5	11	12	12	12	13	13	13	13	13	13	13	13	13	13	13	14
OSCIPANE	7	7	7	7	7	7	7	7	13	7	7	7	7	7	7	7
PFIT1	159	152	155	166	141	165	975	150	165	150	176	141	167	152	164	148
PFIT2	224	212	213	212	213	213	213	212	210	228	260	220	223	235	230	166
PFIT3	289	284	260	272	279	271	1892	264	291	327	317	3965	289	285	315	>
PFIT4	400	396	399	401	403	403	403	368	391	383	483	400	400	372	399	>
POROUS1	34	39	35	39	37	34	37	32	33	32	39	36	40	52	40	51
POROUS1^2	88	81	83	95	74	86	88	78	83	82	77	84	78	70	75	111
POROUS1^3	167	179	171	171	195	164	152	190	155	181	192	195	179	163	162	*
POROUS2	226	106	>	>	>	>	210	>	>	>	90	>	>	>	ss	61
POROUS2^2	97	114	125	110	128	127	116	114	104	112	115	222	111	119	129	125
POROUS2^3	230	219	227	188	223	216	226	218	229	197	224	228	204	212	240	271
POROUS2^4	333	328	343	322	341	*	352	323	303	367	334	*	349	335	324	467
POWELLBS	96	90	86	87	99	99	99	99	99	99	99	99	99	99	99	1314
POWELLSQ	13	13	13	13	12	13	14	17	18	17	16	13	13	13	13	16
QR3D	339	236	157	165	173	180	164	160	153	161	161	168	168	168	168	205
QR3D^2	1538	1344	1069	1016	942	969	859	919	946	968	957	950	954	954	954	947
QR3DBD	91	80	68	63	64	63	64	69	70	63	63	64	64	64	64	122
QR3DBD^2	868	954	713	697	769	733	640	686	670	737	716	667	604	593	593	728
RECIPE	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	15
SINVALNE	23	22	21	21	23	25	37	23	24	23	23	21	23	23	23	26
SPMSQRT	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	29
TRIGGER	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	10
WOODSNE	23	26	27	28	28	28	28	28	28	28	28	28	28	28	28	41
YATP1SQ	41	46	43	43	46	45	52	43	43	43	46	49	47	41	43	39
YATP1SQ^2	23	23	23	23	24	25	22	23	23	23	23	26	23	23	23	21
YATP1SQ^3	22	22	22	22	23	22	19	22	22	22	22	23	22	22	22	20
YATP2SQ	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	14
YATP2SQ^2	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	14
YATP2SQ^3	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	14
YFITNE	44	44	44	44	44	44	44	44	44	44	44	45	40	41	43	453
ZANGWIL3	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	23

Table 5.6: Results for ACO-LS: parametric study for the interpolation rule (Algorithm 3.1) and g-rule (3.5) (last column).