Working Paper Series, N. 7, June 2011

A Newton's Method for Benchmarking Time Series according to a Growth Rates Preservation Principle

Tommaso Di Fonzo
Department of Statistical Sciences
University of Padua
Italy

Marco Marini
Statistics Department
International Monetary Fund
USA

Abstract: We present a new technique for temporally benchmarking a time series according to the Growth Rates Preservation (*GRP*) principle by Causey and Trager (1981). This procedure basically looks for the solution to a non linear program, according to which $f(\mathbf{x})$, a smooth, non-convex function of the unknown values of the target time series $\{x_t\}$, $t = 1, \dots, n$, has to be minimized subject to linear equality constraints which link the more frequent series $\{x_t\}$ to a given, less frequent benchmark series $\{b_T\}$, $T = 1, \dots, m$. We develop a Newton's method with Hessian modification applied to a suitably reduced-unconstrained problem. This method exploits the analytic Hessian of the *GRP* objective function, making full use of all the derivative information at disposal. We show that the proposed technique is easy to implement, computationally robust and efficient, all features which make it a plausible competitor of other benchmarking procedures (Denton, 1971; Dagum and Cholette, 2006) also in a data-production process involving a considerable amount of series.

Keywords: Benchmarking, Movement preservation, Linearly equality constrained non-linear optimization, Newton's method.
JEL classification codes: C22, C61, C82.

Contents

Department of Statistical Sciences
Via Cesare Battisti, 241
35121 Padova
Italy

tel: +39 049 8274168
fax: +39 049 8274170
http://www.stat.unipd.it

Corresponding author:
Tommaso Di Fonzo
tel: +39 049 827 4158
difonzo@stat.unipd.it
http://www.stat.unipd.it/~difonzo

A Newton's Method for Benchmarking Time Series according to a Growth Rates Preservation Principle

Tommaso Di Fonzo
Department of Statistical Sciences
University of Padua
Italy

Marco Marini
Statistics Department
International Monetary Fund
USA

Abstract: We present a new technique for temporally benchmarking a time series according to the Growth Rates Preservation (*GRP*) principle by Causey and Trager (1981). This procedure basically looks for the solution to a non linear program, according to which $f(\mathbf{x})$, a smooth, non-convex function of the unknown values of the target time series $\{x_t\}$, $t = 1, \ldots, n$, has to be minimized subject to linear equality constraints which link the more frequent series $\{x_t\}$ to a given, less frequent benchmark series $\{b_T\}$, $T = 1, \ldots, m$. We develop a Newton's method with Hessian modification applied to a suitably reduced-unconstrained problem. This method exploits the analytic Hessian of the *GRP* objective function, making full use of all the derivative information at disposal. We show that the proposed technique is easy to implement, computationally robust and efficient, all features which make it a plausible competitor of other benchmarking procedures (Denton, 1971; Dagum and Cholette, 2006) also in a data-production process involving a considerable amount of series.

Keywords: Benchmarking, Movement preservation, Linearly equality constrained nonlinear optimization, Newton's method.
JEL classification codes: C22, C61, C82.

1 Introduction

The *Growth Rates Preservation* (*GRP*) benchmarking procedure by Causey and Trager (1981; see also Trager, 1982, and Bozik and Otto, 1988) is based on a *movement preservation principle*, according to which the sum of squared differences between the growth rates of the target and of the preliminary series is minimized[1].

This benchmarking procedure basically looks for a solution to a constrained Non

[1] Bloem *et al.* (2001, p. 100) claim that this is an "ideal" movement preservation principle, "formulated as an explicit preservation of the period-to-period rate of change" of the preliminary series.

Linear Program (NLP), according to which $f(\mathbf{x})$, a smooth, non-convex function of the n unknown items of vector $\mathbf{x}$, has to be minimized subject to a set of m linear equality constraints, $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a known, full row rank $(m \times n)$ matrix, $m < n$, and $\mathbf{b}$ is a known $(m \times 1)$ vector containing the benchmarks.

Both the original algorithm by Causey and Trager (1981) and a recent proposal by Brown (2010) are first-order (i.e., gradient-based) feasible directions methods, which use the Steepest Descent (SD) and the non-linear Conjugate Gradient (CG) algorithms, respectively, to solve the above NLP problem. However, using only first-derivatives information may result in poorly efficient procedures, characterized by slow convergence and possible troubles in finding actual minima of the objective function.

Still remaining at first-order techniques, more performing unconstrained Quasi-Newton (QN) optimization procedures may be considered, which exploit approximate rather than exact second derivatives, provided the original constrained problem be transformed into an unconstrained one. In addition, improvements in both efficiency and robustness may be obtained by considering the true Hessian matrix of the objective function.

In this paper, (i) we present the explicit expression of the Hessian matrix of the GRP objective function, (ii) show how the original constrained benchmarking problem can be transformed in an equivalent unconstrained non-linear problem, (iii) propose a Newton's method with Hessian modification (MN) to calculate GRP benchmarked estimates, and (iv) compare the performance of MN with gradient-based procedures (SD, CG, QN), in order to show the effectiveness of the proposed benchmarking procedure in terms of both computational efforts and quality of the results.

The paper is organized as follows. In section 2 the GRP benchmarking procedure is described, and the way it takes into account a movement preservation principle is discussed, as compared to the classical benchmarking procedure by Denton (1971), modified by Cholette (1984). The benchmarked estimates through this procedure, which is based on a constrained quadratic minimization problem and can be expressed in closed form, are generally considered as a good approximation of the GRP benchmarked estimates (this issue is discussed by Di Fonzo and Marini, 2010). In section 3 analytic expressions of gradient vector and Hessian matrix of the GRP criterion are presented, and by exploiting the Hessian we check the non-convexity of the objective function. While the gradient vector can be deduced by Causey and Trager (1981, see also Fagan, 1995), as far as we know the result concerning the Hessian matrix is new. In section 4 it is shown how a linear equality constrained problem can be transformed in an unconstrained problem with a reduced number of variables. This permits the user to exploit unconstrained optimization techniques. In addition, numerical results to efficiently transform and reduce the problem are presented. Line-search algorithms for unconstrained minimization are reviewed in section 5. The focus is on Newton-type methods as long as on classical first-order algorithms, namely steepest descent and nonlinear conjugate gradient. Feasible directions algorithms are considered in section 6, where the 'projected versions' of the steepest descent and of the nonlinear conjugate gradient algorithms are described. In order to analyze the distinctive features of the considered procedures, in section

7 are presented applications to the artificial series used by Denton (1971), and to several real-life series, namely 61 quarterly series from the EU Quarterly Sector Accounts (EUQSA), and 236 monthly series from the Canadian Monthly Retail Trade Survey (MRTS). Section 8 presents some final remarks and conclusions, and draws future research lines.

2 Growth Rates Preservation and Temporal Benchmarking

Let b_T, $T = 1, \dots, m$, and p_t, $t = 1, \dots, n$, be, respectively, the temporal benchmarks and the high–frequency preliminary values of an unknown target variable x_t. Let s be the aggregation order (e.g., $s = 4$ for quarterly-to-annual aggregation, $s = 12$ for monthly-to-annual aggregation, $s = 3$ for monthly-to-quarterly aggregation), and let $\mathbf{A}$ be a $(m \times n)$ temporal aggregation matrix, converting n high–frequency values in m low-frequency ones (we assume $n = s \cdot m$). If we denote with $\mathbf{x}$ the $(n \times 1)$ vector of high–frequency values, and with $\mathbf{b}$ the $(m \times 1)$ vector of low–frequency values, the aggregation constraints can be expressed as $\mathbf{Ax} = \mathbf{b}$.

Depending on the nature of the involved variables (e.g., flows, averages, stocks), the temporal aggregation matrix $\mathbf{A}$ usually can be written as

$$\mathbf{A} = \mathbf{I}_m \otimes \mathbf{a}^T, \tag{1}$$

where the $(s \times 1)$ vector $\mathbf{a}$ may assume one of the following forms:

1. flows: $\mathbf{a} = \mathbf{1}_s = \begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix}^T$,
2. averages: $\mathbf{a} = \frac{1}{s}\mathbf{1_s}$,
3. stocks (end-of-the-period): $\mathbf{a} = \begin{pmatrix} 0 & 0 & \dots & 1 \end{pmatrix}^T$,
4. stocks (beginning-of-the-period): $\mathbf{a} = \begin{pmatrix} 1 & 0 & \dots & 0 \end{pmatrix}^T$.

Denoting by $\mathbf{p}$ the $(n \times 1)$ vector of preliminary values ($\mathbf{Ap} \neq \mathbf{b}$), we look for a vector of benchmarked estimates $\mathbf{x}^*$ which should be 'as close as possible' to the preliminary values, and such that $\mathbf{Ax}^* = \mathbf{b}$ (e.g., for flows variables, $\sum\limits_{t \in T} x_t^* = b_T$, $T = 1, \dots, m$).

To this end, some characteristics of the original series p_t should be kept into consideration. For example, in an economic time series framework, the preservation of the temporal dynamics (however defined) of the preliminary series is often a major interest of the practitioner. For flows series, Causey and Trager (1981; see also Monsour and Trager, 1979, and Trager, 1982) consider a criterion to be minimized explicitly related to the growth rate, which is a natural measure of the movement of a time series:

$$f(\mathbf{x}) = \sum_{t=2}^{n} \left(\frac{x_t}{x_{t-1}} - \frac{p_t}{p_{t-1}} \right)^2, \tag{2}$$

and look for values x_t^*, $t = 1, \dots, n$, which minimize the criterion (2) subject to the aggregation constraints $\sum\limits_{t \in T} x_t = b_T$, $T = 1, \dots, m$. In other words, the benchmarked series is estimated in such a way that its temporal dynamics, as expressed by the growth rates $\frac{x_t^*}{x_{t-1}^*}$, $t = 2, \ldots, n$, be 'as close as possible' to the temporal dynamics of the preliminary series, where the 'distance' from the preliminary growth rates $\frac{p_t}{p_{t-1}}$ is given by the sum of the squared differences.

In this paper we consider a more general formulation of the *GRP* benchmarking problem, valid not only for flows variables linked by a simple summation, that is:

$$\min_{\mathbf{x}} f(\mathbf{x}) \qquad \text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b}, \tag{3}$$

where $\mathbf{A}$ is the temporal aggregation matrix (1). The criterion (2) is clearly a non-linear and, as we shall see in the following, non-convex function. The constrained minimization problem (3) has not linear first–order conditions for a stationary point, and thus it is not possible to find an explicit, analytic expression for the solution. On the other hand, provided that both p_t and x_t, $t = 1, \ldots, n-1$, be different from zero, $f(\mathbf{x})$ is a twice continuously differentiable function, making it possible the use of several iterative minimization algorithms (Nocedal and Wright, 2006).

2.1 Modified Denton PFD

Denton (1971) proposed a benchmarking procedure grounded on the *Proportionate First Differences* (PFD) between the target and the original series. Cholette (1984) slightly modified the result of Denton, in order to correctly deal with the starting conditions of the problem. The PFD benchmarked estimates are thus obtained as the solution to the constrained quadratic minimization problem

$$\min_{x_t} \sum_{t=2}^{n} \left(\frac{x_t}{p_t} - \frac{x_{t-1}}{p_{t-1}} \right)^2 \qquad \text{subject to} \quad \mathbf{A}\mathbf{x} = \mathbf{b}. \tag{4}$$

In matrix notation, the PFD benchmarked series is contained in the $(n \times 1)$ vector $\mathbf{x}^{PFD}$ solution to the linear system (Di Fonzo and Marini, 2010)

$$\begin{bmatrix} \mathbf{M} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x}^{PFD} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \end{bmatrix}, \tag{5}$$

where λ is a $(n \times 1)$ vector of Lagrange multipliers, $\mathbf{M} = \mathbf{P}^{-1}\Delta_n^T\Delta_n\mathbf{P}^{-1}$, $\mathbf{P} = \text{diag}(\mathbf{p})$, and Δ_n is the $((n-1) \times n)$ first differences matrix:

$$\begin{pmatrix} -1 & 1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{pmatrix}.$$

Notice that $\Delta_n^T\Delta_n$ has rank $n-1$ (Cohen *et al.*, 1971, p. 122), so $\mathbf{M}$ is singular. However, given that matrix $\mathbf{A}$ has full row rank, and provided no preliminary value is equal to zero[2], the coefficient matrix of system (5) has full rank (Di Fonzo and Marini, 2010).

Causey and Trager (1981) use $\mathbf{x}^{PFD}$ as starting values of the iterative algorithm developed to solve the NLP problem (3). This basically depends on two facts:

1. the optimization procedure starts at a feasible point, as $\mathbf{x}^{PFD}$ clearly is, and at each iteration moves to another feasible point;

2. in the literature (Cholette, 1984, Bloem *et al.*, 2001, Dagum and Cholette, 2006) it is often claimed that the PFD procedure produces results very close to the GRP benchmarking, and thus $\mathbf{x}^{PFD}$, which is considered as a 'good' approximation to the GRP estimates, is a natural candidate to be used as starting point.

Di Fonzo and Marini (2010) discuss this latter issue, showing that PFD and GRP benchmarked estimates are close when the variability of the preliminary series and/or its bias are low with respect to the target variable. When this is not the case (e.g. preliminary series with large growth rates and/or bias), the quality of the approximation worsens. In addition, GRP benchmarking almost always results in a better movement preservation as compared to Denton PFD[3].

3 Gradient Vector and Hessian Matrix of the GRP criterion

Computational studies on a number of test problems of varying complexity demonstrate that the calculation and treatment of the Hessian matrix are fundamental to the observed performance of NLP optimization algorithms. Second order information though costly, because its calculation is often cumbersome, leads to quadratic convergence to a (possibly local) optimal solution, whereas gradient information leads to convergence with a linear convergence rate. There are various alternatives for exploiting second order information about the function of interest. Basically, the Hessian matrix can be calculated analitically or approximated by using finite differences techniques.

In this section we present the analytical expression of the gradient vector of the GRP criterion (2), which has been originally derived by Causey and Trager (1981, see also Fagan, 1995). Then we calculate the analytical expression of the Hessian matrix of the criterion, which can be exploited by Newton-type NLP optimization procedures.

The gradient vector of function (2) is the $(n \times 1)$ vector

$$\nabla f(\mathbf{x}) = \mathbf{g}(\mathbf{x}) = \{g_t\}_{t=1}^{n},$$

[2]When some p_t is null, a standard practice in the benchmarking literature (see, for example, Cholette and Chhab, 1991, p. 413) consists in setting the originally null preliminary data at a very small value, e.g. $p_t = 0.001$.

[3]Empirical comparisons between the Cholette-Dagum regression based benchmarking approach, which can be seen as a generalization of the PFD procedure (Dagum and Cholette, 2006), and the Causey and Trager approach, are shown in Harvill Hood (2005) and Titova *et al.* (2010).

where

$$\begin{aligned}
g_1 &= -2\frac{x_2}{x_1^2}\left(\frac{x_2}{x_1} - \frac{p_2}{p_1}\right) \\
g_t &= \frac{2}{x_{t-1}}\left(\frac{x_t}{x_{t-1}} - \frac{p_t}{p_{t-1}}\right) - 2\frac{x_{t+1}}{x_t^2}\left(\frac{x_{t+1}}{x_t} - \frac{p_{t+1}}{p_t}\right) \quad t = 2, \dots, n-1 \\
g_n &= \frac{2}{x_{n-1}}\left(\frac{x_n}{x_{n-1}} - \frac{p_n}{p_{n-1}}\right).
\end{aligned}$$

Let us denote the elements of the Hessian matrix, $\nabla^2 f(\mathbf{x}) = \mathbf{H}(\mathbf{x})$, as

$$h_{ij} = \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} = \frac{\partial g_i}{\partial x_j}, \quad i, j = 1, \dots, n.$$

Notice that the Hessian matrix is both symmetric and tri-diagonal, that is its non-zero items are $h_{t,t}$, $t = 1, \dots, n$, $h_{t-1,t}$, $t = 2, \dots, n$, and $h_{t+1,t}$, $t = 1, \dots, n-1$. After some calculations, we find:

$$\begin{aligned}
h_{11} &= 2\frac{x_2}{x_1^3}\left(3\frac{x_2}{x_1} - 2\frac{p_2}{p_1}\right) \\
h_{t,t} &= \frac{2}{x_{t-1}^2} + 2\frac{x_{t+1}}{x_t^3}\left(3\frac{x_{t+1}}{x_t} - 2\frac{p_{t+1}}{p_t}\right) \quad t = 2, \dots, n-1 \\
h_{n,n} &= \frac{2}{x_{n-1}^2} \\
h_{ij} &= -\frac{2}{x_i^2}\left(2\frac{x_j}{x_i} - \frac{p_j}{p_i}\right) \quad i = j+1, j = 1, \dots, n-1 \vee i = j-1, j = 2, \dots, n.
\end{aligned}$$

For example, assuming $n = 4$ we have:

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix}
-2\frac{x_2}{x_1^2}\left(\frac{x_2}{x_1} - \frac{p_2}{p_1}\right) \\
\frac{2}{x_1}\left(\frac{x_2}{x_1} - \frac{p_2}{p_1}\right) - 2\frac{x_3}{x_2^2}\left(\frac{x_3}{x_2} - \frac{p_3}{p_2}\right) \\
\frac{2}{x_2}\left(\frac{x_3}{x_2} - \frac{p_3}{p_2}\right) - 2\frac{x_4}{x_3^2}\left(\frac{x_4}{x_3} - \frac{p_4}{p_3}\right) \\
\frac{2}{x_3}\left(\frac{x_4}{x_3} - \frac{p_4}{p_3}\right)
\end{bmatrix},$$

while the lower triangle of the Hessian matrix is:

$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} 2\frac{x_2}{x_1^3}\left(3\frac{x_2}{x_1}-2\frac{p_2}{p_1}\right) & & & \\ -\frac{2}{x_1^2}\left(2\frac{x_2}{x_1}-\frac{p_2}{p_1}\right) & \frac{2}{x_1^2}+2\frac{x_3}{x_2^3}\left(3\frac{x_3}{x_2}-2\frac{p_3}{p_2}\right) & & \\ 0 & -\frac{2}{x_2^2}\left(2\frac{x_3}{x_2}-\frac{p_3}{p_2}\right) & \frac{2}{x_2^2}+2\frac{x_4}{x_3^3}\left(3\frac{x_4}{x_3}-2\frac{p_4}{p_3}\right) & \\ 0 & 0 & -\frac{2}{x_3^2}\left(2\frac{x_4}{x_3}-\frac{p_4}{p_3}\right) & \frac{2}{x_3^2} \end{bmatrix} . .$$

This last formula can be used in a simple numerical example to check that the *GRP* criterion (2) is non-convex. Consider the (4×1) vectors $\mathbf{x} = \begin{pmatrix} 1 & 2 & 3 & 4 \end{pmatrix}^T$ and $\mathbf{p} = \begin{pmatrix} 8 & 5 & 6 & 7 \end{pmatrix}^T$, respectively. Simple calculations give

$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} 19 & -6.75 & 0 & 0 \\ -6.75 & 3.575 & -0.9 & 0 \\ 0 & -0.9 & 0.9938 & -0.3333 \\ 0 & 0 & -0.3333 & 0.2222 \end{bmatrix}, \qquad \text{and} \qquad |\mathbf{H}(\mathbf{x})| = -0.9660.$$

Because its Hessian has a negative determinant, the *GRP* criterion (2) is non-convex. In addition, the upper left (3×3) matrix

$$\begin{bmatrix} 19 & -6.75 & 0 \\ -6.75 & 3.575 & -0.9 \\ 0 & -0.9 & 0.9938 \end{bmatrix}$$

has a positive determinant, equal to 6.8345, thus $\mathbf{H}(\mathbf{x})$ is an indefinite (neither positive nor negative definite) matrix. Thus, unlike the convex case, function $f(\mathbf{x})$ is not guaranteed to have a unique, global minimum. This is an important feature of the minimization problem (3), to be taken into considerations when, in order to solve it, we use minimization procedures which are generally local.

4 From a constrained to an unconstrained minimization problem

One possible approach to solving the linear equality constrained minimization problem (3) is to *eliminate* the constraints, and to solve the resulting problem with algorithms for unconstrained minimization.

In this section we show that the equality constrained problem (3) can be transformed into an equivalent unconstrained problem, after which an unconstrained minimization method can be used to solve it.

As we shall see, the process of eliminating the equality constraints (and reconstructing the solution of the original problem from the solution of the transformed problem) involves standard linear algebra operations. Moreover, for most temporal benchmarking problems, the pattern of the constraint matrix $\mathbf{A}$ (see section

2), makes it possible to develop a simple and numerical stable approach to the elimination of the variables, which requires neither complex nor time consuming computation programs. Thus, unlike the cases where it is better to retain the equality constraints, since eliminating them can make the problem harder to understand and analyze, or ruin the efficiency of an algorithm that solves it[4], for our problem the proposed approach turns out to be simple, cheap and effective in computational terms.

4.1 Eliminating the Linear Equality Constraints

We start by assuming that the temporal aggregation matrix $\mathbf{A}$ defining the constraints in problem (3) has full row rank $m < n$, which means that the constraints are independent (i.e., not redundant). It should be noted that this property is always satisfied by matrices $\mathbf{A}$ as defined in (1).

As many variables (m out of n) as independent constraints can be eliminated by considering a re-parametrization of the affine feasible set defined by the constraints:

$$\mathcal{F} = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{b}\}. \tag{6}$$

Since any n-dimensional vector can be written uniquely as the sum of a range-space and a null space-component, we can write (Nocedal and Wright, 2006):

$$\mathbf{x} = \mathbf{y} + \mathbf{z}, \tag{7}$$

with $\mathbf{y} \in \mathcal{R}(\mathbf{A}^T)$ and $\mathbf{z} \in \mathcal{N}(\mathbf{A})$, where

$$\mathcal{R}(\mathbf{A}^T) = \{\mathbf{y} \in \mathbb{R}^n : \mathbf{y} = \mathbf{A}^T\lambda \quad \text{for some} \quad \lambda \in \mathbb{R}^m\}$$

and

$$\mathcal{N}(\mathbf{A}) = \{\mathbf{z} \in \mathbb{R}^n : \mathbf{A}\mathbf{z} = \mathbf{0}\},$$

denote the range-space of matrix $\mathbf{A}^T$ and the null-space of matrix $\mathbf{A}$, respectively. If we denote by $\mathbf{Y}$ a $(n \times m)$ basis matrix for $\mathcal{R}(\mathbf{A}^T)$, and by $\mathbf{Z}$ a $(n \times (n-m))$ basis matrix for $\mathcal{N}(\mathbf{A})$, expression (7) can be written as

$$\mathbf{x} = \mathbf{Y}\mathbf{x}_Y + \mathbf{Z}\mathbf{x}_Z, \tag{8}$$

where $\mathbf{x}_Y$ and $\mathbf{x}_Z$ are $(m \times 1)$ and $((n-m) \times 1)$ vectors, respectively. Notice that matrix $\mathbf{Z}$ is such that $\mathbf{A}\mathbf{Z} = \mathbf{0}$.

For any feasible point $\mathbf{x}$, the pre-multiplication by $\mathbf{A}$ of expression (8) gives $\mathbf{A}\mathbf{Y}\mathbf{x}_Y = \mathbf{b}$, which means that, for any choice of matrix $\mathbf{Y}$, $\mathbf{x}_Y$ is uniquely determined as

$$\mathbf{x}_Y = (\mathbf{A}\mathbf{Y})^{-1}\mathbf{b}. \tag{9}$$

Ultimately, it follows that any feasible $\mathbf{x}$ can be written as

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{Z}\mathbf{x}_Z, \tag{10}$$

where $\bar{\mathbf{x}} = \mathbf{Y}(\mathbf{A}\mathbf{Y})^{-1}\mathbf{b}$ is a feasible $(n \times 1)$ vector (i.e, $\mathbf{A}\bar{\mathbf{x}} = \mathbf{b}$). In other words, the n constrained variables in $\mathbf{x}$ have been transformed into the $n-m$ unconstrained variables in $\mathbf{x}_Z$.

[4]This could happen, for example, when $\mathbf{x}$ has very large dimension, and eliminating the equality constraints would destroy sparsity of some other useful structure of the problem (Boyd and Vandenberghe, 2004, p. 143).

4.2 Generating an Elimination Matrix by QR Factorization

Matrix $\mathbf{Z}$ in (10) is called *elimination matrix* (Boyd and Vandenberghe, 2004, p. 524). It is not uniquely defined, and its choice should be made carefully, because it could have a deep impact on the performance of the optimization procedure (Nocedal and Wright, 2006, pp. 430-431).

We propose to compute a null-space matrix for $\mathbf{A}$ by a QR factorization of matrix $\mathbf{A}^T$, which permits to define an orthonormal basis of $\mathbb{R}^n$ "ideal from the point of view of numerical stability" (Nocedal and Wright, 2006, p. 433)[5]. We discuss how this result can be obtained, and show that for $\mathbf{A}$ as defined by (1), this factorization involves simple and readily available matrices, which need not to be calculated by any numerical procedure[6].

Let us consider the orthonormal QR factorization (Nocedal and Wright, 2006):

$$\mathbf{A}^T\mathbf{\Pi} = \mathbf{QR}, \tag{11}$$

where $\mathbf{\Pi}$ is an $(m \times m)$ permutation matrix, $\mathbf{Q}$ is an $(n \times n)$ orthogonal matrix (i.e., $\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T = \mathbf{I}_n$), and $\mathbf{R}$ is an upper triangular $(n \times m)$ matrix, respectively. Now, let us partition matrices $\mathbf{Q}$ and $\mathbf{R}$ as:

$$\mathbf{Q} = [\mathbf{Q}_1 \quad \mathbf{Q}_2] \qquad \mathbf{R} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{0} \end{bmatrix}, \tag{12}$$

where $\mathbf{Q}_1$, $\mathbf{Q}_2$ and $\mathbf{R}_1$ are $(n \times m)$, $(n \times (n-m))$ and $(m \times m)$ matrices, respectively. Since both $\mathbf{\Pi}$ and $\mathbf{Q}$ are orthogonal matrices, it follows that $\mathbf{AQ} = \mathbf{\Pi R}^T$, or

$$\mathbf{AQ}_1 = \mathbf{\Pi R}_1^T \quad \text{and} \quad \mathbf{AQ}_2 = \mathbf{0}.$$

Thus, $\mathbf{Y} = \mathbf{Q}_1$ is a basis for the range-space of $\mathbf{A}^T$, and $\mathbf{Z} = \mathbf{Q}_2$ is a basis for the null-space of $\mathbf{A}$, with $\mathbf{Y}^T\mathbf{Y} = \mathbf{I}_m$ and $\mathbf{Z}^T\mathbf{Z} = \mathbf{I}_{n-m}$.

The pattern of the constraint matrix, $\mathbf{A} = \mathbf{I}_m \otimes \mathbf{a}^T$ (see section 2), makes it possible to compute a QR factorization of $\mathbf{A}^T$ involving matrices formulated in compact form and once for all, ready to be implemented in a program code without any further (possibly complex and time consuming) elaboration. A simple, and effective choice for the matrices involved in (11)-(12) is the following:

$$\mathbf{\Pi} = \mathbf{I}_m, \quad \mathbf{Q}_1 = \mathbf{I}_m \otimes \kappa\mathbf{1}_s, \quad \mathbf{Q}_2 = \mathbf{I}_m \otimes \mathbf{K}, \quad \mathbf{R}_1 = \kappa^{-1}\mathbf{I}_m,$$

where the scalar quantity κ and the $(s \times (s-1))$ matrix $\mathbf{K}$ depend on the type of aggregation:

Flows	Average	Stocks (End/Begin-of-period)
$\kappa = -\frac{1}{\sqrt{s}}$	$\kappa = -\frac{s}{\sqrt{s}}$	$\kappa = 1$

[5]Other methods for generating null-space matrices can be found in Griva *et al.* (2009, pp. 86-91).

[6]This fact prevents problems deriving from possible large dimensions of the series to be benchmarked, since in practice no computing effort is required to perform the QR factorization.

Flows and Average	E-o-p stocks	B-o-p stocks
$\mathbf{K} = \begin{bmatrix} \sqrt{\frac{s-1}{s}} & 0 & \cdots & 0 \\ -\sqrt{\frac{1}{s(s-1)}} & \sqrt{\frac{s-2}{s-1}} & \cdots & 0 \\ -\sqrt{\frac{1}{s(s-1)}} & -\sqrt{\frac{1}{(s-1)(s-2)}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -\sqrt{\frac{1}{s(s-1)}} & -\sqrt{\frac{1}{(s-1)(s-2)}} & \cdots & -\sqrt{\frac{1}{2}} \\ -\sqrt{\frac{1}{s(s-1)}} & -\sqrt{\frac{1}{(s-1)(s-2)}} & \cdots & \sqrt{\frac{1}{2}} \end{bmatrix}$	$\mathbf{K} = \begin{bmatrix} \mathbf{I}_{s-1} \\ \mathbf{0}^T \end{bmatrix}$	$\mathbf{K} = \begin{bmatrix} \mathbf{0}^T \\ \mathbf{I}_{s-1} \end{bmatrix}$

Notice that in any case it is $\mathbf{K}^T\mathbf{K} = \mathbf{I}_{s-1}$, while for both types of stocks we have $\mathbf{Q}_1 = \mathbf{A}^T$.

It can be easily shown that, according to this choice of $\mathbf{Q}_1$ and $\mathbf{Q}_2$, matrix $\mathbf{Q}$ is orthonormal, as one can see by checking the following relationships:

$$\begin{aligned} \mathbf{Q}_1^T\mathbf{Q}_1 &= \mathbf{I}_m \\ \mathbf{Q}_2^T\mathbf{Q}_2 &= \mathbf{I}_{n-m} \\ \mathbf{Q}_1^T\mathbf{Q}_2 &= \mathbf{0} \\ \mathbf{Q}_2^T\mathbf{Q}_1 &= \mathbf{0} \\ \mathbf{Q}\mathbf{Q}^T &= \mathbf{Q}^T\mathbf{Q} = \mathbf{I}_n, \end{aligned}$$

where the zero matrices have dimensions $(m \times m(s-1))$ and $(m(s-1) \times m)$, respectively.

For example, for flows variables and $s = 4$, as for the quarterly-to-annual benchmarking, it is

$$\kappa = -0.5 \qquad \text{and} \qquad \mathbf{K} = \begin{bmatrix} 0.8660 & 0 & 0 \\ -0.2887 & 0.8165 & 0 \\ -0.2887 & -0.4082 & -0.7071 \\ -0.2887 & -0.4082 & 0.7071 \end{bmatrix},$$

while for $s = 12$, as for the monthly-to-annual benchmarking, it is $\kappa = -0.2887$ and

$$\mathbf{K} =$$

$$\begin{bmatrix} 0.9574 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.0870 & 0.9535 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.0870 & -0.0953 & 0.9487 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.0870 & -0.0953 & -0.1054 & 0.9428 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.0870 & -0.0953 & -0.1054 & -0.1179 & 0.9354 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.0870 & -0.0953 & -0.1054 & -0.1179 & -0.1336 & 0.9258 & 0 & 0 & 0 & 0 & 0 \\ -0.0870 & -0.0953 & -0.1054 & -0.1179 & -0.1336 & -0.1543 & 0.9129 & 0 & 0 & 0 & 0 \\ -0.0870 & -0.0953 & -0.1054 & -0.1179 & -0.1336 & -0.1543 & -0.1826 & 0.8944 & 0 & 0 & 0 \\ -0.0870 & -0.0953 & -0.1054 & -0.1179 & -0.1336 & -0.1543 & -0.1826 & -0.2236 & 0.8660 & 0 & 0 \\ -0.0870 & -0.0953 & -0.1054 & -0.1179 & -0.1336 & -0.1543 & -0.1826 & -0.2236 & -0.2887 & 0.8165 & 0 \\ -0.0870 & -0.0953 & -0.1054 & -0.1179 & -0.1336 & -0.1543 & -0.1826 & -0.2236 & -0.2887 & -0.4082 & -0.7071 \\ -0.0870 & -0.0953 & -0.1054 & -0.1179 & -0.1336 & -0.1543 & -0.1826 & -0.2236 & -0.2887 & -0.4082 & 0.7071 \end{bmatrix}.$$

4.3 The Reduced Unconstrained Minimization Problem

The optimization problem with equality constraints (3) can be transformed into an equivalent unconstrained problem by incorporating the constraints into the objective function. For, we need only to consider the restricted variables $\mathbf{x} = \bar{\mathbf{x}} + \mathbf{Z}\mathbf{x}_Z$, and the function

$$\tilde{f}(\mathbf{x}_Z) = f(\bar{\mathbf{x}} + \mathbf{Z}\mathbf{x}_Z). \tag{13}$$

The argument of this function, $\mathbf{x}_Z$, is a vector with $n-m$ elements, instead of n as in the original function f. The unconstrained minimum of $\tilde{f}$, say $\mathbf{x}_Z^*$, is however the solution to the original constrained problem.

Since $\mathbf{Z}^T\mathbf{Z} = \mathbf{I}_{n-m}$ and $\mathbf{Z}^T\mathbf{Y} = \mathbf{Q}_2^T\mathbf{Q}_1 = \mathbf{0}$, $\mathbf{x}_Z$ can be written as $\mathbf{x}_Z = \mathbf{Z}^T\mathbf{x}$, and thus the Jacobian of the transformation is

$$\frac{\partial \mathbf{x}_Z}{\partial \mathbf{x}^T} = \mathbf{Z}^T.$$

If we assume differentiability of f, gradient and Hessian of the reduced function can be expressed in terms of the original function:

$$\begin{array}{rcl} \nabla\tilde{f}(\mathbf{x}_Z) & = & \mathbf{Z}^T\left[\nabla f(\mathbf{x})\right] \\ \nabla^2\tilde{f}(\mathbf{x}_Z) & = & \mathbf{Z}^T\left[\nabla^2 f(\mathbf{x})\right]\mathbf{Z} \end{array}. \tag{14}$$

The relationship of the properties of stationary points to the derivatives (14) are the conditions that determine a minimum of the reduced objective function (13). Thus $\mathbf{x}^* = \bar{\mathbf{x}} + \mathbf{Z}\mathbf{x}_Z^*$ is a minimum if and only if

- $\mathbf{Z}^T\left[\nabla f(\mathbf{x}^*)\right] = \mathbf{0}$,
- $\mathbf{Z}^T\left[\nabla^2 f(\mathbf{x}^*)\right]\mathbf{Z}$ is positive definite, and
- $\mathbf{A}\mathbf{x}^* = \mathbf{b}$.

The first two relationships are standard conditions for a minimum, while the third condition derives from the constrained nature of the problem. Considered together, these relationships provide the basis for the solution to the original optimization problem (3).

5 Line-Search Algorithms for Unconstrained Minimization

A general algorithm for solving the unconstrained minimization problem

$$\min_{\mathbf{x}_Z} \tilde{f}(\mathbf{x}_Z) \tag{15}$$

can be stated as follows

1. Specify some initial guess of the solution: $\mathbf{x}_{Z,0} = \mathbf{Z}^T\mathbf{x}_0$;[7]
2. For $k = 0, 1, \ldots$, if $\mathbf{x}_{Z,k}$ is optimal, then stop. Otherwise:

[7] In agreement with Causey and Trager (1981), we assume $\mathbf{x}_0 = \mathbf{x}^{PFD}$, solution to system (5).

(a) determine a *search direction* $\mathbf{d}_k$;

(b) determine a *step length* α_k that leads to an improved estimate of the solution

$$\mathbf{x}_{Z,k+1} = \mathbf{x}_{Z,k} + \alpha_k \mathbf{d}_k. \tag{16}$$

Once the direction $\mathbf{d}_k$ has been computed, the step length α_k is found by solving some auxiliary one-dimensional problem (Nocedal and Wright, 2006). It is typically required that the search direction $\mathbf{d}_k$ be a *descent direction* for the function $\tilde{f}$ at the point $\mathbf{x}_{Z,k}$. This means that for "small" steps taken along $\mathbf{d}_k$ the function value is guaranteed to decrease:

$$\tilde{f}\left(\mathbf{x}_{Z,k} + \alpha \mathbf{d}_k\right) < \tilde{f}\left(\mathbf{x}_{Z,k}\right) \qquad \text{for} \qquad 0 < \alpha \leq \epsilon$$

for some $\epsilon > 0$.

This algorithm with its three major steps - the optimality test, computation of $\mathbf{d}_k$, and computation of α_k through a line-search approach - has been the basis for many successful optimization algorithms, and has been used to develop many software packages for nonlinear optimization. However, it is not the only approach possible. Another effective nonlinear optimization approach is the *trust-region method* (Nocedal and Wright, 2006). In this paper we follow the line-search approach due to its effectiveness and simplicity, also in terms of software implementation.

5.1 Newton's Method with Hessian Modification

In its classical form, Newton's method basically consists in determining $\mathbf{d}_k$ in (16) as the solution to the *Newton equations*

$$\left[\nabla^2 \tilde{f}(\mathbf{x}_{Z,k})\right] \mathbf{d}_k = -\left[\nabla \tilde{f}\left(\mathbf{x}_{Z,k}\right)\right]. \tag{17}$$

Since it can fail or diverge, and even if it does converge, it might not converge to a minimum, Newton's method is rarely used in its classical form. Possible solutions to guarantee that the method will converge to a stationary point and possibly a local minimum, if one exists, is to use the Newton direction within the general recursion (16), and to consider a modification of the Hessian matrix in order to have a positive definite matrix in the Newton equations (17). Thus, a practical version of Newton's method, that is guaranteed to converge and does not assume that $\nabla^2 \tilde{f}\left(\mathbf{x}_{Z,k}\right)$ is positive definite for all values of k, can be summarized as follows.

1. Specify some initial guess of the solution, $\mathbf{x}_{Z,0}$, and specify a convergence tolerance ϵ.

2. For $k = 0, 1, \ldots$, if $\|\nabla \tilde{f}\left(\mathbf{x}_{Z,k}\right)\|_1 < \epsilon$, then stop. Otherwise:

   (a) Compute a modified factorization of the Hessian:

   $$\nabla^2 \tilde{f}\left(\mathbf{x}_{Z,k}\right) + \mathbf{E} = \mathbf{L}\mathbf{D}\mathbf{L}^T,$$

   where $\mathbf{L}$ and $\mathbf{D}$ are lower triangular and diagonal, respectively, $(n-m) \times (n-m)$ matrices. Then, solve

   $$\left(\mathbf{L}\mathbf{D}\mathbf{L}^T\right) \mathbf{d}_k = -\left[\nabla \tilde{f}\left(\mathbf{x}_{Z,k}\right)\right]$$

for the search direction $\mathbf{d}_k$. Notice that $\mathbf{E}$ will be zero if $\nabla^2 \tilde{f}\,(\mathbf{x}_{Z,k})$ is positive definite.

(b) Perform a line search to determine the new estimate of the solution (16).

A principal advantage of the Newton's method with Hessian modification is that it converges rapidly when the current estimate of the variables is close to the solution. Its main disadvantage is represented by possible high computational costs, since it requires the derivation, computation, and storage of the Hessian matrix, and the solution of a system of linear equations. This last task could give raise to high computational costs if the dimension of the problem $(n-m)$ is not small and/or the problem is not sparse.

However, for the problem in hand, in section 4 we have shown the analytical expressions and the patterns of gradient and Hessian matrix of the problem, so we can take advantage of sparsity, and greatly reduce the computational costs of Newton's method, making it an effective tool in practice.

5.2 Steepest Descent and Quasi-Newton Methods

Both steepest descent and quasi-Newton methods can be seen as compromises to Newton's method (Griva *et al.*, 2009), that reduce one or more of its costs. In exchange, these methods generally have slower convergence rates.

These methods can be interpreted as computing the search direction $\mathbf{d}_k$ by solving the linear system

$$\mathbf{B}_k \mathbf{d}_k = -\left[\nabla \tilde{f}\,(\mathbf{x}_{Z,k})\right], \tag{18}$$

where $\mathbf{B}_k$ is a positive-definite matrix. Since in the case of Newton's method, $\mathbf{B}_k = \nabla^2 \tilde{f}\,(\mathbf{x}_{Z,k})$, assuming that the Hessian matrix is positive definite, intuitively $\mathbf{B}_k$ should be some approximation to $\nabla^2 \tilde{f}(\mathbf{x}_{Z,k})$.

5.2.1 Steepest Descent Method

The steepest-descent method computes the search direction by assuming $\mathbf{B}_k = \mathbf{I}_{n-m}$, which gives the search direction $\mathbf{d}_k = -\left[\nabla \tilde{f}\,(\mathbf{x}_{Z,k})\right]$, and then uses a line search to determine the updated approximate solution $\mathbf{x}_{Z,k+1}$ according to (16).

This is an old, widely known and cheap method, whose performance is usually very low. It is much simpler than Newton's method because it does not require the computation of second derivatives, no system of linear equations must be solved to compute the search direction, and no matrix storage is needed. On the negative side, it has a slower convergence rate than Newton's method, and sometimes it converges so slowly that $\mathbf{x}_{Z,k+1} - \mathbf{x}_{Z,k}$ is below the precision of computer arithmetic and the method fails.

5.2.2 Quasi-Newton Methods

Quasi-Newton methods are currently among the most widely used Newton-type methods for nonlinear optimization problems of moderate size, where matrices can be stored. They are incorporated in many software libraries, and they are effective in solving a wide variety of small to mid-size problems, in particular when the Hessian is hard to compute.

There are many different quasi-Newton methods, but they are all based on approximating the Hessian $\nabla^2 \tilde{f}(\mathbf{x}_{Z,k})$ by another matrix $\mathbf{B}_k$ that is available at lower cost. Then the search direction is obtained by solving equation (18). If the matrix $\mathbf{B}_k$ is positive definite, then this is equivalent to minimizing the quadratic model

$$q(\mathbf{d}_k) = \tilde{f}(\mathbf{x}_{Z,k}) + \left[\nabla \tilde{f}(\mathbf{x}_{Z,k})\right]^T \mathbf{d}_k + \frac{1}{2}\mathbf{d}_k^T \mathbf{B}_k \mathbf{d}_k.$$

There are several advantages to this approach. First, an approximation $\mathbf{B}_k$ can be found using only first-derivative information. Second, the search direction can be computed using only $O(n^2)$ operations (*vs.* $O(n^3)$ for Newton's method in the nonsparse case). There are also disadvantages, but they are minor. Quasi-Newton methods do not converge quadratically, but they can converge superlinearly (Nocedal and Wright, 2006). At the precision of computer arithmetic, there is not much practical difference between these two rates of convergence. Also, quasi-Newton methods still require matrix storage, so they are not normally used to solve large problems[8].

The various quasi-Newton methods differ in the choice of $\mathbf{B}_k$. A variety of methods are obtained by imposing conditions on the approximation $\mathbf{B}_k$. These conditions are usually properties of the Hessian matrix that the approximation should share, like symmetry and positive definiteness. Due to its effectiveness, the most widely used expression for $\mathbf{B}_k$ is the update formula by Broyden, Fletcher, Goldfarb, and Shanno (*BFGS*):

$$\mathbf{B}_{k+1} = \mathbf{B}_k - \frac{(\mathbf{B}_k \mathbf{s}_k)(\mathbf{B}_k \mathbf{s}_k)^T}{\mathbf{s}_k^T \mathbf{B}_k \mathbf{s}_k} + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}, \tag{19}$$

where $\mathbf{s}_k = \mathbf{x}_{Z,k+1} - \mathbf{x}_{Z,k}$, $\mathbf{y}_k = \nabla \tilde{f}(\mathbf{x}_{Z,k+1}) - \nabla \tilde{f}(\mathbf{x}_{Z,k})$, and $\mathbf{B}_0 = \mathbf{I}_{n-m}$.

5.3 Nonlinear Conjugate Gradient

Conjugate gradient methods are generally considered as an excellent choice to solve a nonlinear unconstrained minimization problem, since they do not require the evaluation of the Hessian matrix neither the storage of an approximation of it. Nonlinear conjugate gradient algorithms are of a considerable interest from both theoretical and practical points of view, particularly for their convergence properties (Hager and Zhang, 2006), a very easy implementation effort in computer programs, and their efficiency in solving large-scale problems.

Starting from an initial guess $\mathbf{x}_{Z,0}$, the nonlinear conjugate gradient method generates a sequence $\mathbf{x}_{Z,k}$ according to the relationship (16), where the positive step size α_k is obtained by a line search, and the direction $\mathbf{d}_k$ is recursively defined by

$$\mathbf{d}_k = \begin{cases} -\tilde{\mathbf{g}}_0 & \text{for } k = 0 \\ -\tilde{\mathbf{g}}_k + \beta_k \mathbf{d}_{k-1} & \text{for } k \geq 1 \end{cases}, \tag{20}$$

[8]This drawback can be overcome by using a 'limited' version of the algorithm, like the *LBFGS* method (Nocedal and Wright, 2006).

where $\tilde{\mathbf{g}}_k = \nabla \tilde{f}(\mathbf{x}_{Z,k})$, and β_k is the CG update parameter. Different CG methods correspond to different choices for the scalar β_k (Hager and Zhang, 2006, Andrei, 2008). Table 1 provides a (partial) list of some choices for the CG update parameters.

CG update parameter	*Authors*
$\beta_k^{HS} = \frac{\tilde{\mathbf{g}}_k^T \mathbf{y}_{k-1}}{\mathbf{d}_{k-1}^T \mathbf{y}_{k-1}}$	Hestenes and Stiefel (1952)
$\beta_k^{FR} = \frac{\tilde{\mathbf{g}}_k^T \tilde{\mathbf{g}}_k}{\tilde{\mathbf{g}}_{k-1}^T \tilde{\mathbf{g}}_{k-1}}$	Fletcher and Reeves (1964)
$\beta_k^{PRP} = \frac{\tilde{\mathbf{g}}_k^T \mathbf{y}_{k-1}}{\tilde{\mathbf{g}}_{k-1}^T \tilde{\mathbf{g}}_{k-1}}$	Polak and Ribière (1969) and Polyak (1969)
$\beta_k^{PRP+} = \max\{0, \frac{\tilde{\mathbf{g}}_{k+1}^T \mathbf{y}_k}{\tilde{\mathbf{g}}_k^T \tilde{\mathbf{g}}_k}\}$	Powell (1984)
$\beta_k^{CD} = \frac{\tilde{\mathbf{g}}_{k+1}^T \tilde{\mathbf{g}}_k}{-\mathbf{d}_{k-1}^T \tilde{\mathbf{g}}_{k-1}}$	Fletcher (1987) (CD stands for Conjugate Descent)
$\beta_k^{DY} = \frac{\tilde{\mathbf{g}}_k^T \tilde{\mathbf{g}}_k}{\mathbf{d}_{k-1}^T \mathbf{y}_{k-1}}$	Dai and Yuan (1999)

Table 1: Various choices for the CG update parameter ($\mathbf{y}_k = \tilde{\mathbf{g}}_k - \tilde{\mathbf{g}}_{k-1}$)

Notice that if $\tilde{f}$ is a strongly convex quadratic, then in theory all 6 choices for the update parameter in table 1 are equivalent with an exact line search. For non-quadratic functions, however, each choice for the update parameter leads to different performance.

Since numerical experience indicates that the $PRP+$ algorithm tends to be more robust and efficient (Gilbert and Nocedal, 1992; Nocedal and Wright, 2006, pp. 122-124), in this paper we use β^{PRP+} as CG update parameter. We consider also a *restarting strategy*[9] suggested by Powell (1977), which consists in setting $\beta_k = 0$ whenever two consecutive gradients are far from orthogonal, as measured by the condition (Nocedal and Wright, 2006, p. 125)

$$\frac{|\tilde{\mathbf{g}}_k^T \tilde{\mathbf{g}}_{k-1}|}{\tilde{\mathbf{g}}_k^T \tilde{\mathbf{g}}_k} \geq 0.1. \tag{21}$$

6 Projected Steepest Descent and Conjugate Gradient Directions

Causey and Trager (1981) developed a benchmarking procedure grounded on a constrained Steepest Descent (SD) algorithm. Brown (2010) suggests to apply a similar procedure by using the non-linear Conjugate Gradient (CG) algorithm. In both cases the minimization problem is solved in the original variables $\mathbf{x}$, by using a *feasible directions method* according to which the iterations are given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k,$$

[9]As Nocedal and Wright (2006, p. 124) stress, "Restarting serves to periodically refresh the algorithm, erasing old information that may not be beneficial".

where α_k is a positive step length, and $\mathbf{v}_k$ is such that $\mathbf{A}\mathbf{v}_k = \mathbf{0}$. Thanks to this property, if $\mathbf{x}_k$ is feasible ($\mathbf{A}\mathbf{x}_k = \mathbf{b}$), then $\mathbf{x}_{k+1}$ is feasible too (i.e, $\mathbf{v}_k$ is a feasible direction).

The main idea is to 'project' at each iteration the unconstrained search direction $\mathbf{d}_k \in \mathbb{R}^n$ onto the null-space of matrix $\mathbf{A}$ by means of the $(n \times n)$ orthogonal projection matrix

$$\mathbf{V}_k = \frac{1}{v_k}\left[\mathbf{I}_n - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A}\right] = \frac{1}{v_k}\mathbf{N},$$

where $v_k = \left(\mathbf{d}_k^T\mathbf{N}\mathbf{d}_k\right)^{\frac{1}{2}}$, and to compute $\mathbf{v}_k = -\mathbf{V}_k\mathbf{d}_k$. Since $\mathbf{A}\mathbf{v}_k = \mathbf{0}$, $\mathbf{v}_k$ is a feasible direction. It can be shown (see the Appendix 1) that matrix $\mathbf{V}_k$ is an orthogonal projection matrix onto the affine feasible set $\mathcal{F}$ defined by the constraints.

Causey and Trager (1981) compute the unconstrained search direction as the steepest descent direction, namely $\mathbf{d}_k = -\mathbf{g}_k$. Brown (2010) suggests the update formula $\mathbf{d}_k = -\mathbf{g}_k + \beta_k^{PRP}\mathbf{d}_{k-1}$, within a non-linear *CG-PRP* algorithm with a restart procedure (Powell, 1977; see section 5.3).

7 Solvers' efficiency and quality

In this section we present numerical results about the performance of the techniques considered so far on the artificial series used by Denton (1971) in his seminal paper on benchmarking (section 7.2), and in benchmarking 61 quarterly series and 236 monthly series to their annual counterparts (section 7.3).

The *GRP*-benchmarked series are computed by applying the following unconstrained non-linear optimization algorithms to the reduced problem (15) obtained by transformation and elimination of some original variables:

- Steepest Descent (*SD*),
- Conjugate Gradient (*CG*),
- Quasi-Newton *BFGS* (*QN-BFGS*),
- Newton's method with Hessian modification (*MN*).

Even though this is not the main focus of the paper, when presenting the results we will look at the ability of the *GRP* benchmarked estimates in preserving the dynamics of the preliminary series, as compared to the Denton's *PFD* solution. We use the two indices (Di Fonzo and Marini, 2010):

$$r_q = \left(\frac{\sum_{t=2}^{n}\left|\frac{x_t^{GRP}}{x_{t-1}^{GRP}} - \frac{p_t}{p_{t-1}}\right|^q}{\sum_{t=2}^{n}\left|\frac{x_t^{PFD}}{x_{t-1}^{PFD}} - \frac{p_t}{p_{t-1}}\right|^q}\right)^{\frac{1}{q}}, \qquad q = 1, 2, \tag{22}$$

where the series $\mathbf{x}^{GRP}$ have been calculated using the algorithms described so far.

Index r_1 can be seen as the ratio between two mean absolute differences between the growth rates of the benchmarked (*GRP* and *PFD*, respectively) and the preliminary series. Sometimes this index can be larger than 1, thus indicating that, according to this metric, the movement is better preserved by Denton *PFD*. When $q = 2$, the index (22) is simply the square root of the ratio between the Causey and Trager movement preservation criteria (2), computed for the *GRP* and the *PFD* benchmarked estimates, respectively. Obviously, we expect the *GRP* technique always reaches a lower (or at least equal) value of the chosen criterion than *PFD*, and thus the index r_2 should be never larger than 1.

We have used the Matlab function `minFunc` (Schmidt, 2006), which is a free analogous of the function `Fminunc` of the Optimization Toolbox of Matlab (The Mathworks, 2009). A valuable feature of `minFunc` is that the scripts of the function are available to the user, who can change them according to her/his needs[10].

As for the options used, the conjugate gradient's update parameter (β_k) is computed according to the *PRP+* formula (see table 1), and the restart condition (21) is considered. Convergence is achieved when the norm of the reduced gradient of the objective function is negligible. More precisely, a *GRP* benchmarked series $\mathbf{x}^* = \bar{\mathbf{x}} + \mathbf{Z}\mathbf{x}_Z^*$ is obtained when

$$\|\nabla\tilde{f}\,(\mathbf{x}_Z^*)\,\|_1 \equiv \sum_{i=1}^{n-m} |\tilde{g}_i\,(\mathbf{x}_Z^*)| \leq 10^{-7}, \tag{23}$$

where $\tilde{g}_i\,(\mathbf{x}_Z^*)$, $i = 1, \ldots, n-m$, is the generic element of the reduced gradient vector $\nabla\tilde{f}\,(\mathbf{x}_Z^*)$. If condition (23) is not satisfied after 5,000 iterations, the algorithm ends, and returns the most recent (feasible) solution[11].

For comparisons' completeness, we consider also the *GRP* benchmarked series produced by the DOS-executable programme `BMK1.exe`, based on the projected steepest descent algorithm by Causey and Trager (1981, see section 6), which has been used for a long time by the U.S. Bureau of the Census.

The convergence condition of this programme is

$$\frac{f\,(\mathbf{x}_{k-1})}{f\,(\mathbf{x}_k)} < 1.00001,$$

which must be fulfilled within 200 iterations. No information on the number of function evaluations is given, and the $\mathbf{x}^{PFD}$ series is returned as the final solution if the algorithm has a breakdown (this never happened for the series we consider in the paper).

Due to the limited possibilities of 'tuning' the optimization options of `BMK1`, we used it as a sort of 'black-box'. The comparisons could thus seem rather unfair. Indeed, in our view such comparisons should only serve to give an idea of the improvements (if any) we can obtain by using the procedures we present in this

[10]For example, the original function does not compute the *PRP+* variant of the *CG* algorithm, and considers a restart condition slightly different from (21).

[11] The experiments were run on a PC equipped with a 32-bit Intel I5 processor with 4GB of RAM memory and Windows 7 Professional.

paper, as compared to the only (as far as we know) currently available tool for GRP benchmarking.

According to the specialized literature (Mittelmann and Pruessner, 2006), in order to compare different solvers/algorithms for NLP problems we should consider (i) efficiency, (ii) robustness, and (iii) quality of solution of the solvers.

Efficiency, which refers to the amount of computation resources needed to find the solution, is generally measured in terms of solver resource time (runtime). Robustness refers to the ability of the solver to succeed in finding one solution, and is generally measured by the number of problems for which a feasible solution is produced (the labelling of a solution as either 'successfull' or not, is usually summarized by a solve status return code). While considering these two aspects is sufficient when dealing with convex minimization problems (such as in linear programs or for certain quadratic programs), where the found minimum is generally the global one, for non-convex models, which may admit several local minima, other factors involving solution quality play an important role as well. For example, one solver may indeed be more efficient (i.e., faster), but the solution may be worse than that of a solver which is slower in terms of elapsed time.

For the problem in hand, however, robustness is not a concern, since all the techniques we consider are 'feasible point methods' - i.e. at each iterate they produce series in line with the temporal aggregation constraints - designed in such a way as they always give solutions 'not worse' than $\mathbf{x}^{PFD}$. In other words, in any case a feasible solution, say $\tilde{\mathbf{x}}$, is obtained, such that $\mathbf{A}\tilde{\mathbf{x}} = \mathbf{b}$, and $f(\tilde{\mathbf{x}}) \leq f\left(\mathbf{x}^{PFD}\right)$.

Therefore, if we were only interested in the efficiency in finding a local minimum, we would simply look for the fastest solver. Instead, if we wish that the comparison takes into account also the quality of the solution, it seems sensible to consider the best solution within the available ones,

$$\hat{\mathbf{x}} = \arg\min_{\tilde{\mathbf{x}}} f(\tilde{\mathbf{x}}),$$

and to refer to the relative objective value error between $\tilde{\mathbf{x}}$ and $\hat{\mathbf{x}}$. More precisely, given a positive small tolerance δ, the expression[12]

$$\frac{f(\tilde{\mathbf{x}}) - f(\hat{\mathbf{x}})}{f(\hat{\mathbf{x}})} \leq \delta \tag{24}$$

can be used to define a simple quality ranking between the solutions provided by different solvers, which turns out to be effective when a large number of problems has to be considered. Clearly, the 'true' best objective value corresponds to a choice of $\delta = 0$, but actually a tolerance close to 0 is used (e.g., $\delta = 0.0001$). Thus we say that the solution $\tilde{\mathbf{x}}$ is

1. the best, if expression (24) holds for $\delta = 0.0001$;

2. very accurate, if expression (24) holds for $\delta = 0.001$;

[12]The relative objective value error is usually defined by considering the absolute values of both the numerator and denominator of expression (24). Here this is not necessary, because it is always $f(\tilde{\mathbf{x}}) \leq f(\hat{\mathbf{x}})$, and $f(\hat{\mathbf{x}}) > 0$.

3. accurate, if expression (24) holds for $\delta = 0.01$;

4. acceptable, if expression (24) holds for $\delta = 0.1$.

In other words, we consider very accurate a solution whose objective function is within 0.1% of the best possible solution, accurate within 1%, and acceptable within 10%. A solution for which the objective value is 10% larger than the best one, is considered of bad quality.

The main concern in presenting an efficiency comparison involving several solvers, is in removing some of the ambiguity in interpreting the results, mostly if the number of problems is high (in our case, each series to be benchmarked gives raise to a NLP problem). At this end, we accompany the above quality information with the performance profiles (Dolan and Morè, 2002), a descriptive tool providing a wealth of information such as efficiency, robustness and probability of success of the technique in a very impressive graphical form, which can also include information on quality of solution, as it has been previously defined. We briefly summarize this effective tool in the next section.

7.1 Performance Profiles

Dolan and Moré (2002) define an efficiency comparison[13] in terms of a set $\mathcal{P}$ of n_p problems to be solved, and a set $\mathcal{S}$ of n_s optimization algorithms (solvers). Let $t_{p,s}$ be (say) the solver resource time used by solver s on problem p. A *performance ratio* can be defined as

$$\rho_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : 1 \leq s \leq n_s\}}. \tag{25}$$

For solvers s that do not solve problem p, we adopt the convention $\rho_{p,s} = \infty$ (in practice, we set a value $\rho_M = 2 \max \rho_{p,s}$). The *performance profile* of a solver $s \in \mathcal{S}$ is defined as the fraction of problems where the performance ratio is at most τ:

$$\rho_s(\tau) = \frac{1}{n_p} \text{size}\{p \in P : \rho_{p,s} \leq \tau\}. \tag{26}$$

Thus $\rho_s(\tau) : \mathbb{R} \rightarrow [0, 1]$ is the probability that a performance ratio $\rho_{p,s}$ is within τ of the best ratio. The function in equation (26) is the cumulative distribution function for the performance ratio in equation (25). Furthermore, it is piecewise constant, monotonically increasing and continuous from the right at each of the breakpoints. Note that the best solver for a particular problem attains the lower bound $\rho_{p,s} = 1$.

A performance profile seeks to capture how well the solver performs relative to the other solvers in $\mathcal{S}$ on the set of problems in $\mathcal{P}$. Note, in particular, that $\rho_s(1)$ is the fraction of problems for which solver $s \in \mathcal{S}$ performs the best and that for τ sufficiently large, $\rho_s(\tau)$ is the fraction of problems solved by $s \in \mathcal{S}$. In general, $\rho_s(\tau)$ is the fraction of problems with a performance ratio $\rho_{p,s}$ bounded by τ, and thus solvers with high values for $\rho_s(\tau)$ are preferable.

[13] In order to avoid confusion in the terminology, we changed the original term 'benchmark', used by Dolan and Moré (2002), with 'efficiency comparison'.

The profile gives much information, including information about solver robustness and efficiency. If a user is only interested in solver efficiency, then he can examine profile values $\rho_s(\tau)$ for $\tau = 1$ of different solvers s. The values $\rho_s(1)$ specifies the probability that a solver will 'win' over all other solvers. For the profile defined in terms of computing time, we define a 'win' as the solver who finds any optimal solution in the least amount of time. It is clearly possible to choose different definitions of 'win' based on different performance ratios (e.g., defined in terms of number of iterations, of function evaluations, of objective functions at the minimum).

If a user is only interested in the probability of success of a solver for the problem set $\mathcal{P}$, then the user may examine

$$\lim_{\tau \to \infty} \rho_s(\tau) \tag{27}$$

For ratios τ approaching ∞, we are looking at the probability of success of a solver given unlimited resource time.

If the user is also interested in information on quality of the solution returned by a solver (which is of particular interest for non-convex problems, as the one of interest for us, see section 3), the ratio (25) can be modified as follows. If $o_{p,s}$ is the value of the objective function at the solution found by solver s for problem p, and b_p is the best value within those found by all solvers $s \in S$ for problem p, we define a new performance ratio as

$$\rho_{p,s} = \begin{cases} \dfrac{t_{p,s}}{\min\{t_{p,s} : 1 \leq s \leq n_s\}} & \text{if } \left|\dfrac{o_{p,s} - b_p}{b_p}\right| \leq \delta \\ \rho_M & \text{if } \left|\dfrac{o_{p,s} - b_p}{b_p}\right| > \delta \end{cases}, \tag{28}$$

where δ is, as before, a user-defined relative objective function difference threshold, and ρ_M is an upper bound on $\rho_{p,s}$ over all problems p and solvers s. The ratio (28) is similar as before, except that we consider a solver successful only if the returned solution is within δ of the best solution found.

7.2 Denton (1971) series

The first example we consider is the series used in the seminal paper of Denton (1971). It consists of a five-year artificial quarterly series, with a fixed seasonal pattern invariant from year to year. The values are 50, 100, 150 and 100 in the four quarters, for a total yearly amount of 400. The annual benchmarks are assumed to be 500, 400, 300, 400 and 500 in the five successive years. The corresponding discrepancies (i.e., the differences between the known benchmarks and the annual sums of the preliminary series) are therefore 100, 0, -100, 0 and 100, respectively.

The performance of the *GRP* benchmarked series as compared to the *PFD* one has been analyzed by Di Fonzo and Marini (2010). Here suffice to say that, as expected, the *GRP* procedure shows better results as regards the movement preservation ($r_1 = 0.539$ and $r_2 = 0.553$).

Figure 1 and Table 2 show the different performances obtained by the considered procedures[14]. The Newton's method, which uses the analytic Hessian formula, need very few iterations and function evaluations (in both cases, 4) to converge, whereas after 4 iterations the other algorithms are rather far from the minimum. However, all the procedures succeed in finding the minimum of the objective function. According to the quality ranking defined in section 7, all solvers yield 'very accurate' solutions, the 'best' being given by quasi-Newton and modified Newton's methods.

The steepest descent (as we implemented it) appears to be less performing as compared to the conjugate gradient (a larger objective function, 36 iterations *vs.* 15, and 113 function evaluations *vs.* 57). Notice that the two steepest descent-based techniques - $BMK1$ and SD - show very similar performance, with a slight preference for $BMK1$, both in terms of target criterion and number of iterations. As for the quasi-Newton $BFGS$ performance, the objective function and the number of function evaluations are less than SD and CG, but the number of iterations is larger.

Algorithm	n. of iterations	n. of function evaluations	Objective function
Steepest descent ($BMK1$)	30	n.a.	0.04412603
Steepest descent	36	113	0.04412774
Conjugate gradient	15	57	0.04412700
Quasi-Newton $BFGS$	39	41	0.04411658
Newton with Hessian modification	4	4	0.04411656

Table 2: Denton series: iterations, function evaluations and final GRP function

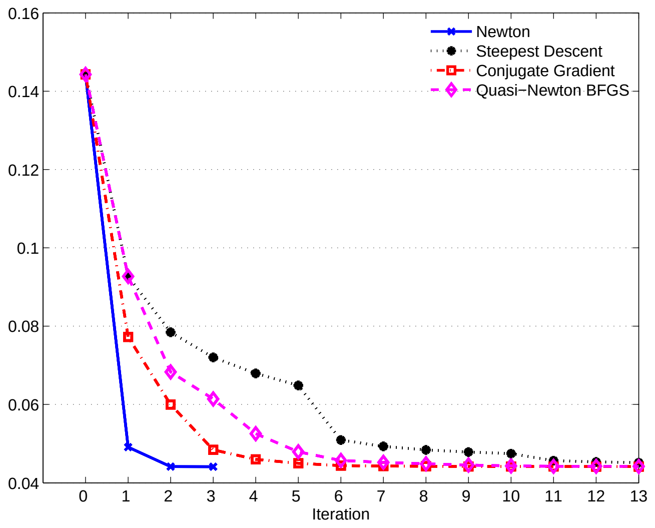


Figure 1: Denton (1971) series: GRP objective function in the first 14 iterations steps

[14]In figure 1 $BMK1$ is not present, because it does not provide the needed information.

7.3 EUQSA and MRTS series

The EUQSA dataset consists of 61 raw (not seasonally adjusted) quarterly series from the European Quarterly Sector Accounts, which are not in line with their known annual counterpart. The preliminary series span the period from 1999-Q1 to 2005-Q4 (28 quarters), and annual benchmarks are available for each variable[15].

As shown in Appendix 2, these variables present temporal discrepancies which are in some cases very small (less than 0.5% of the original level) and in other cases rather large (up to 50% of the original level). As it is usual in National Accounts, the temporal discrepancies for a single variable are often either all negative or all positive, which is a clear indication that the preliminary quarterly series are biased with respect to the annual benchmarks.

We consider also 236 monthly series of the Canadian Monthly Retail Trade Survey. For 226 out of 236 series, the dataset covers the period from January 1991 to December 2003, while the remaining 10 series start on January 1999. When these flow series are seasonally adjusted (SA), the temporal aggregation constraints valid for the raw series are typically 'destroyed', since the annual sum of the SA series might show differences with the annual totals from the raw series, due to the fact that a non-deterministic seasonal component is normally assumed.

We mimic the situations faced by a data-producer wishing to restore the temporal additivity relationships between the SA and the raw data. The X12-ARIMA procedure was applied to the 236 monthly series with automatic options. Obviously, we did not use the optional spec `FORCE` (U.S. Census Bureau, 2009), so the yearly sums of the SA series are in general different from those of the original series[16]. The computations have been done using the interface program Demetra (version 2.2, see Eurostat, 2007). The SA series resulting from X12-ARIMA have thus been considered as preliminary SA series to be temporally benchmarked.

The temporal discrepancies are quite variable (see Appendix 3), but less marked than those of the EUQSA series. In fact, the mean absolute percentage discrepancy ranges from 0.02% to 4.52%.

Table 3 reports on the quality, according to the previously defined metric, of the solutions found for the 61 EUQSA series and the 236 MRTS series. The first column refers to the series benchmarked according to Denton PFD, which is used as starting point by all the NLP solvers considered in this work. Clearly, Denton PFD is not a true GRP benchmarking procedure, but it is generally considered a good approximation of it. In this comparison it is used as a sort of 'baseline': for the whole set of 297 series, and with reference to the GRP objective function (2), Denton PFD yields solutions which are acceptable in about 80% of cases, accurate in about 37% and very accurate in about 2%, thus confirming the good approximation property generally claimed in literature. Anyway, in about 20% of cases this does not hold true, as the solutions by Denton PFD attain a GRP criterion which is

[15]A complete description of both EUQSA and MRTS datasets can be found in Di Fonzo and Marini (2011).

[16]We stress that the quality of seasonal adjustment is not a primary concern of the paper. We have replicated the exercise performing the seasonal adjustment by TRAMO-SEATS, and the results we found as regards the different impact of the GRP and PFD benchmarking procedures on the temporal profiles of the SA series, were not significantly affected.

more than 10% larger than the best one.

Quality of solution (tol.%)	**Denton PFD**	**SD BMK1**	**SD**	**CG**	**QN BFGS**	**Newton**
			EUQSA (61 series)			
Bad (>10%)	26	1	4	3	1	0
Acceptable (10%)	35	60	57	58	60	61
Accurate (1%)	15	60	49	51	54	61
Very accurate (0.1%)	3	60	30	34	44	61
Best (0.01%)	0	51	21	25	38	61
			MRTS (236 series)			
Bad (>10%)	33	7	6	1	0	0
Acceptable (10%)	203	229	230	235	236	236
Accurate (1%)	96	227	207	216	218	236
Very accurate (0.1%)	3	212	102	125	162	236
Best (0.01%)	0	94	46	74	153	235
			TOTAL (297 series)			
Bad (>10%)	59	8	10	4	1	0
Acceptable (10%)	238	289	287	293	296	297
Accurate (1%)	111	287	256	267	272	297
Very accurate (0.1%)	6	272	132	159	206	297
Best (0.01%)	0	145	67	99	191	296
			TOTAL (%)			
Bad (>10%)	19.87	2.69	3.37	1.35	0.34	0
Acceptable (10%)	80.13	97.31	96.63	98.65	99.66	100
Accurate (1%)	37.37	96.63	86.20	89.90	91.58	100
Very accurate (0.1%)	2.02	91.58	44.44	53.54	69.36	100
Best (0.01%)	0.00	48.82	22.56	33.33	64.31	99.66

Table 3: EUQSA and MRTS series: quality of the solutions found with different GRP benchmarking procedures

Passing now to consider the 'true' NLP solvers, the *a priori* expectation of a predominance of the Newton's method with Hessian modification is fully confirmed by the results: the Hessian-based procedure never results in solutions of bad quality, and produces by far the best results for almost all series, the unique exception being one of the MRTS series, for which the solution is very accurate, but cannot be considered as the best.

All the gradient-based procedures produce some solutions of bad quality (1 series out of 296 for $QN\text{-}BFGS$, 4 for CG, 8 for $SD\text{-}BMK1$ and 10 for SD). Furthermore, we note that the $SD\text{-}BMK1$ algorithm is uniformly better than the SD solver, and produces very accurate solutions in over 91% of cases, a very good performance as compared to more sophisticated optimization algorithms, as CG and $QN\text{-}BFGS$ are.

Table 4 presents some information on the computational efforts made by each solver, namely median, standard deviation and maximum values of number of iterations, function evaluations, and elapsed time. We restrict this comparison on very accurate solutions only (reported in the first column), that is on those problems where solvers achieve a final objective function satisfying condition (24) with $\delta = 10^{-3}$. It is confirmed the good performance of the Newton's method, that solved all problems within few iterations and function evaluations, with a maximum runtime under 0.1 seconds.

		Iterations			**Func. Evaluations**			**Runtime**		
Algorithm	# series	*med*	*std*	*max*	*med*	*std*	*max*	*med*	*std*	*max*
					EUQSA (61 series)					
*SD-BMK*1	60	11	18	70	n.a.	n.a.	n.a.	0.06	0.26	2.04
SD	30	35	95	352	219	693	2578	0.08	0.21	0.79
CG	34	10	499	2922	84	1031	6094	0.03	0.32	1.92
QN-BFGS	44	13	40	263	20	40	264	0.01	0.01	0.08
MN	61	1	1	4	2	1	7	0.00	0.01	0.09
					MRTS (236 series)					
*SD-BMK*1	212	13	11	64	n.a.	n.a.	n.a.	0.07	0.01	0.12
SD	102	38	662	3307	262	2487	10004	0.08	2.43	13.39
CG	125	17	590	4967	128	1244	10001	0.04	2.56	27.67
QN-BFGS	162	37	154	1060	45	153	1061	0.03	0.17	1.57
MN	236	1	1	6	2	1	8	0.01	0.00	0.03

SD: steepest descent; *CG*: conjugate gradient; *QN*: quasi-Newton; *MN*: modified Newton.

Table 4: EUQSA and MRTS series: statistics on iterations, function evaluations and runtime for very accurate solutions ($\delta = 10^{-3}$)

Table 5 shows the same statistics on the subset of series for which each solver did not achieve a very accurate solution. For these problems, SD-$BMK1$ often reached the maximum number of iterations allowed by the programme (201). Steepest descent algorithms often display slow convergence rates and zigzagging when the objective function is flat around the minimum, as it might be the case for these series and other cases where SD shows a similar behavior. On the other hand, very often CG and QN-$BFGS$ stop after few iterations, when a significant descent direction is not found.

		Iterations			**Func. Evaluations**			**Runtime**		
Algorithm	# series	*med*	*std*	*max*	*med*	*std*	*max*	*med*	*std*	*max*
					EUQSA (61 series)					
*SD-BMK*1	1	201	0	201	n.a.	n.a.	n.a.	0.09	0.00	0.09
SD	31	1	897	5000	1	1459	8152	0.01	0.81	4.54
CG	27	1	2	8	1	15	59	0.00	0.00	0.02
QN-BFGS	17	1	1	7	1	3	15	0.00	0.00	0.01
MN	0	-	-	-	-	-	-	-	-	-
					MRTS (236 series)					
*SD-BMK*1	24	103	59	201	n.a.	n.a.	n.a.	0.11	0.02	0.16
SD	134	1	1169	5000	1	1714	10001	0.01	5.00	27.24
CG	111	1	604	4990	1	1335	10001	0.01	3.17	27.72
QN-BFGS	74	1	29	254	1	30	258	0.01	0.02	0.16
MN	0	-	-	-	-	-	-	-	-	-

SD: steepest descent; *CG*: conjugate gradient; *QN*: quasi-Newton; *MN*: modified Newton.

Table 5: EUQSA and MRTS series: statistics on iterations, function evaluations and runtime for not very accurate solutions ($\delta = 10^{-3}$)

Looking at these solutions, the movements in the original series are better preserved by the Newton's method. Tables 6 and 7 report median, standard deviation, minimum and maximum values of r_1 and r_2 indices, as defined by (22). The median value of MD is always smaller than those of other algorithms. Gradient-based solvers do not move away from the starting condition (i.e. the Denton PFD solution), and thus $r_2 = 1$ in most cases. However, it should be noted that the impact of these differences in the movements of the benchmarked series is relatively small, since it is practically impossible to detect them on a time-series graph.

	# series	r_1				r_2			
Algorithm	out of 61	*med*	*std*	*min*	*max*	*med*	*std*	*min*	*max*
*SD-BMK*1	1	0.323	0.000	0.323	0.323	0.233	0.000	0.233	0.233
MN		*0.133*	*0.000*	*0.133*	*0.133*	*0.108*	*0.000*	*0.108*	*0.108*
SD	31	1.000	0.186	0.230	1.004	1.000	0.198	0.181	1.000
MN		*0.979*	*0.213*	*0.133*	*1.005*	*0.987*	*0.212*	*0.108*	*0.999*
CG	27	1.000	0.153	0.389	1.000	1.000	0.155	0.361	1.000
MN		*0.979*	*0.169*	*0.306*	*1.005*	*0.988*	*0.162*	*0.324*	*0.999*
QN-BFGS	17	1.000	0.108	0.556	1.000	1.000	0.095	0.609	1.000
MN		*0.981*	*0.106*	*0.552*	*1.005*	*0.997*	*0.094*	*0.608*	*0.999*

SD: steepest descent; *CG*: conjugate gradient; *QN*: quasi-Newton; *MN*: modified Newton.

Table 6: EUQSA series: statistics on r_1 and r_2 indices for not very accurate solutions ($\delta = 10^{-3}$), and comparison with modified Newton's results

	# series	r_1				r_2			
Algorithm	out of 236	*med*	*std*	*min*	*max*	*med*	*std*	*min*	*max*
*SD-BMK*1	24	0.908	0.122	0.614	1.005	0.859	0.222	0.195	0.971
MN		*0.874*	*0.158*	*0.449*	*1.015*	*0.845*	*0.238*	*0.146*	*0.971*
SD	134	1.000	0.071	0.595	1.000	1.000	0.122	0.206	1.000
MN		*0.996*	*0.089*	*0.449*	*1.015*	*0.996*	*0.130*	*0.146*	*0.999*
CG	111	1.000	0.047	0.526	1.000	1.000	0.082	0.163	1.000
MN		*0.996*	*0.054*	*0.449*	*1.003*	*0.996*	*0.084*	*0.146*	*0.999*
QN-BFGS	74	1.000	0.018	0.843	1.000	1.000	0.029	0.747	1.000
MN		*0.997*	*0.018*	*0.847*	*1.000*	*0.997*	*0.029*	*0.745*	*0.999*

SD: steepest descent; *CG*: conjugate gradient; *QN*: quasi-Newton; *MN*: modified Newton.

Table 7: MRTS series: statistics on indices r_1 and r_2 for not very accurate solutions ($\delta = 10^{-3}$), and comparison with modified Newton's results

Figures 2 and 3 show the performance profiles, where the X-axis is expressed in $\log_2$-scale, based on runtime for, respectively, $\delta = 0.1$ and $\delta = 0.001$ in formula (28). The profiles refer to all series (297), and show the performance, as measured by the resource time[17], when acceptable solutions ($\delta = 0.1$) and very accurate solutions ($\delta = 0.001$) are considered. The former case gives us information about the efficiency of the solvers, while the latter shows their quality.

The best performance of the Newton's method, both in terms of efficiency and quality, is now confirmed also visually: the solutions are almost always of better quality ($\rho_s(1)$ is about 0.6 for acceptable solutions, and 0.7 for very accurate solutions),

[17]In Appendix 4 we present the performance profiles distinct by dataset, based on runtime and also on number of iterations and of function evaluations. In this last case, *SD-BMK*1 is missing, since the programme does not provide the needed information.

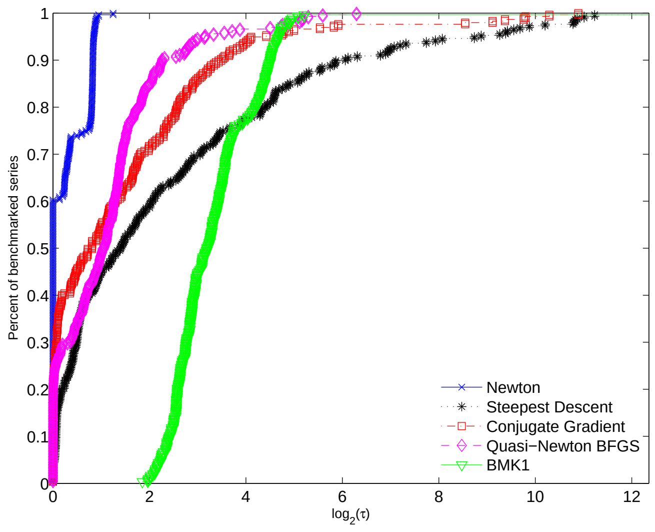


Figure 2: EUQSA and MRTS series: performance profiles for acceptable solutions ($\delta = 10^{-1}$)

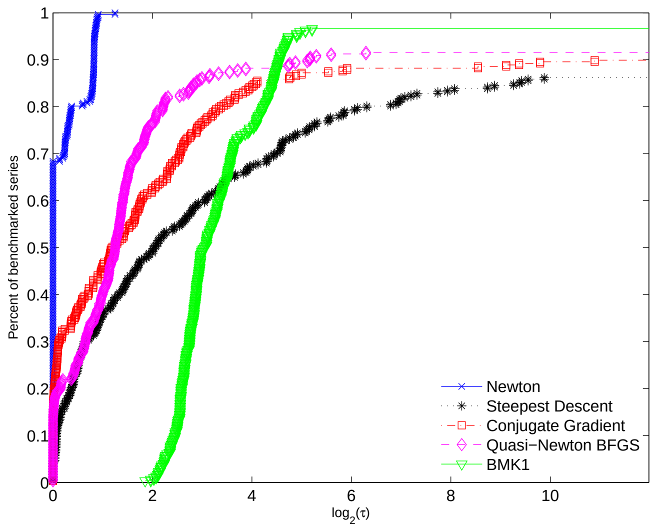


Figure 3: EUQSA and MRTS series: performance profiles for very accurate solutions ($\delta = 10^{-3}$)

and required less computation efforts than the other solvers. The quasi-Newton algorithm shows a valuable level of efficiency (almost all successful solutions in less time) as compared to the other gradient-based methods, followed by the conjugate gradient solver. It is also confirmed the good quality performance of $SD\text{-}BMK1$: for $\delta = 0.001$ and large τ, its curve is higher than the other gradient based methods[18].

[18]We think that one possible reason for this interesting result could be the line-search procedure worked out by Causey and Trager (1981), which looks for a step-length α_k within a specific bounded interval, whose limits depend on the objective function. This point is currently under study.

8 Conclusions

In this work, we present a Newton's method with Hessian modification for benchmarking a time series according to the growth rates preservation principle by Causey and Trager (1981). This method exploits the analytic Hessian of the GRP objective function, making full use of all the derivative information at disposal. In addition, we show that the proposed technique is easy to implement and robust, making it a plausible competitor of the classical modified Denton's PFD benchmarking procedure, also in a data-production process involving a considerable amount of series.

From the analysis of the artificial series of Denton (1971), and of quarterly and monthly real-life time series, it is strongly demonstrated that the new proposed technique overwhelmingly outperforms first-order (gradient-based) techniques, as the projected steepest descent technique originally proposed by Causey and Trager (1981), a technique based on the conjugate gradient algorithm (according to the recent proposal by Brown, 2010), and a Quasi-Newton method which at each iteration makes use of an approximated Hessian according to the $BFGS$ algorithm.

Our future work will be concentrated in (i) expanding these results also to the extrapolation case, when the benchmarks (one or more) for the most recent years are unavailable, and the relevant preliminary values have to be benchmarked, (ii) developing benchmarking techniques focussed on growth rates preservation principles other than GRP, for example by considering the absolute value instead of the square of the difference between target and preliminary growth rates:

$$\sum_{t=2}^{n} \left| \frac{x_t}{x_{t-1}} - \frac{p_t}{p_{t-1}} \right|,$$

(iii) considering also the 'annual' growth rates in the movement to be preserved (i.e., $\frac{x_t}{x_{t-s}}$ instead of, or combined with, $\frac{x_t}{x_{t-1}}$, see Fagan, 1995), and (iv) investigating the possibility of including the GRP principle in a reconciliation framework, where a system of time series has to be benchmarked in order to be in line with both temporal and contemporaneous constraints (Dagum and Cholette, 2006; Fortier and Quenneville, 2009; Di Fonzo and Marini, 2011).

As regards the first point, we are also studying the 'GRP-analog' of the regression-based benchmarking model by Dagum and Cholette (2006), according to which (in a simplified form) the benchmarked estimates are the solution to the constrained minimization problem of an objective function in the proportionate corrections $\frac{x_t - p_t}{p_t}$:

$$\sum_{t=2}^{n} \left(\frac{x_t - p_t}{p_t} - \phi \frac{x_{t-1} - p_{t-1}}{p_{t-1}} \right)^2,$$

where $0 < \phi < 1$ is a *smoothing* parameter, usually fixed by the user, which turns out to be useful mostly in extrapolation situations (Quenneville *et al.*, 2003). A rather natural extension would in fact consider

$$\sum_{t=2}^{n} \left[\left(\frac{x_t}{x_{t-1}} - 1 \right) - \phi \left(\frac{p_t}{p_{t-1}} - 1 \right) \right]^2 \quad \text{or} \quad \sum_{t=2}^{n} \left| \left(\frac{x_t}{x_{t-1}} - 1 \right) - \phi \left(\frac{p_t}{p_{t-1}} - 1 \right) \right|.$$

Appendix 1. Feasible direction according to Causey and Trager (1981)

Given an $(n \times 1)$ (unconstrained) direction vector $\mathbf{d}$, we are looking for an $(n \times 1)$ vector $\mathbf{v}$, solution to the following constrained linear problem:

$$\min_{\mathbf{v}} \mathbf{v}^T \mathbf{d} \quad \text{s.t.} \quad \mathbf{v}^T \mathbf{v} = 1 \quad \text{and} \quad \mathbf{A}\mathbf{v} = \mathbf{0}. \tag{29}$$

In other terms, we wish that the direction $\mathbf{v}$ have unitary norm ($\mathbf{v}^T\mathbf{v} = 1$), and be feasible ($\mathbf{A}\mathbf{v} = \mathbf{0}$).

Consider the Lagrangean function

$$\mathcal{L} = \mathbf{v}^T\mathbf{d} + \frac{\lambda^*}{2}\left(\mathbf{v}^T\mathbf{v} - 1\right) + \lambda^T\left(\mathbf{A}\mathbf{v}\right), \tag{30}$$

where λ^* and λ are a scalar and a $(m \times 1)$ vector of Lagrange multipliers, respectively. The first order condition is given by

$$\left\{\begin{array}{rcl} \dfrac{\partial \mathcal{L}}{\partial \mathbf{v}} & = & \mathbf{0} \\ \dfrac{\partial \mathcal{L}}{\partial \lambda^*} & = & 0 \\ \dfrac{\partial \mathcal{L}}{\partial \lambda} & = & \mathbf{0} \end{array}\right. \quad \Rightarrow \quad \begin{array}{rcl} \mathbf{d} + \lambda^*\mathbf{v} + \mathbf{A}^T\lambda & = & \mathbf{0} \\ \mathbf{v}^T\mathbf{v} & = & 1 \\ \mathbf{A}\mathbf{v} & = & \mathbf{0} \end{array}. \tag{31}$$

Pre-multiplying the first equation in (31) by matrix $\mathbf{A}$ and solving for λ, we find

$$\lambda = -\left(\mathbf{A}\mathbf{A}^T\right)^{-1}\mathbf{A}\mathbf{d}, \tag{32}$$

and, by sostitution in (31),

$$\lambda^*\mathbf{v} = -\left[\mathbf{I}_n - \mathbf{A}^T\left(\mathbf{A}\mathbf{A}^T\right)^{-1}\mathbf{A}\right]\mathbf{d}. \tag{33}$$

Noting that matrix $\mathbf{N} = \left[\mathbf{I}_n - \mathbf{A}^T\left(\mathbf{A}\mathbf{A}^T\right)^{-1}\mathbf{A}\right]$ is idempotent ($\mathbf{N}^T\mathbf{N} = \mathbf{N}$), and given that $\mathbf{v}^T\mathbf{v} = 1$, by taking the square of both sides of (33), we obtain:

$$\left(\lambda^*\right)^2\mathbf{v}^T\mathbf{v} = \mathbf{d}^T\mathbf{N}\mathbf{d} \quad \Rightarrow \quad \lambda^* \equiv v = \left(\mathbf{d}^T\mathbf{N}\mathbf{d}\right)^{\frac{1}{2}}.$$

Finally, by denoting $\mathbf{V} = \frac{1}{v}\mathbf{N}$, and after substitution in (33), we obtain the result:

$$\mathbf{v} = -\mathbf{V}\mathbf{d}.$$

Appendix 2. Temporal discrepancies (%) of the EUQSA series

Series	mean	absm	st. dev	min	max	range
1	0.00	0.11	0.12	-0.13	0.23	0.35
2	0.00	0.04	0.05	-0.11	0.05	0.16
3	0.00	0.02	0.03	-0.07	0.03	0.10
4	0.01	0.25	0.31	-0.44	0.49	0.92
5	0.01	0.22	0.23	-0.24	0.35	0.58
6	0.73	8.49	11.50	-11.96	26.00	37.96
7	1.08	14.07	17.42	-12.33	38.20	50.52
8	-0.06	0.67	0.78	-1.37	0.77	2.13
9	-0.01	0.48	0.54	-0.73	0.91	1.64
10	0.09	1.42	1.77	-2.29	3.45	5.74
11	0.06	1.09	1.36	-1.83	2.59	4.42
12	0.00	0.22	0.23	-0.24	0.34	0.58
13	-0.41	4.83	6.34	-10.59	10.69	21.29
14	0.41	12.07	14.08	-28.59	13.37	41.96
15	-0.09	0.67	0.80	-1.43	0.71	2.13
16	-0.34	0.57	0.57	-1.02	0.48	1.50
17	-0.01	0.13	0.14	-0.17	0.19	0.36
18	-0.01	0.23	0.24	-0.26	0.33	0.59
19	0.04	0.68	0.77	-1.09	1.22	2.31
20	0.02	0.52	0.59	-0.84	0.92	1.76
21	0.00	0.03	0.03	-0.04	0.05	0.09
22	-0.01	0.09	0.11	-0.20	0.11	0.31
23	-0.05	0.70	0.82	-1.43	0.80	2.24
24	0.03	0.41	0.46	-0.63	0.71	1.34
25	0.00	0.13	0.15	-0.20	0.25	0.45
26	1.80	12.08	14.52	-15.19	25.56	40.75
27	0.75	7.06	7.93	-10.36	10.62	20.98
28	-0.02	0.94	1.22	-2.43	1.49	3.92
29	0.04	0.77	1.02	-1.69	1.84	3.53
30	0.05	0.67	0.86	-0.66	1.87	2.53
31	-0.04	3.97	4.42	-6.91	6.40	13.31
32	0.00	0.03	0.03	-0.06	0.05	0.11
33	0.00	0.22	0.24	-0.34	0.24	0.58
34	0.39	3.60	4.82	-11.23	3.91	15.13
35	0.54	4.24	5.16	-5.82	10.62	16.44
36	0.22	2.45	3.35	-7.80	2.78	10.58
37	0.00	0.05	0.07	-0.12	0.09	0.22
38	-0.05	0.79	0.94	-1.49	1.07	2.56
39	-0.04	0.42	0.48	-0.74	0.63	1.37
40	0.00	0.13	0.19	-0.25	0.42	0.68
41	0.01	0.22	0.24	-0.34	0.25	0.58
42	-0.33	14.43	20.20	-17.50	47.11	64.60
43	-0.01	0.17	0.23	-0.27	0.45	0.72
44	-0.05	5.43	7.88	-6.49	18.31	24.80
45	0.01	0.06	0.08	-0.15	0.12	0.28
46	-0.11	0.76	0.94	-1.55	0.99	2.54
47	-0.09	0.46	0.50	-0.85	0.52	1.37
48	0.00	0.14	0.20	-0.25	0.44	0.69
49	0.01	0.23	0.24	-0.34	0.25	0.59
50	-0.01	0.51	0.60	-0.95	0.88	1.83
51	-0.01	0.51	0.60	-0.95	0.88	1.83
52	0.00	0.08	0.10	-0.19	0.15	0.34
53	0.00	0.02	0.02	-0.03	0.03	0.06
54	-0.06	0.78	0.94	-1.50	1.05	2.55
55	-0.02	0.42	0.48	-0.74	0.68	1.42
56	0.06	0.96	1.24	-2.71	1.02	3.73
57	-0.01	0.11	0.13	-0.25	0.13	0.38
58	0.00	0.13	0.19	-0.25	0.43	0.68
59	0.00	0.86	1.06	-0.77	2.31	3.08
60	0.04	0.61	0.71	-0.70	1.30	2.00
61	-0.29	3.73	4.09	-5.40	5.70	11.10

Appendix 3. Temporal discrepancies (%) of the MRTS series

Series	mean	absm	std	min	max	range
1	-0.02	0.34	0.55	-1.23	1.39	2.62
2	-0.10	0.41	0.43	-0.78	0.48	1.26
3	-0.06	0.15	0.17	-0.34	0.31	0.65
4	-0.13	0.28	0.32	-0.63	0.39	1.01
5	-0.22	0.39	0.37	-0.62	0.42	1.04
6	-0.07	0.15	0.16	-0.35	0.16	0.50
7	-0.07	0.16	0.20	-0.40	0.30	0.70
8	-0.09	0.14	0.16	-0.42	0.17	0.58
9	-0.06	0.23	0.28	-0.58	0.28	0.86
10	-0.02	0.12	0.18	-0.44	0.40	0.84
11	0.26	0.64	0.78	-1.05	1.90	2.95
12	0.28	0.51	0.65	-1.01	1.78	2.80
13	0.21	0.35	0.59	-0.46	2.08	2.55
14	-0.14	0.36	0.45	-1.23	0.49	1.72
15	-0.16	0.40	0.46	-0.93	0.77	1.70
16	-0.23	0.69	0.83	-2.11	1.06	3.17
17	-0.07	0.18	0.22	-0.52	0.36	0.87
18	-0.08	0.30	0.35	-0.79	0.49	1.28
19	0.25	0.42	0.49	-0.40	1.42	1.82
20	0.05	0.26	0.31	-0.50	0.68	1.18
21	0.00	0.20	0.26	-0.53	0.52	1.06
22	-0.09	0.26	0.27	-0.46	0.36	0.82
23	-0.08	0.60	0.75	-1.26	1.19	2.45
24	0.02	0.55	0.71	-0.91	1.91	2.82
25	-2.28	3.36	2.70	-4.69	2.69	7.38
26	-0.16	0.92	1.12	-2.13	2.49	4.61
27	-0.18	0.28	0.31	-0.67	0.33	1.00
28	-0.18	0.34	0.44	-1.14	0.37	1.51
29	-0.15	0.34	0.38	-0.63	0.61	1.24
30	-0.23	0.39	0.39	-0.83	0.40	1.23
31	-0.11	0.23	0.25	-0.39	0.62	1.00
32	-0.25	0.32	0.26	-0.73	0.23	0.96
33	-0.28	0.42	0.46	-1.30	0.54	1.83
34	-0.20	0.27	0.24	-0.61	0.28	0.89
35	-0.18	0.24	0.25	-0.59	0.20	0.80
36	-0.10	1.11	1.24	-2.08	1.82	3.90
37	-0.38	0.50	0.55	-1.31	0.33	1.64
38	0.72	0.78	0.75	-0.45	2.19	2.64
39	-0.31	0.85	0.89	-2.00	1.35	3.35
40	-0.05	0.72	0.84	-1.49	1.38	2.87
41	-0.37	0.50	0.53	-1.46	0.56	2.02
42	-0.18	0.23	0.24	-0.59	0.21	0.80
43	-0.26	0.32	0.32	-0.73	0.41	1.14
44	-0.09	0.32	0.40	-1.04	0.38	1.41
45	-0.16	0.40	0.44	-0.80	0.72	1.52
46	-0.25	0.31	0.26	-0.82	0.38	1.20
47	-0.19	0.25	0.22	-0.56	0.21	0.77
48	-0.36	0.72	0.77	-1.68	0.89	2.57
49	-0.01	2.36	2.94	-6.05	4.44	10.49
50	1.41	4.52	5.33	-6.60	10.99	17.59
51	0.72	2.65	3.13	-6.02	5.41	11.43
52	0.31	0.70	0.76	-1.02	1.50	2.51
53	1.11	2.17	2.49	-2.26	6.17	8.43
54	0.02	0.77	0.90	-1.36	1.82	3.18
55	0.02	0.44	0.51	-0.88	0.85	1.73
56	-0.01	0.27	0.35	-0.77	0.45	1.22
57	-0.15	0.19	0.14	-0.31	0.13	0.44
58	-0.06	0.34	0.41	-0.68	0.70	1.38
59	-0.08	0.16	0.21	-0.51	0.36	0.87
60	0.12	0.33	0.34	-0.58	0.52	1.11

% continue

% Appendix 3 continued

Series	mean	absm	std	min	max	range
61	0.57	1.35	1.89	-2.44	5.62	8.05
62	-1.21	2.85	3.21	-6.00	3.72	9.72
63	-0.49	0.94	1.01	-2.18	1.63	3.81
64	-0.48	1.27	1.64	-4.56	1.34	5.90
65	-0.30	0.50	0.56	-1.36	0.68	2.03
66	-0.25	0.64	0.73	-1.38	1.43	2.81
67	-0.31	0.48	0.53	-1.23	0.60	1.83
68	-0.30	0.58	0.61	-1.23	0.93	2.16
69	-0.40	0.46	0.51	-1.65	0.25	1.91
70	-0.46	0.85	0.89	-1.69	1.65	3.34
71	-0.42	0.43	0.39	-1.38	0.03	1.41
72	-0.36	0.45	0.35	-0.78	0.36	1.14
73	0.11	1.18	1.69	-2.27	5.03	7.29
74	-0.09	0.73	0.93	-1.28	2.24	3.52
75	-0.64	0.87	0.99	-2.65	1.49	4.15
76	-0.67	1.08	1.20	-2.68	0.99	3.67
77	-0.36	0.54	0.50	-1.29	0.93	2.22
78	-0.23	0.44	0.50	-1.02	0.46	1.49
79	-0.12	0.50	0.57	-1.00	0.83	1.83
80	-0.18	0.50	0.59	-1.32	1.01	2.34
81	-0.18	0.66	0.76	-1.64	1.32	2.96
82	-0.42	0.87	0.91	-2.04	1.33	3.37
83	-0.28	0.54	0.57	-1.25	1.13	2.38
84	0.02	0.76	0.91	-1.63	1.39	3.02
85	-0.17	1.31	1.52	-2.02	2.88	4.90
86	-0.21	2.38	2.80	-4.55	4.44	8.99
87	-0.18	0.49	0.56	-1.22	0.85	2.07
88	0.48	1.70	1.90	-2.98	3.06	6.04
89	0.21	0.73	1.64	-1.03	5.78	6.81
90	-0.77	1.16	1.51	-5.07	1.44	6.51
91	-0.09	0.56	0.73	-1.19	1.50	2.69
92	-0.33	0.54	0.53	-1.18	0.40	1.58
93	0.35	1.63	2.36	-2.23	7.20	9.43
94	0.03	0.71	1.14	-2.12	3.27	5.39
95	0.96	0.96	1.37	0.18	5.56	5.38
96	0.02	0.46	0.55	-0.60	1.12	1.72
97	1.52	2.89	4.02	-3.00	11.98	14.98
98	1.15	3.24	4.99	-6.62	14.20	20.83
99	-0.04	0.16	0.18	-0.30	0.19	0.49
100	-0.07	0.16	0.17	-0.30	0.22	0.52
101	0.03	0.09	0.12	-0.21	0.24	0.46
102	-0.05	0.15	0.17	-0.37	0.27	0.63
103	-0.03	0.13	0.14	-0.25	0.22	0.46
104	-0.05	0.14	0.16	-0.34	0.27	0.61
105	-0.06	0.14	0.16	-0.33	0.21	0.53
106	-0.08	0.13	0.15	-0.35	0.19	0.53
107	-0.07	0.12	0.13	-0.30	0.13	0.43
108	-0.04	0.13	0.16	-0.30	0.20	0.50
109	-0.12	0.15	0.15	-0.43	0.16	0.60
110	0.07	0.16	0.18	-0.28	0.34	0.62
111	-0.13	0.19	0.18	-0.36	0.15	0.51
112	-0.04	0.16	0.19	-0.34	0.23	0.56
113	-0.27	0.30	0.43	-1.64	0.14	1.78
114	-0.18	0.39	0.58	-1.97	0.41	2.38
115	-0.11	0.14	0.18	-0.49	0.10	0.59
116	-0.05	0.09	0.08	-0.19	0.10	0.28
117	-0.09	0.16	0.18	-0.46	0.25	0.71
118	-0.06	0.15	0.16	-0.27	0.20	0.47
119	-0.07	0.16	0.18	-0.35	0.40	0.75
120	-0.03	0.05	0.10	-0.36	0.05	0.41

% continue

% Appendix 3 continued

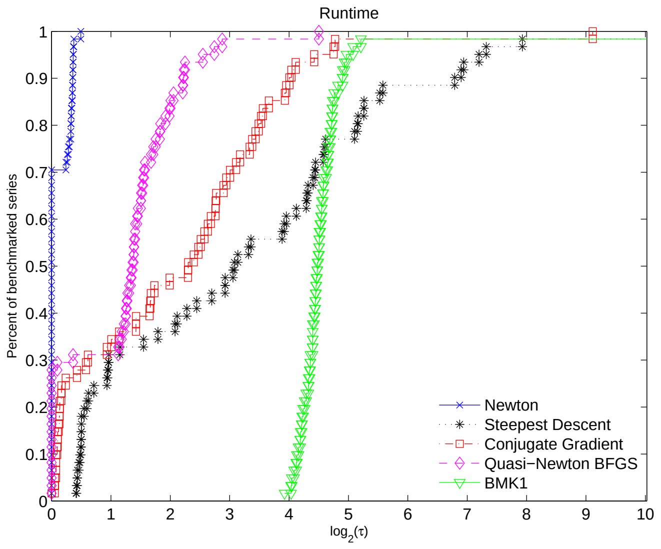
Series	mean	absm	std	min	max	range
121	-0.10	0.22	0.26	-0.61	0.36	0.97
122	-0.02	0.37	0.51	-0.48	1.47	1.95
123	-0.01	0.29	0.35	-0.67	0.39	1.06
124	-0.02	0.27	0.32	-0.43	0.50	0.93
125	0.03	1.10	1.43	-3.08	2.33	5.41
126	-0.03	0.38	0.49	-0.86	1.05	1.91
127	-0.14	0.20	0.21	-0.47	0.33	0.80
128	-0.27	0.36	0.37	-1.06	0.44	1.50
129	-0.52	0.71	0.81	-1.90	0.32	2.22
130	-0.19	0.44	0.53	-1.17	0.67	1.84
131	-0.14	0.23	0.30	-0.84	0.34	1.18
132	-0.07	0.44	0.62	-1.01	1.29	2.30
133	-0.26	0.27	0.28	-0.96	0.04	1.00
134	-0.09	0.32	0.37	-0.67	0.46	1.12
135	0.33	0.52	0.66	-0.53	1.85	2.38
136	-0.09	0.32	0.38	-0.94	0.61	1.55
137	0.05	1.65	2.15	-1.53	4.24	5.77
138	-0.04	0.12	0.16	-0.23	0.42	0.65
139	-0.02	0.08	0.10	-0.13	0.28	0.42
140	0.01	0.08	0.10	-0.14	0.19	0.33
141	-0.10	0.16	0.15	-0.32	0.20	0.52
142	0.03	0.09	0.10	-0.17	0.16	0.33
143	0.00	0.09	0.12	-0.34	0.17	0.51
144	-0.04	0.10	0.15	-0.40	0.23	0.62
145	-0.04	0.04	0.04	-0.11	0.01	0.12
146	-0.06	0.06	0.08	-0.24	0.02	0.26
147	-0.11	0.16	0.20	-0.63	0.27	0.90
148	0.01	0.02	0.02	-0.04	0.05	0.08
149	-0.03	0.15	0.20	-0.58	0.16	0.74
150	-0.05	0.10	0.16	-0.36	0.09	0.45
151	-0.02	0.17	0.22	-0.39	0.42	0.80
152	-0.04	0.38	0.47	-0.74	0.75	1.49
153	0.06	0.29	0.34	-0.42	0.73	1.15
154	0.00	0.21	0.24	-0.31	0.40	0.71
155	0.06	0.13	0.16	-0.22	0.42	0.64
156	0.02	0.12	0.17	-0.19	0.47	0.66
157	0.04	0.10	0.12	-0.19	0.22	0.41
158	0.00	0.10	0.12	-0.25	0.25	0.50
159	-0.03	0.10	0.12	-0.30	0.17	0.47
160	0.05	0.15	0.20	-0.30	0.47	0.78
161	-0.11	0.47	0.62	-1.44	0.81	2.25
162	0.04	0.38	0.47	-0.64	0.85	1.49
163	0.06	0.14	0.16	-0.16	0.33	0.49
164	-0.35	0.69	0.68	-1.41	1.11	2.53
165	-0.09	0.93	1.30	-2.56	2.65	5.21
166	-0.08	0.34	0.47	-1.10	0.86	1.97
167	-0.08	0.45	0.53	-0.87	0.96	1.83
168	-0.13	0.19	0.23	-0.78	0.21	0.99
169	-0.21	0.33	0.35	-1.07	0.50	1.57
170	-0.03	0.18	0.23	-0.42	0.47	0.89
171	-0.13	0.44	0.57	-1.30	0.59	1.89
172	-0.15	0.38	0.50	-1.36	0.57	1.93
173	-0.24	0.32	0.40	-1.28	0.26	1.54
174	-0.41	0.76	0.87	-2.14	1.02	3.16
175	-0.42	1.04	1.54	-4.97	1.50	6.46
176	-0.02	0.59	0.78	-1.72	1.22	2.94
177	-0.37	0.46	0.59	-2.06	0.45	2.51
178	0.07	0.42	0.54	-0.66	1.36	2.02
179	-0.11	0.65	0.74	-1.62	0.98	2.60
180	-0.13	0.29	0.31	-0.58	0.53	1.10

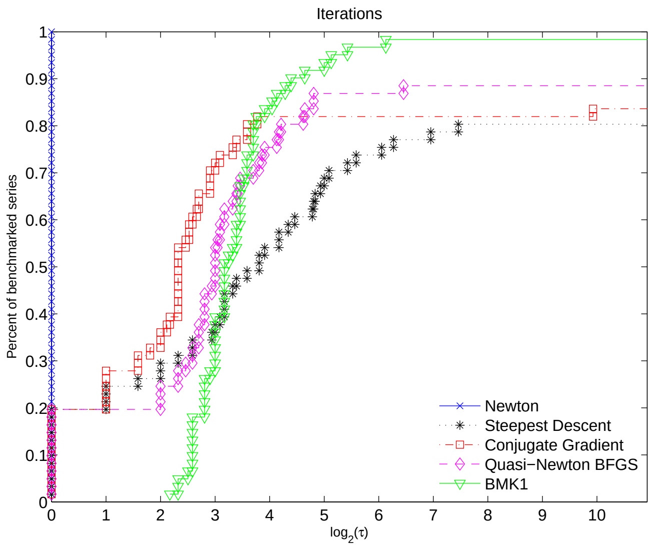
% continue

% Appendix 3 continued

Series	mean	absm	std	min	max	range
181	0.02	0.43	0.56	-0.59	1.34	1.93
182	-0.11	0.42	0.58	-1.66	0.71	2.37
183	-0.12	0.31	0.38	-0.73	0.59	1.32
184	-0.16	0.28	0.37	-0.98	0.33	1.32
185	-0.15	0.45	0.54	-1.24	0.86	2.10
186	-0.25	1.30	1.79	-4.63	1.52	6.16
187	-0.14	0.91	1.06	-2.30	1.25	3.55
188	-0.15	0.44	0.51	-1.04	0.59	1.62
189	-0.23	0.54	0.69	-1.64	0.78	2.43
190	-0.21	0.43	0.52	-1.42	0.63	2.05
191	-0.11	0.49	0.61	-1.14	1.32	2.46
192	-0.08	0.28	0.33	-0.66	0.35	1.01
193	-0.11	0.36	0.43	-0.84	0.89	1.73
194	-0.18	0.28	0.35	-0.74	0.32	1.06
195	-0.12	0.34	0.48	-1.15	0.82	1.97
196	-0.11	0.27	0.30	-0.67	0.51	1.18
197	-0.10	0.35	0.47	-1.32	0.41	1.73
198	-0.28	0.58	0.69	-1.70	0.98	2.69
199	0.21	0.77	0.86	-1.30	1.37	2.67
200	-0.31	0.45	0.46	-1.07	0.65	1.72
201	-0.13	1.17	1.75	-3.48	3.69	7.17
202	-0.31	0.47	0.51	-1.28	0.45	1.74
203	-0.31	0.40	0.36	-1.13	0.24	1.37
204	-0.20	0.25	0.20	-0.44	0.31	0.75
205	-0.19	0.37	0.38	-0.78	0.39	1.17
206	-0.16	0.24	0.30	-0.74	0.36	1.10
207	-0.17	0.19	0.17	-0.52	0.08	0.60
208	-0.22	0.39	0.44	-1.11	0.46	1.57
209	-0.04	0.23	0.27	-0.48	0.47	0.95
210	-0.23	0.23	0.17	-0.55	-0.02	0.52
211	0.15	0.50	0.66	-0.97	1.35	2.32
212	-0.22	0.46	0.50	-1.05	0.64	1.69
213	-0.08	0.29	0.33	-0.48	0.69	1.17
214	-0.07	0.32	0.46	-1.39	0.52	1.91
215	-0.23	0.26	0.23	-0.73	0.09	0.82
216	-0.18	0.23	0.23	-0.68	0.12	0.80
217	-0.14	0.23	0.26	-0.72	0.28	1.00
218	-0.13	0.23	0.27	-0.65	0.29	0.93
219	-0.13	0.29	0.32	-0.68	0.46	1.14
220	-0.20	0.26	0.26	-0.69	0.19	0.88
221	-0.57	0.72	0.66	-1.54	0.44	1.97
222	-0.07	0.15	0.15	-0.28	0.17	0.45
223	-0.10	0.17	0.17	-0.33	0.15	0.48
224	-0.15	0.64	0.83	-1.92	1.30	3.23
225	-0.64	1.09	1.27	-3.34	0.91	4.25
226	-0.30	0.59	0.58	-0.93	1.18	2.11
227	-0.03	0.03	0.02	-0.06	-0.01	0.05
228	0.03	0.13	0.16	-0.31	0.29	0.60
229	-0.07	0.08	0.07	-0.19	0.03	0.22
230	-0.29	0.36	0.36	-1.01	0.35	1.36
231	-0.02	0.34	0.41	-0.64	0.85	1.49
232	-0.17	0.21	0.20	-0.56	0.14	0.70
233	-0.13	0.22	0.24	-0.58	0.26	0.84
234	0.95	1.58	1.74	-1.65	3.66	5.31
235	0.18	0.77	0.92	-1.60	1.94	3.54

Appendix 4. Performance profiles





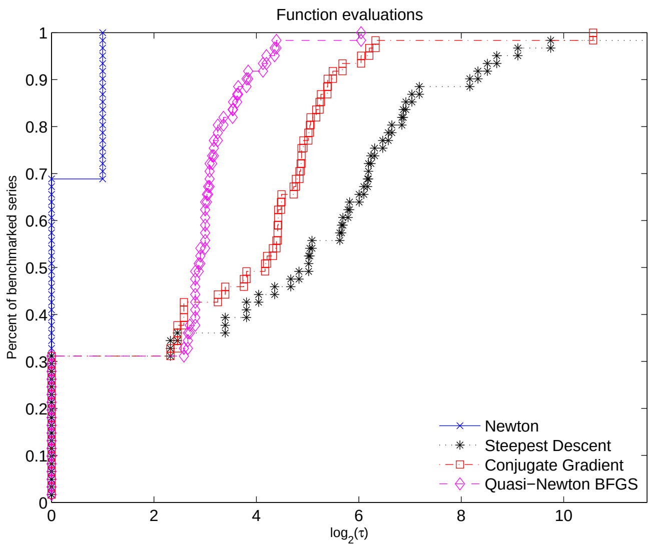


Figure 4: EUQSA series: performance profiles for acceptable solutions ($\delta = 10^{-1}$)

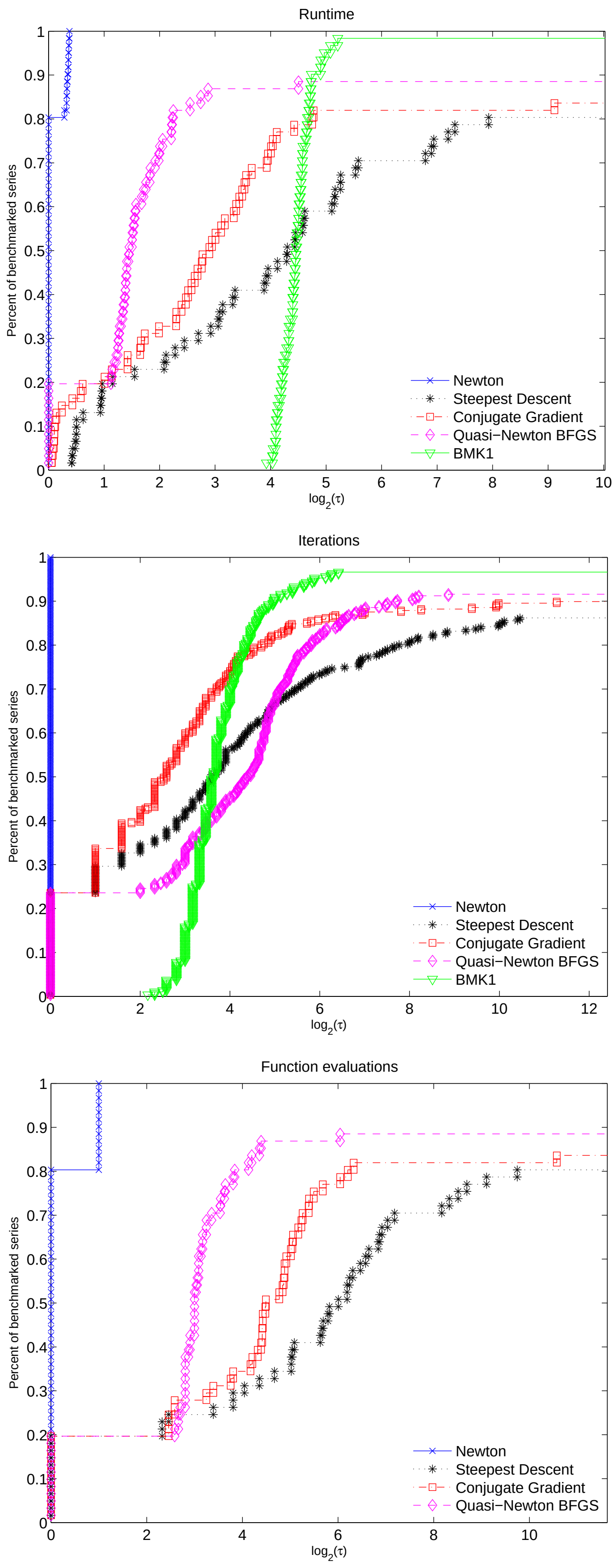
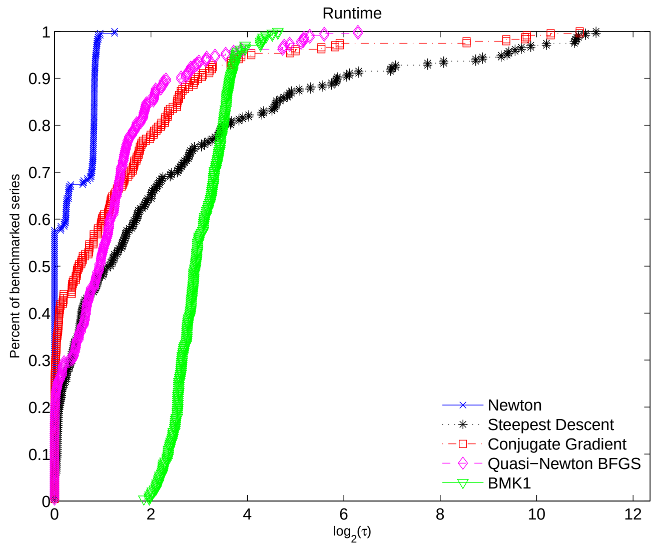
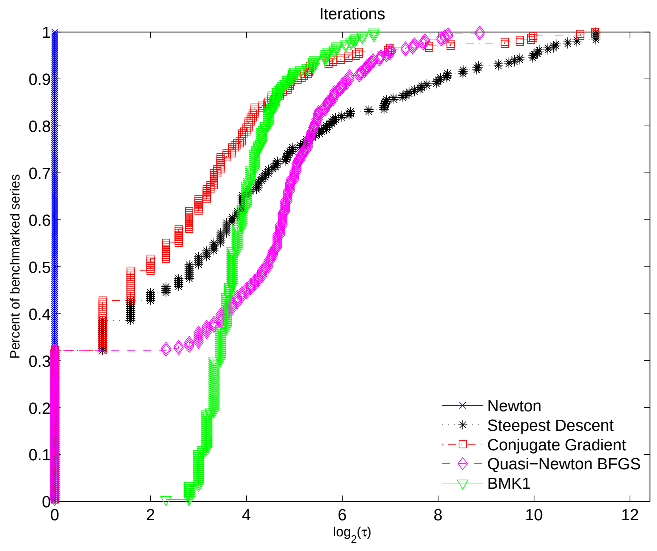


Figure 5: EUQSA series: performance profiles for very accurate solutions ($\delta = 10^{-3}$)





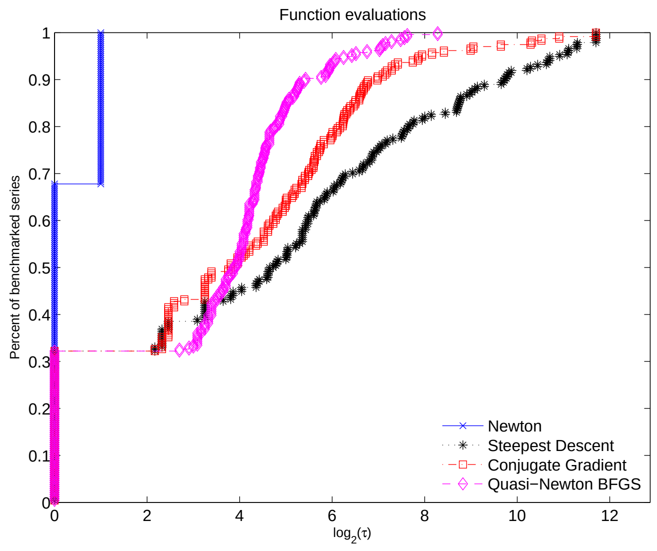
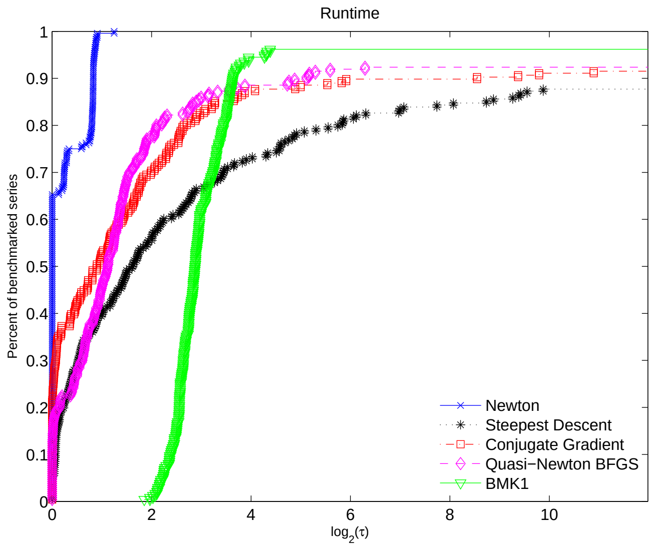
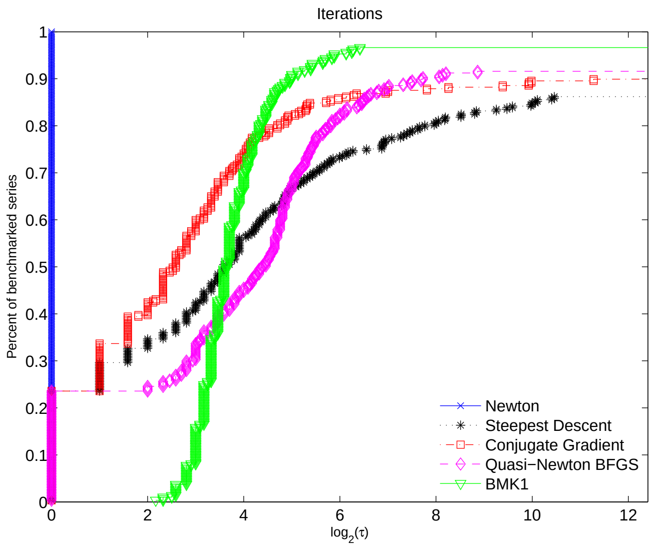


Figure 6: MRTS series: performance profiles for acceptable solutions ($\delta = 10^{-1}$)





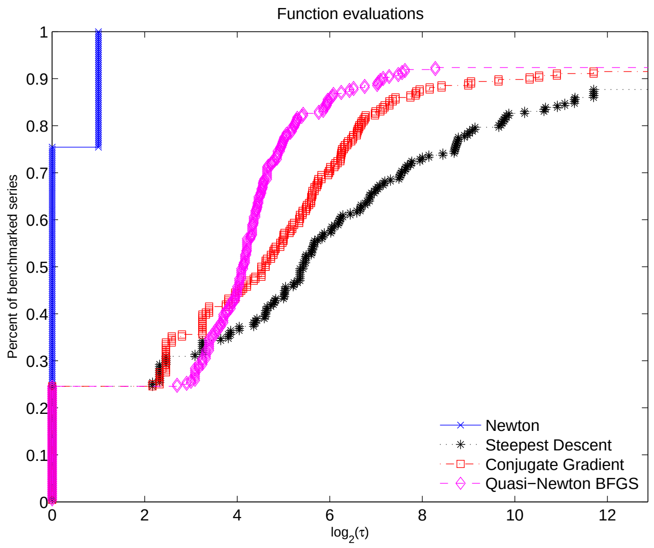


Figure 7: MRTS series: performance profiles for very accurate solutions ($\delta = 10^{-3}$)

References

[1] Andrei, N. (2008), *40 conjugate gradient algorithms for unconstrained optimization. A survey on their definition*, ICI Technical Report No. 13/08. `http://camo.ici.ro/neculai/p13a08.pdf`

[2] Bloem, A., Dippelsman, R., Mæhle, N. (2001), *Quarterly National Accounts Manual. Concepts, Data Sources, and Compilation*, International Monetary Fund, Washington DC.

[3] Boyd, S., Vandenberghe, L. (2004), *Convex Optimization*, Springer, New York.

[4] Bozik, J.E., Otto, M.C. (1988), Benchmarking: Evaluating methods that preserve month-to-month changes. Bureau of the Census - Statistical Research Division, RR-88/07. `http://www.census.gov/srd/papers/pdf/rr88-07.pdf`

[5] Brown, I. (2010), An empirical comparison of constrained optimization methods for benchmarking economic time series. In *JSM 2009 Proceedings, Business and Economic Statistics Section*, 2131–2143.

[6] Causey, B., Trager, M.L. (1981), *Derivation of Solution to the Benchmarking Problem: Trend Revision.* Unpublished research notes, U.S. Census Bureau, Washington D.C.. Available as an appendix in Bozik and Otto (1988).

[7] Cholette, P. (1984), "Adjusting sub-annual series to yearly benchmarks", *Survey Methodology*, **10**, 35–49.

[8] Cholette, P.A., Chhab, N.B. (1991), "Converting aggregates of weekly data into monthly values", *Applied Statistics*, **40**, 411–422.

[9] Cohen, K.J, Müller, W.M., Padberg M.W. (1971), "Autoregressive approaches to disaggregation of time series data", *Applied Statistics*, **20**, 119–129.

[10] Dagum, E.B., Cholette, P.A. (2006), *Benchmarking, Temporal Distribution, and Reconciliation Methods for Time Series.* Springer, New York.

[11] Dai, Y.H., Yuan Y. (1999), "A nonlinear conjugate gradient method with a strong global convergence property", *SIAM Journal on Optimization*, **10**, 177–182.

[12] Denton, F. (1971), "Adjustment of monthly or quarterly series to annual totals: An approach based on quadratic minimization", *Journal of the American Statistical Association*, **66**, 99–102.

[13] Di Fonzo, T., Marini, M. (2010), *Benchmarking and movement preservation. Evidences from real-life and simulated series.* Department of Statistical Sciences, University of Padua, Working Paper Series, n. 14. `http://www.stat.unipd.it/ricerca/fulltext?wp=422`

[14] Di Fonzo, T., Marini, M. (2011), "Simultaneous and Two-step Reconciliation of Systems of Time Series: Methodological and Practical Issues", *Journal of the Royal Statistical Society C*, **60**, 143–164.

[15] Dolan, E.D., Moré, J.J. (2002), "Benchmarking optimization software with performance profiles", *Mathematical Programming*, **91**, 201–213.

[16] Eurostat (2007), “Demetra. Advanced Seasonal Adjustment Interface”. `http://circa.europa.eu/irc/dsis/eurosam/info/data/demetra.htm`

[17] Fagan, J. (1995), *Benchmarking.* Unpublished note (mimeo).

[18] Fletcher, R., Reeves, J. (1964), “Function minimization by conjugate gradients”, *The Computer Journal*, **7**, 149–154.

[19] Fletcher, R. (1987), *Practical methods of optimization vol. 1: Unconstrained optimization.* Wiley, New York.

[20] Fortier, S., Quenneville, B. (2009), Reconciliation and Balancing of Accounts and Time Series. From Concepts to a SAS procedure. In *JSM 2009 Proceedings*, Business and Economic Statistics Section. Alexandria, VA: American Statistical Association, pp. 130–144.

[21] Gilbert, J.C., Nocedal, J.(1992), “Global convergence properties of conjugate gradient methods for optimization”, *SIAM Journal on Optimization*, **2**, 21–42.

[22] Griva, I., Nash, S.G., Sofer, A. (2009), *Linear and Nonlinear Optimization.* SIAM, Philadelphia.

[23] Hager, W.H., Zhang, H. (2006), “A survey of nonlinear conjugate gradient methods”, *Pacific Journal of Optimization*, **2**, 35–58.

[24] Harvill Hood, C.C. (2005), *An empirical comparison of methods for benchmarking seasonally adjusted series to annual totals.* Paper presented at the Workshop on Frontiers in Benchmarking Techniques and their Applications to Official Statistics, Luxembourg, 7-8 April 2005.

[25] Hestenes, M.R., Stiefel, E.L. (1952), “Methods of conjugate gradients for solving linear systems”, *Journal of Research of the National Bureau of Standards*, **49**, 409–436.

[26] Mittelmann, H.D., Pruessner, A. (2006), “A server for automated performance analysis of benchmarking data”, *Optimization Methods and Software*, **21**, 105–120.

[27] Monsour, N.J., Trager, M.L. (1979), Revision and benchmarking of business time series. American Statistical Association, *Proceedings of the Business and Economic Statistics Section*, 333-337.

[28] Nocedal, J., Wright, S. (2006), *Numerical Optimization.* 2nd edition. Springer, New York.

[29] Polak E., Ribière G. (1969), “Note sur la convergence de directions conjugées”, *Revue Francaise d'Informatique et Recherche Opérationelle*, **16**, 35–43.

[30] Polyak T. (1969), “The conjugate gradient method in extremal problems”, *U.S.S.R. Computational Mathematics and Mathematical Physics*, **9**, 94–112.

[31] Powell, M.J.D. (1977), “Restart procedures for the conjugate gradient method”, *Mathematical Programming*, **12**, 241–254.

[32] Powell, M.J.D. (1984), "Nonconvex minimization calculations and the conjugate gradient method". In *Lecture Notes in Mathematics*, vol. 1066, Springer, Berlin, 122–141.

[33] Quenneville, B., Huot, G., Cholette, P., Chiu, K., Di Fonzo, T. (2003), Adjustment of seasonally adjusted series to annual totals. In *Proceedings of Statistics Canada Symposium 2003 - Challenges in Survey Taking for the Next Decade*, 2–12.

[34] Schmidt, M. (2006), The `minFunc` Toolbox for Matlab. `http://www.cs.ubc.ca/~schmidtm`

[35] The MathWorks (2009), *Optimization Toolbox^{TM}4 User's Guide*. Natick, MA.

[36] Titova, N, Findley, D., Monsell, B.C. (2010), Comparing the Causey-Trager method to the multiplicative Cholette-Dagum regression-based method of benchmarking sub-annual data to annual benchmarks. In *JSM 2010 Proceedings, Business and Economic Statistics Section*, 3007–3021.

[37] Trager, M.L. (1982), *Derivation of Solution to the Benchmarking Problem: Relative Revision*. Unpublished research notes, U.S. Census Bureau, Washington D.C.. Available as an appendix in Bozik and Otto (1988).

[38] U.S. Census Bureau (2009), *X-12-ARIMA Reference manual, Version 0.3*. U.S. Census Bureau, U.S. Department of Commerce, Washington D.C.. `http://www.census.gov/srd/www/x12a/`

Acknowledgements

We are grateful to Brian Monsell and Irene Brown, of the U.S. Bureau of the Census. The former made the executable of program `BMK1` available to us, the latter informed us about important characteristics of `BMK1`, not described in the available documentation. We alone are responsible for the errors of the paper. The views expressed herein are those of the authors and should not be attributed to the IMF, its Executive Board, or its management. Financing from MIUR is also gratefully acknowledged by T. Di Fonzo.