Noname manuscript No.
(will be inserted by the editor)

Secant Update Version of Quasi-Newton PSB with Weighted Multisecant Equations

Nicolas Boutet · Rob Haelterman · Joris Degroote

Received: date / Accepted: date

Abstract Quasi-Newton methods are often used in the frame of non-linear optimization. In those methods, the quality and cost of the estimate of the Hessian matrix has a major influence on the efficiency of the optimization algorithm, which has a huge impact for computationally costly problems.

One strategy to create a more accurate estimate of the Hessian consists in maximizing the use of available information during this computation. This is done by combining different characteristics. The Powell-Symmetric-Broyden method (PSB) imposes, for example, the satisfaction of the last secant equation, which is called secant update property, and the symmetry of the Hessian [22].

Imposing the satisfaction of more secant equations should be the next step to include more information into the Hessian. However, Schnabel proved that this is impossible [25]. Penalized PSB (pPSB), works around the impossibility by giving a symmetric Hessian and penalizing the non-satisfaction of the multiple secant equations by using weight factors [13]. Doing so, he loses the secant update property.

In this paper, we combine the properties of PSB and pPSB by adding to pPSB the secant update property. This gives us the Secant Update Penalized PSB (SUpPSB). This new formula that we propose also avoids matrix inversions, which makes it easier to compute. Next to that, SUpPSB also performs globally better compared to pPSB.

Keywords non-linear optimization · quasi-Newton formulae · weighted multiple secant equations · symmetric gradient

Nicolas Boutet · Joris Degroote
Ghent University, Dept. Flow, Heat and Combustion Mechanics, Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium
E-mail: nicolas.boutet@ugent.be

Nicolas Boutet · Rob Haelterman
Royal Military Academy, Dept. Mathematics, Renaissancelaan 30, 1000 Brussels, Belgium

Mathematics Subject Classification (2010) 90C53, 49M15

1 Introduction

The evolution of numerical methods, the development of specific tools and the increasing computational power make it possible to solve heavier and more complex problems in multiple domains. This complexity often increases when different systems interact such as fluid and structure, heat and electricity, heat and chemical reagent.

Being able to solve a given complex problem often leads to a new question: which value of the parameters should we take in order to optimize some objective function? For instance, if we are able to compute the air flow, the deformation and the lift for a drone blade, how should we design the blade in order to minimize the power consumption?

This kind of problem can be expressed as

$$\min_{\mathbf{x}} g(\mathbf{x}) \tag{1.1}$$

with $g : D_F \subset \mathbb{R}^n \to \mathbb{R}$, which is called the objective function.

This leads to the root finding problem

$$\nabla g(\mathbf{x}) = \mathbf{f}(\mathbf{x}) = \mathbf{0} \tag{1.2}$$

There exist plenty of methods for solving this root finding problem. In this paper, we work with methods developed from Broyden's method [5,6,15] by adding some characteristics such as the symmetry of the Hessian of $g(\mathbf{x})$/ Jacobian of $\mathbf{f}(\mathbf{x})$ (Powell-Symmetric-Broyden method or PSB [22,23], generalized PSB [25]) or the least squares approximation of the Hessian of $g(\mathbf{x})$/ Jacobian of $\mathbf{f}(\mathbf{x})$ (Quasi-Newton Least Squares Method [8,14], penalized PSB [13]). The PSB-like methods are described in more detail in section 2.

In our study, we assume the problem has the following characteristics:

(1) The value of the objective function $g(\mathbf{x})$ can be calculated with some code, which can be composed of subproblems that can be solved separately. For instance, one subproblem can describe the lift of a drone propeller blade based on its geometry while another describes the deformation of the blade due to the lift.
(2) The analytic form of $g(\mathbf{x})$ is unknown which prevents the use of Newton's method, for instance.
(3) The gradient of the problem $\nabla g(\mathbf{x})$ can be estimated with specific methods (e.g. with the adjoint state method [9,12,21] or a Monte Carlo estimation in high dimension [20]).
(4) Evaluating $g(\mathbf{x})$ and $\nabla g(\mathbf{x})$ is computationally costly because of the size of the problem (high dimensional) or because of the complexity of the root finding problem even when it is of limited dimension. This means that the required number of evaluations (or 'function calls') to reach convergence is a good proxy of the performance of an algorithm. This also reduces the interest of line-search techniques.

2 Method development

2.1 Maximize used information

In order to find a solution to the root finding problem $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, taking into account the characteristic (2) above, we use Quasi-Newton methods. A step is then given by $\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha_i [B_i]^{-1}\mathbf{f}(\mathbf{x}_i)$. We define a given step of the optimization algorithm (optimization path) by using the subscript i. $\mathbf{f}(\mathbf{x}_i)$ is then the gradient of the objective function calculated at the i-th point of the optimization path. $[B_i]^{-1}$ is the inverse of an approximation of the Jacobian matrix $J_{\mathbf{f}}(\mathbf{x}_i)$ of $\mathbf{f}$ evaluated for $\mathbf{x}_i$. α_i is the step length; its default value is 1 but it can be changed when using line search.

Based on the characteristic (4) above, some methods have been developed that tend to maximize the amount of information that is used at each optimization step. A first idea has been to force the approximation of the Jacobian matrix to fulfill the last secant equation $\mathbf{x}_i = \mathbf{x}_{i-1} + [B_i]^{-1}(\mathbf{f}(\mathbf{x}_i) - \mathbf{f}(\mathbf{x}_{i-1}))$. These methods, for example Broyden's method [5,6,15], are called Secant Update Methods.

Going further, it is possible to impose multiple secant equations $\mathbf{x}_i = \mathbf{x}_{i-k} + [B_i]^{-1}(\mathbf{f}(\mathbf{x}_i) - \mathbf{f}(\mathbf{x}_{i-k}))$ for $k = 1, 2, \ldots, m \leq n$ and $m \leq i$. m being the maximal number of secant equations that are imposed. This equation can also be expressed as $B_{i+1}S_i - Y_i = 0$ where S_i contains in its columns successively the vectors $\mathbf{s}_k = \mathbf{x}_i - \mathbf{x}_{i-k}$ and Y_i the differences $\mathbf{y}_k = \mathbf{f}(\mathbf{x}_i) - \mathbf{f}(\mathbf{x}_{i-k})$. This multisecant construction is the basis, for example, of the Quasi-Newton Least Squares Method and its inverse [8,14].

Instead of working with $[B_i]^{-1}$ being the inverse of an approximation of the Jacobian matrix, one can also work directly with the estimate of inverse of the Jacobian $[H_i]$. Note that while $H = B^{-1}$, its estimate $[H]$ is in most of the cases different than the inverse of the estimate of the Jacobian $[B]^{-1}$. The step is then $\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha_i [H_i]\mathbf{f}(\mathbf{x}_i)$ where $[H_i]$ is the approximation of the inverse of the Jacobian matrix $J_{\mathbf{f}}(\mathbf{x}_i)$ evaluated for $\mathbf{x}_i$.

Methods using formulas based on $[B_i]^{-1}$ are called "direct" methods in the present paper. When the formula uses $[H_i]$, we will call it an "inverse" method.

2.2 PSB-like methods

2.2.1 Broyden's Method

In this paper, we will work with least change methods. This means that the new estimate of the Hessian matrix is chosen to be as close a possible to the previous one. One of the first least change methods has been developed by Broyden in 1965. This is Broyden's first formula (also called Broyden's Good formula) [5,6]. B_{i+1}, which is given in Formula 1, is chosen such that:

$$\begin{aligned} &\arg\min_{B_{i+1}} \tfrac{1}{2} \|B_i - B_{i+1}\|^2_{Fr} \\ &\text{such that} \quad B_{i+1}\mathbf{s}_i - \mathbf{y}_i \quad = 0 \end{aligned}$$

As the conditions $B_{i+1}\mathbf{s}_i - \mathbf{y}_i = 0$ is generally not sufficient to give one single solution, the arg min is used in order to reduce the remaining degrees of freedom by selecting the approximation being the closest to the previous approximation according to a given norm.

Formula 1: Broyden's Good - Direct Broyden

Let $B_i \in \mathbb{R}^{n\times n}$, $\mathbf{y}_i$ and $\mathbf{s}_i \in \mathbb{R}^{n\times 1}$. Let B_{i+1} such that:

- $B_{i+1}\mathbf{s}_i = \mathbf{y}_i$
- $\|B_{i+1} - B_i\|_{Fr}$ is minimal

Then, B_{i+1} is given by:

$$B_{i+1} = B_i + \frac{(\mathbf{y}_i - B_i\mathbf{s}_i)\mathbf{s}_i^T}{\mathbf{s}_i^T\mathbf{s}_i} = B_i + \frac{\mathbf{w}_i\mathbf{s}_i^T}{\mathbf{s}_i^T\mathbf{s}_i}$$

with $\mathbf{w}_i = \mathbf{y}_i - B_i\mathbf{s}_i$.

The second Broyden's method is similar to the first one, but instead of taking the Jacobian closest to the previous one, it takes the inverse of the Jacobian which is the closest to the previous one. The formula is built in the same way as for Broyden's first method. This gives Formula 2.

Formula 2: Broyden's Bad - Inverse Broyden

Let $H_i \in \mathbb{R}^{n\times n}$, $\mathbf{y}_i$ and $\mathbf{s}_i \in \mathbb{R}^{n\times 1}$. Let H_{i+1} such that:

- $H_{i+1}\mathbf{y}_i = \mathbf{s}_i$
- $\|H_{i+1} - H_i\|_{Fr}$ is minimal

Then, H_{i+1} is given by:

$$H_{i+1} = H_i + \frac{(\mathbf{s}_i - H_i\mathbf{y}_i)\mathbf{y}_i^T}{\mathbf{y}_i^T\mathbf{y}_i} = H_i + \frac{\mathbf{v}_i\mathbf{y}_i^T}{\mathbf{y}_i^T\mathbf{y}_i}$$

with $\mathbf{v}_i = \mathbf{s}_i - H_i\mathbf{y}_i$.

The name "good" and "bad" initially comes from Broyden, because the first one seemed to give better results [4]. However, depending on the problem, the "bad" method can give better results than the "good" one. Both are used in more recent applications.

2.2.2 Powell-Symmetric-Broyden (PSB)

When trying to solve the optimization problem (1.1), we can use one extra piece of information. Indeed, the Jacobian of (1.2) is in fact the Hessian of (1.1). So, if the function is of class $\mathcal{C}^2$, we know that the Jacobian matrix is symmetric and its estimate should be symmetric too.

If we enforce the symmetry and the satisfaction of the last secant equation, the problem is then to find an update of the estimate of the Hessian such that:

$$\begin{aligned} &\arg\min_{B_{i+1}} \tfrac{1}{2} \|B_i - B_{i+1}\|_{Fr}^2 \\ &\text{such that} \quad B_{i+1}\mathbf{s}_i - \mathbf{y}_i &= 0 \\ &\qquad\qquad\quad B_{i+1} - B_{i+1}^T &= 0 \end{aligned}$$

This symmetric Secant-Update formulation leads to the Powell-Symmetric-Broyden method (PSB) [22,23]. The formula is given as Formula 3.

Formula 3: PSB

Let $B_i \in \mathbb{R}^{n\times n}$, $\mathbf{y}_i$ and $\mathbf{s}_i \in \mathbb{R}^{n\times 1}$. Let B_{i+1} such that:

- $B_{i+1} = B_{i+1}^T$
- $B_{i+1}\mathbf{s}_i = \mathbf{y}_i$
- $\|B_{i+1} - B_i\|_{Fr}$ is minimal

Then, B_{i+1} is given by:

$$B_{i+1} = B_i + \frac{\mathbf{w}_i\mathbf{s}_i^T}{\mathbf{s}_i^T\mathbf{s}_i} + \frac{\mathbf{s}_i\mathbf{w}_i^T}{\mathbf{s}_i^T\mathbf{s}_i} - \frac{\mathbf{w}_i^T\mathbf{s}_i}{(\mathbf{s}_i^T\mathbf{s}_i)^2}\mathbf{s}_i\mathbf{s}_i^T$$

with $\mathbf{w}_i = \mathbf{y}_i - B_i\mathbf{s}_i$.

Applying the same development but using $H_i = B_i^{-1}$ instead of B_i leads to the dual (or inverse) version of PSB. To our knowledge, this formula has never been published. However, as the formula is simply obtained by switching $\mathbf{y}_i$ and $\mathbf{s}_i$, we give the formula of IPSB in Formula 4.

2.2.3 Generalized PSB (gPSB)

Similarly to the construction of the Quasi-Newton Least Squares Method starting from Broyden's method [14,15], one can start from the PSB and add the multisecant property. The problem is then to solve:

$$\begin{aligned} &\arg\min_{B_{i+1}} \tfrac{1}{2} \|B_i - B_{i+1}\|_{Fr}^2 \\ &\text{such that } B_{i+1}S_i - Y_i = 0 \\ &\qquad\qquad\quad B_{i+1} - B_{i+1}^T = 0 \end{aligned}$$

Formula 4: IPSB

Let $H_i \in \mathbb{R}^{n\times n}$, $\mathbf{y}_i$ and $\mathbf{s}_i \in \mathbb{R}^{n\times 1}$. Let H_{i+1} such that:
- $H_{i+1} = H_{i+1}^T$
- $H_{i+1}\mathbf{y}_i = \mathbf{s}_i$
- $\|H_{i+1} - H_i\|_{Fr}$ is minimal

Then, H_{i+1} is given by:

$$H_{i+1} = H_i + \frac{\mathbf{v}_i\mathbf{y}_i^T}{\mathbf{y}_i^T\mathbf{y}_i} + \frac{\mathbf{y}_i\mathbf{v}_i^T}{\mathbf{y}_i^T\mathbf{y}_i} - \frac{\mathbf{v}_i^T\mathbf{y}_i}{(\mathbf{y}_i^T\mathbf{y}_i)^2}\mathbf{y}_i\mathbf{y}_i^T$$

with $\mathbf{v}_i = \mathbf{s}_i - H_i\mathbf{y}_i$.

This has been done by Schnabel who called his method generalized PSB or gPSB[25]:

$$\begin{aligned} B_{i+1} = B_i + Y_iS_i^+ - B_iS_iS_i^+ + (S_i^+)^TY_i - (S_i^+)^TS_i^TB_i^T \\ - (S_i^+)^TY_i^TS_iS_i^+ + (S_i^+)^TS_i^TB_i^TS_iS_i^+ \end{aligned}$$

In this formula, we used $S_i^+ = (S_i^TS_i)^{-1}S_i^T$.

Schnabel has however proven that his formula only exists if $Y_i^TS_i$ is symmetric. It is the case, for example, when the equation (1.2) is quadratic but this is a very restrictive condition.

2.2.4 Penalized PSB (pPSB)

As it is, in general, impossible to satisfy the symmetry and the multiple secant equations at the same time, Gratton et al. proposed to force the symmetry and to penalize the non-satisfaction of the secant equations [13]. As mentioned in their introduction, this approach can be used in a stochastic context in which secant equations are enforced on average, which is not the case in our problem setting. They then solved the system

$$\begin{aligned} &\arg\min_{B_{i+1}} \tfrac{1}{2}\|B_i - B_{i+1}\|_{Fr}^2 + \tfrac{1}{2}\sum_{k=1}^{m}\omega_k\|B_{i+1}\mathbf{s}_k - \mathbf{y}_k\|_2^2 \\ &\text{such that } \; B_{i+1} - B_{i+1}^T = 0 \end{aligned}$$

where ω_k ($k = 1, \dots, m$ with $m \le i$) are the positive weights applied to the non-satisfaction of the multiple secant equations and have to be chosen by the user.

The formula they found is given in Formula 5. Here again, it is possible to develop an inverse version for the update of $H_i = B_i^{-1}$ instead of B_i and this leads to the formula of IpPSB in Formula 6.

These formulas have, to our eyes, several disadvantages:

- There are two $m \times m$ matrices that have to be inverted.

Formula 5: pPSB

Let $B_i \in \mathbb{R}^{n\times n}$, Y_i and $S_i \in \mathbb{R}^{n\times m}$ with $m \leq n$, $m \leq i$, $\mathbf{s}_k$ is the k-th column of S_i, $\mathbf{y}_k$ the k-th column of Y_i and S_i full-ranked. Let ω_k positive scalar weight parameters for $k = 1, \dots, m$. Let B_{i+1} such that:

- $B_{i+1} = B_{i+1}^T$
- $\|B_i - B_{i+1}\|_{Fr}^2 + \sum_{k=1}^{m} \omega_k \, \|B_{i+1}\mathbf{s}_k - \mathbf{y}_k\|_2^2$ is minimal

Then, B_{i+1} is given by:

$$B_{i+1} = B_i + W_i(2\Omega^{-1} + S_i^T S_i)^{-1} S_i^T + S_i(2\Omega^{-1} + S_i^T S_i)^{-1} W_i^T + S_i \Omega^{\frac{1}{2}} X_2 \Omega^{\frac{1}{2}} S^T$$

where X_2 solves

$$(I + \Omega^{\frac{1}{2}} S_i^T S_i \Omega^{\frac{1}{2}}) X_2 + X_2 (I + \Omega^{\frac{1}{2}} S_i^T S_i \Omega^{\frac{1}{2}}) = -\Omega^{\frac{1}{2}} S_i^T W_i \Omega^{\frac{1}{2}} X_1 - X_1 \Omega^{\frac{1}{2}} W_i^T S_i \Omega^{\frac{1}{2}}$$

with

- $W_i = Y_i - B_i S_i$
- $\Omega = \text{diag}(\omega_k)$, a diagonal matrix having ω_k as elements
- $X_1 = (2I + \Omega^{\frac{1}{2}} S_i^T S_i \Omega^{\frac{1}{2}})^{-1}$

Formula 6: IpPSB

Let $H_i \in \mathbb{R}^{n\times n}$, Y_i and $S_i \in \mathbb{R}^{n\times m}$ with $m \leq n$, $m \leq i$, $\mathbf{y}_k$ the k-th column of Y_i, $\mathbf{s}_k$ the k-th column of S_i and Y_i full-ranked. Let ω_k positive scalar weight parameters for $k = 1, \dots, m$. Let H_{i+1} such that:

- $H_{i+1} = H_{i+1}^T$
- $\|H_i - H_{i+1}\|_{Fr}^2 + \sum_{k=1}^{m} \omega_k \, \|H_{i+1}\mathbf{y}_k - \mathbf{s}_k\|_2^2$ is minimal

Then, H_{i+1} is given by:

$$H_{i+1} = H_i + V_i(2\Omega^{-1} + Y_i^T Y_i)^{-1} Y_i^T + Y_i(2\Omega^{-1} + Y_i^T Y_i)^{-1} V_i^T + Y_i \Omega^{\frac{1}{2}} X_2 \Omega^{\frac{1}{2}} Y_i^T$$

where X_2 solves

$$(I + \Omega^{\frac{1}{2}} Y_i^T Y_i \Omega^{\frac{1}{2}}) X_2 + X_2 (I + \Omega^{\frac{1}{2}} Y_i^T Y_i \Omega^{\frac{1}{2}}) = -\Omega^{\frac{1}{2}} Y_i^T V_i \Omega^{\frac{1}{2}} X_1 - X_1 \Omega^{\frac{1}{2}} V_i^T Y_i \Omega^{\frac{1}{2}}$$

with

- $V_i = S_i - H_i Y_i$
- $\Omega = \text{diag}(\omega_k)$, a diagonal matrix having ω_k as elements
- $X_1 = (2I + \Omega^{\frac{1}{2}} Y_i^T Y_i \Omega^{\frac{1}{2}})^{-1}$

- The solution requires to solve a Lyapunov equation (in order to find the value of X_2) which adds a numerical step in the process. This can be solved, for instance, with the Bartels-Stewart algorithm which has a complexity of $\mathcal{O}(n^3)$ [3,16].

– The formula does not have the Secant Update property as the most recent secant equation may be left unsatisfied because of the use of penalization factors.

2.3 Secant Update penalized PSB

Working on a non-stochastic problem, we want to try to improve the penalized PSB formula by enforcing the most recent secant equation (Secant Update). Due to the numerical limitation of previous formula, we want to find a analytical expression fulfilling the last secant equation. So we start with the following optimization problem:

$$\begin{aligned} \arg\min_{B_{i+1}} \ & \tfrac{1}{2}\left\|B_i - B_{i+1}\right\|_{Fr}^2 + \tfrac{1}{2}\sum_{k=2}^{m}\omega_k \left\|B_{i+1}\mathbf{s}_k - \mathbf{y}_k\right\|_2^2 \\ \text{such that } & B_{i+1} - B_{i+1}^T = 0 \\ & B_{i+1}\mathbf{s}_1 - \mathbf{y}_1 = 0 \end{aligned} \tag{2.1}$$

We take the Lagrangian of the system (2.1):

$$\begin{aligned} \mathcal{L}(B_{i+1}, \boldsymbol{\lambda}, M) = {} & \frac{1}{2}\left\|B_i - B_{i+1}\right\|_{Fr}^2 + \frac{1}{2}\sum_{k=2}^{m}\omega_k \left\|B_{i+1}\mathbf{s}_k - \mathbf{y}_k\right\|_2^2 \\ & + \boldsymbol{\lambda}^T(B_{i+1}\mathbf{s}_1 - \mathbf{y}_1) + \sum_{j=1}^{n-1}\sum_{k=j+1}^{n}\mu_{j,k}(B_{i+1}^{j,k} - B_{i+1}^{k,j}) \end{aligned}$$

where $B_{i+1}^{j,k}$ is the scalar corresponding to the j-th row and k-th column of matrix B_{i+1}.

Taking the partial derivative in function of $B_{i+1}^{j,k}$ ($j = 1, \dots, n;\ k = j+1, \dots, n$), $\boldsymbol{\lambda}$ and $\mu_{j,k}$ ($j = 1, \dots, n-1;\ k = j+1, \dots, n$), we find the following system:

$$\begin{cases} B_{i+1} - B_i + \sum_{k=2}^{m}\omega_k(B_{i+1}\mathbf{s}_k - \mathbf{y}_k)\mathbf{s}_k^T + \boldsymbol{\lambda}\mathbf{s}_1^T + M = 0 & \text{(2.2a)} \\ B_{i+1}\mathbf{s}_1 - \mathbf{y}_1 = 0 & \text{(2.2b)} \\ B_{i+1} - B_{i+1}^T = 0 & \text{(2.2c)} \end{cases}$$

with

$$M_{i,j} = \begin{cases} \mu_{i,j} \ i < j \\ -\mu_{j,i} \ j < i \\ 0 \ i = j \end{cases}$$

Combining (2.2a) and (2.2c), using symmetry of B_{i+1} and anti-symmetry of M, gives

$$\begin{aligned}
0 &= B_{i+1} - B_{i+1}^T \\
&= -B_i + \sum_{k=2}^{m} \omega_k (B_{i+1}\mathbf{s}_k - \mathbf{y}_k)\mathbf{s}_k^T \\
&\quad + \boldsymbol{\lambda}\mathbf{s}_1^T + M + B_i^T - \sum_{k=2}^{m} \omega_k \mathbf{s}_k (\mathbf{s}_k^T B_{i+1}^T - \mathbf{y}_k^T) - \mathbf{s}_1\boldsymbol{\lambda}^T - M^T \\
-2M &= -B_i + \sum_{k=2}^{m} \omega_k (B_{i+1}\mathbf{s}_k - \mathbf{y}_k)\mathbf{s}_k^T + \boldsymbol{\lambda}\mathbf{s}_1^T \\
&\quad + B_i^T - \sum_{k=2}^{m} \omega_k \mathbf{s}_k (\mathbf{s}_k^T B_{i+1}^T - \mathbf{y}_k^T) - \mathbf{s}_1\boldsymbol{\lambda}^T \\
M &= \frac{B_i - B_i^T}{2} + \sum_{k=2}^{m} \omega_k \frac{\mathbf{y}_k\mathbf{s}_k^T - \mathbf{s}_k\mathbf{y}_k^T}{2} \\
&\quad - \sum_{k=2}^{m} \omega_k \frac{B_{i+1}\mathbf{s}_k\mathbf{s}_k^T - \mathbf{s}_k\mathbf{s}_k^T B_{i+1}^T}{2} + \frac{\mathbf{s}_1\boldsymbol{\lambda}^T - \boldsymbol{\lambda}\mathbf{s}_1^T}{2}
\end{aligned}$$

We then substitute M in (2.2a). Defining $\bar{B}_i = \frac{B_i + B_i^T}{2}$, this leads to

$$B_{i+1} = \bar{B}_i - \frac{\mathbf{s}_1\boldsymbol{\lambda}^T + \boldsymbol{\lambda}\mathbf{s}_1^T}{2} - \sum_{k=2}^{m} \omega_k \left(\frac{B_{i+1}\mathbf{s}_k\mathbf{s}_k^T + \mathbf{s}_k\mathbf{s}_k^T B_{i+1}^T}{2} - \frac{\mathbf{y}_k\mathbf{s}_k^T + \mathbf{s}_k\mathbf{y}_k^T}{2} \right) \tag{2.3}$$

We now use equation (2.2b) pre-multipled by $\mathbf{s}_1^T$:

$$\begin{aligned}
0 &= \mathbf{s}_1^T B_{i+1}\mathbf{s}_1 - \mathbf{s}_1^T\mathbf{y}_1 \\
&= \mathbf{s}_1^T \bar{B}_i \mathbf{s}_1 - (\mathbf{s}_1^T\mathbf{s}_1)(\boldsymbol{\lambda}^T\mathbf{s}_1) - \mathbf{s}_1^T\mathbf{y}_1 - \sum_{k=2}^{m} \omega_k \left((\mathbf{y}_1^T\mathbf{s}_k)(\mathbf{s}_k^T\mathbf{s}_1) - (\mathbf{s}_1^T\mathbf{y}_k)(\mathbf{s}_k^T\mathbf{s}_1) \right) \\
(\mathbf{s}_1^T\mathbf{s}_1)(\boldsymbol{\lambda}^T\mathbf{s}_1) &= -\mathbf{s}_1^T\mathbf{y}_1 + \mathbf{s}_1^T\bar{B}_i\mathbf{s}_1 - \sum_{k=2}^{m} \omega_k \left((\mathbf{y}_1^T\mathbf{s}_k)(\mathbf{s}_k^T\mathbf{s}_1) - (\mathbf{s}_1^T\mathbf{y}_k)(\mathbf{s}_k^T\mathbf{s}_1) \right) \\
(\boldsymbol{\lambda}^T\mathbf{s}_1) &= -\frac{\mathbf{s}_1^T\bar{\mathbf{w}}_1}{\mathbf{s}_1^T\mathbf{s}_1} - \sum_{k=2}^{m} \omega_k \left(\frac{(\mathbf{y}_1^T\mathbf{s}_k)(\mathbf{s}_k^T\mathbf{s}_1)}{\mathbf{s}_1^T\mathbf{s}_1} - \frac{(\mathbf{s}_1^T\mathbf{y}_k)(\mathbf{s}_k^T\mathbf{s}_1)}{\mathbf{s}_1^T\mathbf{s}_1} \right)
\end{aligned}$$

In the last step, we have used $\bar{\mathbf{w}}_1 = \mathbf{y}_1 - \bar{B}_i\mathbf{s}_1$.

Equations (2.2b) and (2.2c) can also be combined as $B_{i+1}^T\mathbf{s}_1 = \mathbf{y}_1$. We reuse the last result to replace $\boldsymbol{\lambda}^T\mathbf{s}_1$ and equation (2.3) to replace B_{i+1}.

$$\begin{aligned}
\mathbf{y}_1 &= B_{i+1}\mathbf{s}_1 \\
&= \bar{B}_i{}^T\mathbf{s}_1 - \frac{\boldsymbol{\lambda}(\mathbf{s}_1^T\mathbf{s}_1)}{2} \\
&\quad + \frac{\mathbf{s}_1}{2}\left(\frac{\mathbf{s}_1^T\bar{\mathbf{w}}_1}{\mathbf{s}_1^T\mathbf{s}_1} + \sum_{k=2}^{m}\omega_k\left(\frac{(\mathbf{y}_1^T\mathbf{s}_k)(\mathbf{s}_k^T\mathbf{s}_1)}{\mathbf{s}_1^T\mathbf{s}_1} - \frac{(\mathbf{s}_1^T\mathbf{y}_k)(\mathbf{s}_k^T\mathbf{s}_1)}{\mathbf{s}_1^T\mathbf{s}_1}\right)\right) \\
&\quad - \sum_{k=2}^{m}\omega_k\left(\frac{B_{i+1}\mathbf{s}_k(\mathbf{s}_k^T\mathbf{s}_1)}{2} + \frac{\mathbf{s}_k(\mathbf{s}_k^T\mathbf{y}_1)}{2} - \frac{\mathbf{s}_k(\mathbf{y}_k^T\mathbf{s}_1) + \mathbf{y}_k(\mathbf{s}_k^T\mathbf{s}_1)}{2}\right)
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{\lambda} = -\frac{2\bar{\mathbf{w}}_1}{\mathbf{s}_1^T\mathbf{s}_1} + \frac{\mathbf{s}_1(\mathbf{s}_1^T\bar{\mathbf{w}}_1)}{(\mathbf{s}_1^T\mathbf{s}_1)^2} + \sum_{k=2}^{m}\omega_k\Bigg(&\frac{\mathbf{s}_1(\mathbf{y}_1^T\mathbf{s}_k)(\mathbf{s}_k^T\mathbf{s}_1)}{(\mathbf{s}_1^T\mathbf{s}_1)^2} - \frac{\mathbf{s}_1(\mathbf{s}_1^T\mathbf{y}_k)(\mathbf{s}_k^T\mathbf{s}_1)}{(\mathbf{s}_1^T\mathbf{s}_1)^2} \\
&- \frac{\mathbf{s}_k(\mathbf{s}_k^T\mathbf{y}_1)}{\mathbf{s}_1^T\mathbf{s}_1} - \frac{B_{i+1}\mathbf{s}_k(\mathbf{s}_k^T\mathbf{s}_1)}{\mathbf{s}_1^T\mathbf{s}_1} + \frac{\mathbf{s}_k(\mathbf{y}_k^T\mathbf{s}_1) + \mathbf{y}_k(\mathbf{s}_k^T\mathbf{s}_1)}{\mathbf{s}_1^T\mathbf{s}_1}\Bigg)
\end{aligned}$$

Finally, we can express equation (2.2a) with only known variables, starting from equation (2.3):

$$\begin{aligned}
B_{i+1} &= B_{i+1}^T \\
&= \bar{B}_i - \frac{\mathbf{s}_1\boldsymbol{\lambda}^T + \boldsymbol{\lambda}\mathbf{s}_1^T}{2} - \sum_{k=2}^{m}\omega_k\left(\frac{B_{i+1}\mathbf{s}_k\mathbf{s}_k^T + \mathbf{s}_k\mathbf{s}_k^T B_{i+1}^T}{2} - \frac{\mathbf{y}_k\mathbf{s}_k^T + \mathbf{s}_k\mathbf{y}_k^T}{2}\right) \\
&= \bar{B}_i + \frac{\mathbf{s}_1\bar{\mathbf{w}}_1^T}{\mathbf{s}_1^T\mathbf{s}_1} - \frac{(\bar{\mathbf{w}}_1^T\mathbf{s}_1)\mathbf{s}_1\mathbf{s}_1^T}{2(\mathbf{s}_1^T\mathbf{s}_1)^2} + \frac{\bar{\mathbf{w}}_1\mathbf{s}_1^T}{\mathbf{s}_1^T\mathbf{s}_1} - \frac{(\mathbf{s}_1^T\bar{\mathbf{w}}_1)\mathbf{s}_1\mathbf{s}_1^T}{2(\mathbf{s}_1^T\mathbf{s}_1)^2} \\
&\quad - \sum_{k=2}^{m}\omega_k\left(\frac{(\mathbf{s}_1^T\mathbf{s}_k)(\mathbf{s}_k^T\mathbf{y}_1) - (\mathbf{s}_1^T\mathbf{y}_k)(\mathbf{s}_k^T\mathbf{s}_1)}{(\mathbf{s}_1^T\mathbf{s}_1)^2}\mathbf{s}_1\mathbf{s}_1^T\right) \\
&\quad + \sum_{k=2}^{m}\omega_k\left(\frac{(\mathbf{y}_1^T\mathbf{s}_k)\mathbf{s}_1\mathbf{s}_k^T}{2\mathbf{s}_1^T\mathbf{s}_1} - \frac{(\mathbf{s}_1^T\mathbf{y}_k)\mathbf{s}_1\mathbf{s}_k^T}{2\mathbf{s}_1^T\mathbf{s}_1} - \frac{(\mathbf{s}_1^T\mathbf{s}_k)\mathbf{s}_1\mathbf{y}_k^T}{2\mathbf{s}_1^T\mathbf{s}_1}\right) \\
&\quad + \sum_{k=2}^{m}\omega_k\left(\frac{(\mathbf{s}_k^T\mathbf{y}_1)\mathbf{s}_k\mathbf{s}_1^T}{2\mathbf{s}_1^T\mathbf{s}_1} - \frac{(\mathbf{y}_k^T\mathbf{s}_1)\mathbf{s}_k\mathbf{s}_1^T}{2\mathbf{s}_1^T\mathbf{s}_1} - \frac{(\mathbf{s}_k^T\mathbf{s}_1)\mathbf{y}_k\mathbf{s}_1^T}{2\mathbf{s}_1^T\mathbf{s}_1}\right) \\
&\quad + \sum_{k=2}^{m}\omega_k\left(\frac{B_{i+1}\mathbf{s}_k\mathbf{s}_1^T(\mathbf{s}_k^T\mathbf{s}_1) + (\mathbf{s}_k^T\mathbf{s}_1)\mathbf{s}_1\mathbf{s}_k^T B_{i+1}^T}{2(\mathbf{s}_1^T\mathbf{s}_1)^2}\right) \\
&\quad - \sum_{k=2}^{m}\omega_k\left(\frac{B_{i+1}\mathbf{s}_k\mathbf{s}_k^T + \mathbf{s}_k\mathbf{s}_k^T B_{i+1}^T}{2} - \frac{\mathbf{y}_k\mathbf{s}_k^T + \mathbf{s}_k\mathbf{y}_k^T}{2}\right)
\end{aligned}$$

Simplifying and putting terms together gives:

$$
\begin{aligned}
B_{i+1}X + X^T B_{i+1}^T = \bar{B}_i &+ \frac{\mathbf{s}_1\bar{\mathbf{w}}_1^T + \bar{\mathbf{w}}_1\mathbf{s}_1^T}{\mathbf{s}_1^T\mathbf{s}_1} + \sum_{k=2}^{m}\omega_k\frac{\mathbf{y}_k\mathbf{s}_k^T + \mathbf{s}_k\mathbf{y}_k^T}{2} \\
&- \sum_{k=2}^{m}\omega_k\left(\frac{\left((\mathbf{s}_1^T\mathbf{y}_k) - (\mathbf{y}_1^T\mathbf{s}_k)\right)\mathbf{s}_1\mathbf{s}_k^T}{2\mathbf{s}_1^T\mathbf{s}_1}\right) \\
&- \sum_{k=2}^{m}\omega_k\left(\frac{\left((\mathbf{y}_k^T\mathbf{s}_1) - (\mathbf{s}_k^T\mathbf{y}_1)\right)\mathbf{s}_k\mathbf{s}_1^T}{2\mathbf{s}_1^T\mathbf{s}_1}\right) \\
&- \sum_{k=2}^{m}\omega_k\left(\frac{(\mathbf{s}_1^T\mathbf{s}_k)\mathbf{s}_1\mathbf{y}_k^T + (\mathbf{s}_k^T\mathbf{s}_1)\mathbf{y}_k\mathbf{s}_1^T}{2\mathbf{s}_1^T\mathbf{s}_1}\right) \\
&- \sum_{k=2}^{m}\omega_k\left(\frac{(\mathbf{s}_1^T\mathbf{s}_k)(\mathbf{s}_k^T\mathbf{y}_1) - (\mathbf{s}_1^T\mathbf{y}_k)(\mathbf{s}_k^T\mathbf{s}_1)}{(\mathbf{s}_1^T\mathbf{s}_1)^2}\right)\mathbf{s}_1\mathbf{s}_1^T \\
&- \frac{(\bar{\mathbf{w}}_1^T\mathbf{s}_1) + (\mathbf{s}_1^T\bar{\mathbf{w}}_1)}{2(\mathbf{s}_1^T\mathbf{s}_1)^2}\mathbf{s}_1\mathbf{s}_1^T
\end{aligned}
$$

where

$$
X = \frac{1}{2}\left(I + \sum_{k=2}^{m}\omega_k(\mathbf{s}_k\mathbf{s}_k^T - \frac{\mathbf{s}_k\mathbf{s}_1^T(\mathbf{s}_k^T\mathbf{s}_1)}{\mathbf{s}_1^T\mathbf{s}_1})\right)
$$

On the right side, we have a sum of rank 1 matrices as each term has the form of $\mathbf{uv}^T$.

We can now finally express the formula for Secant Update penalized PSB, which is given in Formula 7. We can also work with $H_{i+1} = B_{i+1}^{-1}$ instead of B_{i+1} and switching Y_i and S_i which will give the dual or inverse formula (Formula 8).

One may believe that there is almost no difference between pPSB and SUpPSB. One can expect that it is possible to simulate the results of SUpPSB by letting the weight of the last equation to be much larger than the other weights in pPSB. However, the calculation of X_1 in pPSB requires the inversion of $X_1^{-1} = (2I + \Omega^{\frac{1}{2}}S_i^T S_i\Omega^{\frac{1}{2}})$. If one wants to simulate the Secant Update property, one should take a large value for ω_1, which is the first element on the diagonal matrix Ω. Looking at the elements of matrix X_1^{-1}, we see that the element at the intersection of its first column and that first row is proportional to ω_1, and that the rest of the matrix does not contain any term in function of ω_1. Taking ω_1 much larger than the other ω_i leads then to a matrix that is ill-conditioned. As we take the inverse of this matrix, this can lead to numerical problems. Even as those problems can be mitigated by preconditioning the matrix, for example with Ruiz's algorithm [24], taking a ratio of several orders of magnitude between the greatest and the smallest coefficient is a risk of losing numerical precision. SUpPSB avoids this risk.

In addition to imposing the most recent equation (Secant Update), we have shown that we have a formulation for SUpPSB where no matrix inversion is

Formula 7: SUpPSB

Let $B_i \in \mathbb{R}^{n\times n}$, Y_i and $S_i \in \mathbb{R}^{n\times m}$ with $m \leq n$, $m \leq i$, $\mathbf{s}_k$ is the k-th column of S_i, $\mathbf{y}_k$ the k-th column of Y_i and S_i full-ranked. Let ω_k positive scalar weight parameters for $k = 2, \ldots, m$. Let B_{i+1} such that:

- $B_{i+1}\mathbf{s}_1 = \mathbf{y}_1$
- $B_{i+1} = B_{i+1}^T$
- $\|B_i - B_{i+1}\|_{Fr}^2 + \sum\limits_{k=2}^{m} \omega_k \|B_{i+1}\mathbf{s}_k - \mathbf{y}_k\|_2^2$ is minimal

Then, B_{i+1} is the solution of the Lyapunov equation :

$$B_{i+1}X_1 + X_1^T B_{i+1}^T = X_2$$

where

$$X_1 = \frac{1}{2}\left(I + \sum_{k=2}^{m} \omega_k (\mathbf{s}_k\mathbf{s}_k^T - \frac{\mathbf{s}_k\mathbf{s}_1^T(\mathbf{s}_k^T\mathbf{s}_1)}{\mathbf{s}_1^T\mathbf{s}_1})\right)$$

and

$$\begin{aligned} X_2 = \bar{B}_i &+ \frac{\mathbf{s}_1\bar{\mathbf{w}}_1^T + \bar{\mathbf{w}}_1\mathbf{s}_1^T}{\mathbf{s}_1^T\mathbf{s}_1} - \frac{(\bar{\mathbf{w}}_1^T\mathbf{s}_1) + (\mathbf{s}_1^T\bar{\mathbf{w}}_1)}{2(\mathbf{s}_1^T\mathbf{s}_1)^2}\mathbf{s}_1\mathbf{s}_1^T + \sum_{k=2}^{m}\omega_k\frac{\mathbf{y}_k\mathbf{s}_k^T + \mathbf{s}_k\mathbf{y}_k^T}{2} \\ &- \sum_{k=2}^{m}\omega_k\left(\frac{\left((\mathbf{s}_1^T\mathbf{y}_k) - (\mathbf{y}_1^T\mathbf{s}_k)\right)\mathbf{s}_1\mathbf{s}_k^T}{2\mathbf{s}_1^T\mathbf{s}_1}\right) - \sum_{k=2}^{m}\omega_k\left(\frac{\left((\mathbf{y}_k^T\mathbf{s}_1) - (\mathbf{s}_k^T\mathbf{y}_1)\right)\mathbf{s}_k\mathbf{s}_1^T}{2\mathbf{s}_1^T\mathbf{s}_1}\right) \\ &- \sum_{k=2}^{m}\omega_k\left(\frac{(\mathbf{s}_1^T\mathbf{s}_k)\mathbf{s}_1\mathbf{y}_k^T + (\mathbf{s}_k^T\mathbf{s}_1)\mathbf{y}_k\mathbf{s}_1^T}{2\mathbf{s}_1^T\mathbf{s}_1}\right) \\ &- \sum_{k=2}^{m}\omega_k\left(\frac{(\mathbf{s}_1^T\mathbf{s}_k)(\mathbf{s}_k^T\mathbf{y}_1) - (\mathbf{s}_1^T\mathbf{y}_k)(\mathbf{s}_k^T\mathbf{s}_1)}{(\mathbf{s}_1^T\mathbf{s}_1)^2}\right)\mathbf{s}_1\mathbf{s}_1^T \end{aligned}$$

with

- $\bar{B}_i = \frac{B_i + B_i^T}{2}$
- $\bar{\mathbf{w}}_i = \mathbf{y}_i - \bar{B}_i\mathbf{s}_i$

needed. This makes an important difference for the numerical evaluation of the Hessian matrix or its inverse.

Note however that we still have a Lyapunov equation to solve in order to find the solution.

3 Numerical experiments

This section is devoted to numerical experiments. Our tests are executed in three phases. In a first phase, we test the reliability and the robustness of the formulas on small dimensional problems. In a second step, we compare the results of the algorithms on a selection of high dimensional problems. Finally, we use the formulas on a real high dimensional application.

Formula 8: ISUpPSB

Let $H_i \in \mathbb{R}^{n\times n}$, S_i and $Y_i \in \mathbb{R}^{n\times m}$ with $m \leq n$, $m \leq i$, $\mathbf{y}_k$ the k-th column of Y_i, $\mathbf{s}_k$ the k-th column of S_i and Y_i full-ranked. Let ω_k positive scalar weight parameters for $k = 2, \ldots, m$. Let H_{i+1} such that:

- $H_{i+1}\mathbf{y}_1 = \mathbf{s}_1$
- $H_{i+1} = H_{i+1}^T$
- $\|H_i - H_{i+1}\|_{Fr}^2 + \sum\limits_{k=2}^{m} \omega_k \|H_{i+1}\mathbf{y}_k - \mathbf{s}_k\|_2^2$ is minimal

Then, H_{i+1} is the solution of the Lyapunov equation :

$$H_{i+1}X_1 + X_1^T H_{i+1}^T = X_2$$

where

$$X_1 = \frac{1}{2}\left(I + \sum_{k=2}^{m} \omega_k (\mathbf{y}_k\mathbf{y}_k^T - \frac{\mathbf{y}_k\mathbf{y}_1^T(\mathbf{y}_k^T\mathbf{y}_1)}{\mathbf{y}_1^T\mathbf{y}_1})\right)$$

and

$$\begin{aligned}X_2 = \bar{H}_i &+ \frac{\mathbf{y}_1\bar{\mathbf{v}}_1^T + \bar{\mathbf{v}}_1\mathbf{y}_1^T}{\mathbf{y}_1^T\mathbf{y}_1} - \frac{(\bar{\mathbf{v}}_1^T\mathbf{y}_1) + (\mathbf{y}_1^T\bar{\mathbf{v}}_1)}{2(\mathbf{y}_1^T\mathbf{y}_1)^2}\mathbf{y}_1\mathbf{y}_1^T + \sum_{k=2}^{m}\omega_k \frac{\mathbf{s}_k\mathbf{y}_k^T + \mathbf{y}_k\mathbf{s}_k^T}{2} \\ &- \sum_{k=2}^{m}\omega_k\left(\frac{\left((\mathbf{y}_1^T\mathbf{s}_k) - (\mathbf{s}_1^T\mathbf{y}_k)\right)\mathbf{y}_1\mathbf{y}_k^T}{2\mathbf{y}_1^T\mathbf{y}_1}\right) - \sum_{k=2}^{m}\omega_k\left(\frac{\left((\mathbf{s}_k^T\mathbf{y}_1) - (\mathbf{y}_k^T\mathbf{s}_1)\right)\mathbf{y}_k\mathbf{y}_1^T}{2\mathbf{y}_1^T\mathbf{y}_1}\right) \\ &- \sum_{k=2}^{m}\omega_k\left(\frac{(\mathbf{y}_1^T\mathbf{y}_k)\mathbf{y}_1\mathbf{s}_k^T + (\mathbf{y}_k^T\mathbf{y}_1)\mathbf{s}_k\mathbf{y}_1^T}{2\mathbf{y}_1^T\mathbf{y}_1}\right) \\ &- \sum_{k=2}^{m}\omega_k\left(\frac{(\mathbf{y}_1^T\mathbf{y}_k)(\mathbf{y}_k^T\mathbf{s}_1) - (\mathbf{y}_1^T\mathbf{s}_k)(\mathbf{y}_k^T\mathbf{y}_1)}{(\mathbf{y}_1^T\mathbf{y}_1)^2}\right)\mathbf{y}_1\mathbf{y}_1^T\end{aligned}$$

with

- $\bar{H}_i = \frac{H_i + H_i^T}{2}$
- $\bar{\mathbf{v}}_i = \mathbf{s}_i - \bar{H}_i\mathbf{y}_i$

3.1 Reliability and robustness

We have chosen to use the testing method proposed by Moré, Burton and Garbow [18]. The purpose is to test the reliability and the robustness of an algorithm by testing it on multiple small dimensional problems. The test functions are commonly used unconstrained test problems with standardized starting points. The used problems are given in Table 1. If needed to be defined, the values of the dimension parameter were chosen according to what exists in the literature [1,7,10,19,26]. The programs are written in Matlab.

Table 1: Test Problems

No	n	Name	No	n	Name
1	2	Rosenbrock function	19	11	Osborne 2 function
2	2	Freudenstein and Roth function	20	9	Watson function
3	2	Powell badly scaled function	21	10	Extended Rosenbrock function
4	2	Brown badly scaled function	22	4	Extended Powell singular function
5	2	Baele function	23	4	Penalty function I
6	2	Jennrich and Sampson function	24	4	Penalty function II
7	3	Helical valley function	25	10	Variably dimensioned function
8	3	Bard function	26	10	Trigonometric function
9	3	Gaussian function	27	(p.m. not avail. in the MATLAB library)	
10	3	Meyer function	28	10	Discrete boundary value function
11	3	Gulf R&D function	29	10	Discrete integral equation function
12	3	Box three-dimensional function	30	10	Broyden tridiagonal function
13	4	Powell singular function	31	10	Broyden banded function
14	4	Wood function	32	10	Linear function - full rank
15	4	Kowalik and Osborne function	33	10	Linear function - rank 1
16	4	Brown and Dennis function	34	10	Linear function - 0 columns & rows
17	5	Osborne 1 function	35	(p.m. not avail. in the MATLAB library)	
18	6	Biggs EXP6 function			

3.1.1 Evaluation method

As we are testing multiple algorithms on multiple test problems, we have to find a way for analyzing the data. We therefor use the perfomance profile proposed by Dolan and Moré [11]. The performance profile for a solver $\rho(\tau) : \tau \in [1 , +\infty [\rightarrow [0, 1]$ is a nondecreasing, piecewise constant function, continuous from the right at each breakpoint. This benchmarking tool makes it possible to graphically compare the cumulative distribution function for a performance metric. We use the number of gradient function calls as the metric (the results for function calls are however similar).

The value of $\rho(1)$ is the probability that the solver will solve the problem more efficiently than the rest of the solvers (with a better value of the metric, less function calls, less time...). On the other side, the value of $\rho(+\infty)$ is the probability that the solver will eventually solve the problem. In one graph, we are able to compare the results of multiple algorithms. If the curve is higher, the algorithm solves more problems. If the curve is more on the left side, the algorithm solves the problems more quickly.

The methods we implemented are PSB, pPSB, SUpPSB with direct updates on the one side and with inverse update on the other side (IPSB, IpPSB, ISUpPSB). The initial approximations are $B_0 = H_0 = I$. We use a backtracking line search to determine step lengths. The initial value of the coefficient α_i in the formula $\mathbf{x}_{i+1} = \mathbf{x}_i - \alpha_i[B_i]^{-1}\mathbf{f}(\mathbf{x}_i)$ is 2. This value is successively divided by 2 until until the Armijo–Goldstein conditions are fulfilled or after dividing 10 times [1, 2].

The iteration is terminated when one of the following conditions is satisfied:

– $||\nabla g(\mathbf{x}_i)||_2 \leq 10^{-6}$

Table 2: List of tested formulas with parameters

Direct Formula		Inverse Formula					
Id	Type	Id	Type	ω_1	ω_2	ω_3	ω_4
A	PSB	a	IPSB	-	-	-	-
B	pPSB	b	IpPSB	1E+00	-	-	-
C	pPSB	c	IpPSB	1E+03	-	-	-
D	pPSB	d	IpPSB	1E+03	1E+02	-	-
E	pPSB	e	IpPSB	1E+03	1E+01	-	-
F	pPSB	f	IpPSB	1E+03	1E+02	1E+01	-
G	pPSB	g	IpPSB	1E+03	1E+01	1E-01	-
H	pPSB	h	IpPSB	1E+03	1E+02	1E+01	1E+00
I	pPSB	i	IpPSB	1E+03	1E+01	1E-01	1E-03
J	pPSB	j	IpPSB	1E+06	1E+02	-	-
K	pPSB	k	IpPSB	1E+09	1E+02	-	-
L	pPSB	l	IpPSB	1E+06	1E+00	-	-
M	pPSB	m	IpPSB	1E+09	1E+00	-	-
N	pPSB	n	IpPSB	1E+06	1E+02	1E+01	-
O	pPSB	o	IpPSB	1E+09	1E+02	1E+01	-
P	SUpPSB	p	ISUpPSB	-	1E+02	-	-
Q	SUpPSB	q	ISUpPSB	-	1E+01	-	-
R	SUpPSB	r	ISUpPSB	-	1E-05	1E-07	-
S	SUpPSB	s	ISUpPSB	-	1E-02	1E-04	-
T	SUpPSB	t	ISUpPSB	-	1E-03	1E-06	-
U	SUpPSB	u	ISUpPSB	-	1E+01	1E-01	-
V	SUpPSB	v	ISUpPSB	-	1E-05	1E-07	1E-09
W	SUpPSB	w	ISUpPSB	-	1E-02	1E-04	1E-06
X	SUpPSB	x	ISUpPSB	-	1E-03	1E-06	1E-09
Y	SUpPSB	y	ISUpPSB	-	1E+01	1E-01	1E-03

– Or after 5000 gradient function calls (this means that the function call within the line search are not counted)

There are, to our knowledge, no published numerical results about pPSB, nor about the choice of the weight parameters. Our goal is not to make a deep analysis of the best parameters combinations. Therefore, we present a selection of parameter combinations that represent the behavior of the algorithms. In our first analysis, we limited the number of considered secant equations up to the fourth one. The overview of used formulas is given in Table 2. For pPSB, we have tested two different evolution profiles for ω_k: in the first one, the value of ω_k decreases linearly (pPSB linear, formulas B to I); the second one is similar except that the value of ω_1 has been multiplied by 10^3 (pPSB non linear, formulas J to O), which corresponds to a formula where more importance is given to respecting the most recent secant equation.

Note that we also tested the same formulas for other values of ω_k (by multiplying every ω_k by 10^3 and 10^{-3}). The results were not exactly the same but the conclusions are similar.

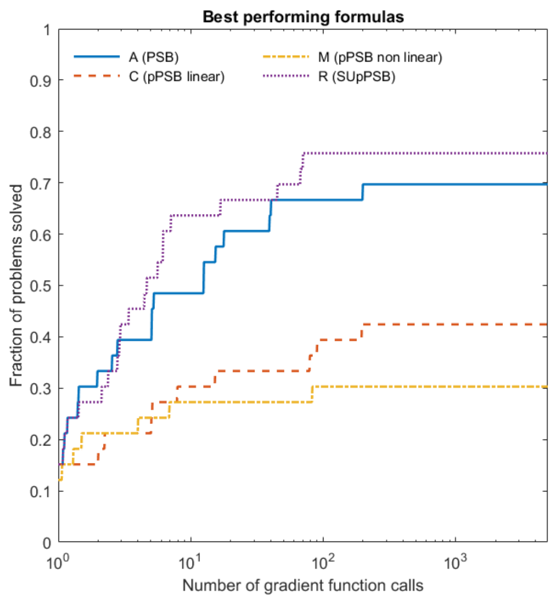

Fig. 1: Best Performance Profiles for direct formulas

3.1.2 Results

The best performance profiles for the direct formulas are given in Figure 1. Out of our experiments, it is apparent that the value of ω_1 in pPSB is of great importance. Its value must be high enough compared to the other ω_i in order to increase the probability of convergence. This can also be shown with the good results of SUpPSB. Indeed, the Secant Update property of SUpPSB can been seen as the case where $\omega_1 = \infty$.

The value of the following ω_i seems to be of less importance for both the old penalized method and the new Secant Update method. However, we observe that the ratio between two successive weight parameters has a clear impact. If the order of magnitude of two successive ω_i is too close, adding extra secant equations leads to worse results. On the other hand, if the difference of the order of magnitude is high enough, the addition of older secant equations improves the results. This could be interpreted the following way. Adding an extra secant equation comes down to using more information which improves the efficiency of the algorithm. However, if one gives too much importance/weight to information coming from older secant equations compared to the more recent secants, we lose some part of the most recent information.

For the inverse update formulas (Figure 2), using the same weight values leads to less obvious results for IpPSB and ISUpPSB. Indeed none of the

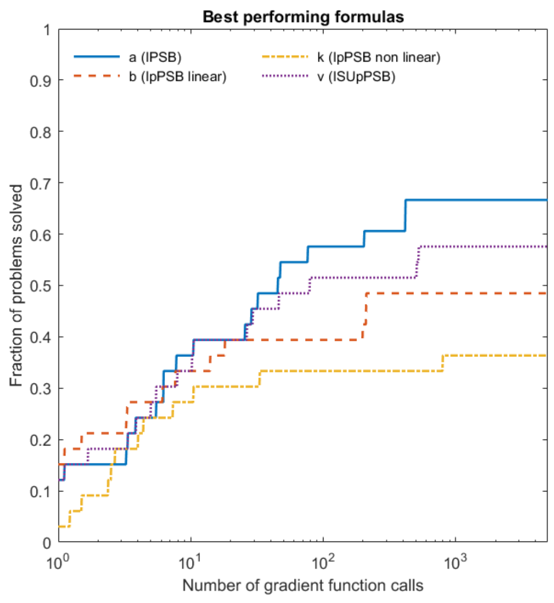

Fig. 2: Best Performance Profiles for inverse formulas

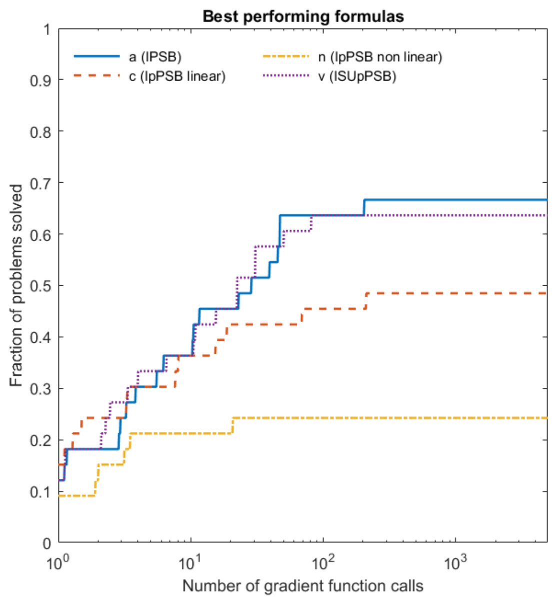

Fig. 3: Best Performance Profiles for inverse formulas, multiplying ω_i by 10^{-3}

parameter combinations shows clearly better results than the standard IPSB. Figure 3 was built by multiplying every ω_i from Table 2 by 10^{-3}. This change leads to results that are more similar to the previous results. We note however that this change creates some very small value of ω_i which could be numerically not suitable because of potential rounding errors or computation of numerical values that are negligible.

Finally, the curves for the direct formulas are globally better than for the inverse update. This means that the algorithm converges less often and less quickly for inverse formulas. But, on the same way that the Broyden's "good" formula is not really better than his "bad" formula, we could simply conclude that our choice for weights is probably not optimal. As mentioned earlier, because of the lack of previous results [13], we did not make a deep analysis of the best parameter combinations.

3.2 Higher dimensions

The test problems used in the previous section are interesting to test the robustness but clearly not high dimensional problems. In this second phase, we will test our formulas on higher dimensional problems.

3.2.1 Evaluation method

While the first problems in Table 1 only exist for a given limited value of n, the last problems of the list also exist in higher dimensions. So it is possible to use them to test the algorithm in higher dimensions. Table 3 gives the problems used for this second phase of the tests.

Table 3: Test Problems in higher dimension

No	n	Name	No	n	Name
21	1000	Extended Rosenbrock function	29	1000	Discrete integral equation function
22	1000	Extended Powell singular function	30	1000	Broyden tridiagonal function
23	1000	Penalty function I	31	1000	Broyden banded function
24	1000	Penalty function II	32	1000	Linear function - full rank
25	1000	Variably dimensioned function	33	1000	Linear function - rank 1
26	1000	Trigonometric function	34	1000	Linear function - 0 columns & rows
28	1000	Discrete boundary value function			

The termination conditions are similar to the previous test case but because of the higher dimension and the longer computation time, we have however added one condition:

– $||\nabla g(\mathbf{x}_i)||_2 \leq 10^{-6}$
– Or after 5000 gradient function calls
– Or after a calculation time of maximum 72 hours, due to technical limitations

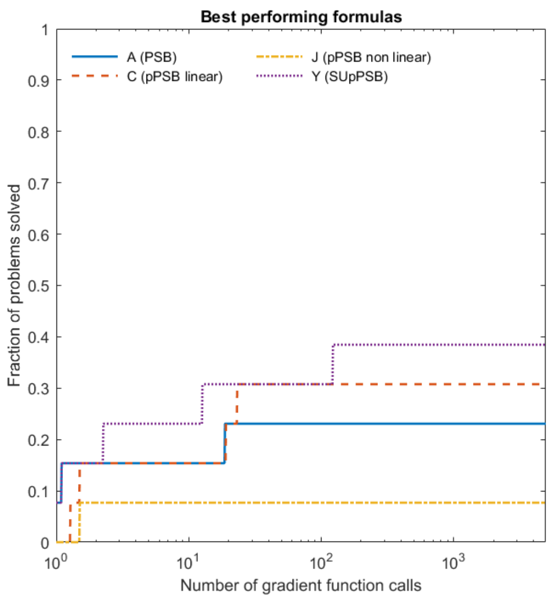

Fig. 4: Best Performance Profiles for direct formulas, with $n = 1000$

An extra relative stopping criterion in the form $||\nabla g(\mathbf{x}_i)||_2 \leq 10^{-6}||\nabla g(\mathbf{x}_0)||_2$ [17] was also tested to try to decrease the number of problems that stopped on the 72 hours limitation, but this did not improve the results.

3.2.2 Results

The performance profile for the direct formulas is given in Figure 4. For these problems, as in the previous section, we see that SUpPSB solves more problems and solves them quicker than the pPSB.

We also observe however that a majority of the problems could not be solved by any of the formulas. This result could induce a bias in our conclusion. To execute a deeper control, we checked which of the stopping criteria was blocking the resolution of the problem. This revealed that pPSB (except for Formulas B and C) diverges for the majority of the problems. On the other hand, PSB and SUpPSB stopped on the maximum number of iterations or time if they did not solve the problem.

Those results confirm the globally better performance of SUpPSB compared to pPSB.

The tests have been executed with and without use of line search. In our application, the line search did not improve the results. When a formula gave good results for a optimization problem, the line search has mainly no impact: the value of $\alpha_i = 1$ satisfied the Armijo conditions [2].

Table 4: List of tested formulas with parameters

Direct Formula					
Id	Type	ω_1	ω_2	ω_3	ω_4
A	PSB	-	-	-	-
Z1	pPSB	1E+15	1E+02	-	-
Z2	pPSB	1E+15	1E+01	-	-
Z3	pPSB	1E+15	1E+02	1E+01	1E+00
Z4	pPSB	1E+15	1E+01	1E-01	1E-03
Z5	SUpPSB	-	1E+02	-	-
Z6	SUpPSB	-	1E+01	-	-
Z7	SUpPSB	-	1E+02	1E+01	1E+00
Z8	SUpPSB	-	1E+01	1E-01	1E-03

On the other side, when the formula could not solve the problem within the limit of 5000 gradient function calls (basically with the pPSB formulas), the line search found no value of α_i satisfied the Armijo conditions. This leads to a number of useless extra function calls and to an extreme reduction of the step length which has a negative influence on the number of steps needed to solve the problem.

Finally, working in higher dimensions also makes it possible to increase the number of secant equations that are used to define the estimate of the gradient (m). In order to observe the influence of a higher value of m, we have extended our formula to $m = 8$ and $m = 16$. We used for these tests $\omega_{i+1} = \frac{\omega_3}{\omega_2}\omega_i$. This test case leads to very few differences compared to the previous case where $m \leq 4$. This is a consequence of our choice of parameters and shows that, when the weights become too small, the extension of the number of fulfilled secant equations does not improve the estimate of the Hessian. In fact, those extra equations are negligible because of the too small value of ω_i.

3.3 Simulate SUpPSB from pPSB

As discussed at the end of section 2.3, one can expect that it is possible to simulate the effect of SUpPSB by letting the weight of the last equation (ω_1) to be much larger than the other weights in pPSB. We therefor used the same test problems (Table 3) and the same stopping conditions as in the previous section. The tested formulas and their parameters are given in Table 4. We have used the same parameters for pPSB and SUpPSB, except for ω_1 which is set to 1E+15 for pPSB.

The results are given in Figure 5. We see that pPSB with a huge value of ω_1 compared to the other ω_i does not perform correctly. It is even less efficient than standard PSB. This confirms the explanation given in section 2.3. The inversion of the ill-conditioned matrix in order to compute the value of X_1 leads to numerical problems in pPSB if the value of ω_1 is too important compared to the other ω_i. Note that the use of Ruiz's algorithm as preconditioner does

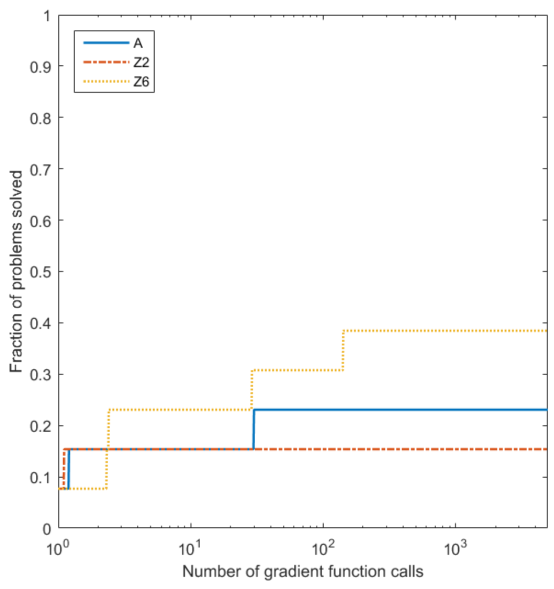

Fig. 5: Comparison Z2 and Z6

not improve the result of pPSB here, which shows that next to the potential problem of ill-conditioning of the matrix, the use of a huge value of ω_1 is still not suitable. By contrast, SUpPSB outperforms PSB and thus pPSB with ω_1=1E+15 in most situations.

3.4 Real application

In the previous section, there are a lot of problems that none of the algorithms could solve. One explanation for that is that those problems are particularly difficult. They were chosen by Moré and co. because they have some 'complicated features' [18]. A lot of real high dimensional problems are however quite smooth. So it could be interesting to test our formulas on smoother problems and on a higher dimension. Therefore, we make a performance test of the formulas on a real application.

3.4.1 Test problem

The application of interest is the identification of the arterial wall's stiffness by comparing the motion of the arterial wall with a reference, possibly obtained from non-invasive imaging [9].

In this problem, the goal is to adjust the stiffness parameters of a one-dimensional elastic tube through which an incompressible fluid flows so that the displacement of the tube wall as a function of time agrees as closely as possible with the displacement data from a non-invasive measurement (or target). The tube is divided into n sections of different but uniform stiffness. As the stiffness parameters are the decision variables for the problem, the dimension of the problem is thus n.

This problem can be reformulated as a minimization problem where the cost function is defined as the difference between the simulation data and the measurement.

3.4.2 Evaluation method

We have executed tests on models of tubes of length $n = 1000$.

The target displacement is a sinusoidal function with a wave length of $2 \times n$. The goal is to minimize the difference between the target displacement and the displacement computed with the decision variables (being the stiffness parameters for each segment of the tube).

Here again, we implemented (I)PSB, (I)pPSB, (I)SUpPSB. We do not use line search as it results in potential extra function calls, which are extremely costly for this application. The value of the weight coefficients ω_j are the same as for the previous tests (Table 2). The initial approximation is $B_0 = H_0 = I$ and the starting point is a stiffness equal to zero. The iteration is terminated when one of the following conditions is satisfied:

- $||\nabla g(\mathbf{x}_i)||_2 \leq 10^{-4}$
- Or after 2000 function calls
- Or after a calculation time of maximum 72 hours, due to technical limitations

3.4.3 Results

As we test the formulas on only one occurrence of one single problem, we can compare the results directly without performance profile. The results for the direct formulas are given in Table 5.

The SUpPSB performs globally better than the pPSB with the selected weight parameter values. Indeed, with most of the pPSB formulas, the algorithm stops early or diverged. The weight combinations for pPSB or SUpPSB that perform the best have a value of parameter ω_1 being high (1E+06, 1E+09). This high value of ω_1 corresponds to trying to satisfy the last secant equation as much as possible. This corresponds precisely to the Secant Update property of the SUpPSB formula.

Looking more in detail to the results of the SUpPSB formulas, we see that many of our examples perform exactly in the same way as PSB. It means, that contrary to the tests problems of the previous section, the value of ω_2, ω_3 and ω_4 had no impact. These weight coefficients seem to be too small for our smoother example. This is confirmed with the good results of the formula used with a higher values of ω_2.

Table 5: Results of the optimization for the model with $n = 1000$

Id	Type	Iter	Remarks				
A	PSB	492		a	IPSB	512	Loc
B	pPSB	-	OOT	b	IpPSB	-	OOT
C	pPSB	-	OOT	c	IpPSB	-	OOT
D	pPSB	-	Div	d	IpPSB	-	OOT
E	pPSB	-	Div	e	IpPSB	-	OOT
F	pPSB	-	Div	f	IpPSB	-	OOT
G	pPSB	-	OOT	g	IpPSB	-	OOT
H	pPSB	-	Div	h	IpPSB	-	OOT
I	pPSB	-	OOT	i	IpPSB	-	OOT
J	pPSB	530		j	IpPSB	-	OOT
K	pPSB	527		k	IpPSB	-	OOT
L	pPSB	-	Div	l	IpPSB	-	OOT
M	pPSB	608		m	IpPSB	-	OOT
N	pPSB	-	Div	n	IpPSB	-	OOT
O	pPSB	7	Loc	o	IpPSB	-	OOT
P	SUpPSB	7	Loc	p	ISUpPSB	185	Loc
Q	SUpPSB	16	Loc	q	ISUpPSB	185	Loc
R	SUpPSB	492		r	ISUpPSB	185	Loc
S	SUpPSB	492		s	ISUpPSB	185	Loc
T	SUpPSB	492		t	ISUpPSB	185	Loc
U	SUpPSB	404		u	ISUpPSB	185	Loc
V	SUpPSB	492		v	ISUpPSB	185	Loc
W	SUpPSB	492		w	ISUpPSB	185	Loc
X	SUpPSB	492		x	ISUpPSB	185	Loc
Y	SUpPSB	473		y	ISUpPSB	185	Loc

Div: Diverged / OOT: Stopped after 72Hr / Loc: Stopped at local optimum

The results of inverse formula also lead to a clear distinction between the IpPSB and the ISUpPSB formulas. Even if none of the formulas does lead to the real optimum, IPSB and ISUpPSB reach some local optimum while IpPSB does not find any solution within the allowed 72 hours.

The fact that none of the formulas reaches the global optimum shows the importance of the weight parameter. Moreover, it seems that the weights must be different for the direct and indirect formulas. However, here again, ISUpPSB leads to better results than IpPSB, which tends to prove that the first secant equation is very important.

Out of these first results, even if a more detailed analysis would be needed in order to confirm this, we can conclude that the (I)SUpPSB formula converges generally more quickly than the (I)pPSB. It also makes it possible to use smaller values for the weight parameters ω_j. The Secant Update property is thus generally an important feature that tends to improve the convergence of the optimization algorithm.

4 Further development

In this section, we discuss some possible further developments or improvements.

First of all, the impact of line search within the algorithm could be analyzed more deeply. Indeed, when solving a system of equations $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ (see equation (1.2)), the complexity of the calculation of the solution can be prohibitive which excludes the use of line search. For an optimization problem, the situation is however different: the computation of the estimation of the gradient of equation (1.1), $\mathbf{f}(\mathbf{x})$, is often more complex than the evaluation of $g(\mathbf{x})$. A simple reasoning based on an estimation of the gradient by using finite differences, shows that the complexity of the calculation of the gradient can be $n+1$ times higher than the calculation of the function $g(\mathbf{x})$ itself. Based on this argument, using a limited number of line search steps within the optimization algorithm makes sense. While we have used a very simple version of line search in this paper, a better line search technique could provide an improvement to the rate of convergence.

Secondly, the analysis of the choice of the weight parameters has to be deepened. It should be possible to define their order of magnitude in function of some characteristics of the problem. For instance, from equations (2.1), we can expect that the order of magnitude of the weight coefficients could be defined in function of the dimension of the problem as the amplitude of the norm will depend on the dimension.

Finally, instead of searching the weight parameter combinations that work well for a given (type) of problem, one could also try to search for an analytical optimum for the coefficients. These optimal values, if they exist, would lead to a weight-free formula, with the values of ω_j being expressed in function of characteristics of the problem ($\mathbf{s}$, $\mathbf{y}$, S, Y...). This solution could make it possible to work without weight parameters which would be a huge advantage as the main difficulty of the (I)SUpPSB is the definition of those weight parameter.

5 Conclusion

We have developed a Secant Update version of the (inverse) penalized PSB formula. This method preserves the symmetry of the gradient estimate and satisfies the most recent secant equation (Secant Update property). As in the (inverse) penalized PBS, the non-satisfaction of the other secant equations penalized by means of weight factors.

The computation of the new formula presents an advantage in comparison to the original (I)pPSB. The new expression for the estimate of the gradient does not require any matrix inversion, which makes it easier to compute and avoids some numerical rounding problems.

Next to this advantage for the computation, (I)SUpPSB performs generally better than (I)pPSB and makes it possible to use limited values of weight factors. Moreover, our results show that the satisfaction of the last secant equation (the Secant Update property) has a significant positive impact on the performance. This is an extra argument for (I)SUpPSB against (I)pPSB.

The choice of the weight parameters is of great importance. Next to the development of the formula of (I)SUpPSB, we have made a first analysis of the choice of the weight parameters. This analysis shows that the ratio between two following coefficients is a key factor for the good convergence of the algorithm. This study must however be conducted further.

For those reasons, SUpPSB and ISUpPSB appear to be interesting alternatives for complex high dimensional optimization problems.

References

1. Ahookhosh, M., Ghaderi, S.: On efficiency of nonmonotone Armijo-type line searches. Applied Mathematical Modelling **43**, 170–190 (2017)
2. Armijo, L.: Minimization of functions having Lipschitz continuous first partial derivatives. Pacific Journal of mathematics **16**(1), 1–3 (1966)
3. Bartels, R.H., Stewart, G.W.: Solution of the matrix equation AX+ XB= C [F4]. Communications of the ACM **15**(9), 820–826 (1972)
4. Broyden, C.: On the discovery of the "good Broyden" method. Mathematical programming **87**(2), 209–213 (2000)
5. Broyden, C.G.: A class of methods for solving nonlinear simultaneous equations. Mathematics of computation **19**(92), 577–593 (1965)
6. Broyden, C.G.: Quasi-Newton methods and their application to function minimisation. Mathematics of Computation **21**(99), 368–381 (1967)
7. Chen, C., Luo, L., Han, C., Chen, Y.: Global convergence of an extended descent algorithm without line search for unconstrained optimization. parameters **1**, 2 (2018)
8. Degroote, J., Bathe, K.J., Vierendeels, J.: Performance of a new partitioned procedure versus a monolithic procedure in fluid–structure interaction. Computers & Structures **87**(11-12), 793–801 (2009)
9. Degroote, J., Hojjat, M., Stavropoulou, E., Wüchner, R., Bletzinger, K.U.: Partitioned solution of an unsteady adjoint for strongly coupled fluid-structure interactions and application to parameter identification of a one-dimensional problem. Structural and Multidisciplinary Optimization **47**(1), 77–94 (2013)
10. Ding, Y., Lushi, E., Li, Q.: Investigation of quasi-Newton methods for unconstrained optimization. Simon Fraser University, Canada (2004)
11. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Mathematical programming **91**(2), 201–213 (2002)
12. Errico, R.M.: What is an adjoint model? Bulletin of the American Meteorological Society **78**(11), 2577–2591 (1997)
13. Gratton, S., Malmedy, V., Toint, P.L.: Quasi-Newton updates with weighted secant equations. Optimization Methods and Software **30**(4), 748–755 (2015)
14. Haelterman, R.: Analytical study of the least squares quasi-Newton method for interaction problems. Ph.D. thesis, Ghent University (2009)
15. Haelterman, R., Bogaers, A., Degroote, J., Boutet, N.: Quasi-Newton methods for the acceleration of multi-physics codes. International Journal of Applied Mathematics **47**(3) (2017)
16. Jarlebring, E.: KTH royal institute of technology in Stockholm, lecture notes: Numerical methods for Lyapunov equations. Url: https://people.kth.se/~eliasj/NLA/matrixeqs.pdf. Last visited on 2018/01/07
17. Kelley, C.T.: Iterative methods for optimization. SIAM (1999)
18. Moré, J.J., Garbow, B.S., Hillstrom, K.E.: Testing unconstrained optimization software. ACM Transactions on Mathematical Software (TOMS) **7**(1), 17–41 (1981)

19. Neumaier, A.: Universität Wien , personnal webpage: Global optimization test problems. Url: http://www.mat.univie.ac.at/~neum/glopt/test.html & http://www.mat.univie.ac.at/~neum/glopt/bounds.html. Last visited on 2018/02/04
20. Patelli, E., Pradlwarter, H.J.: Monte Carlo gradient estimation in high dimensions. International journal for numerical methods in engineering **81**(2), 172–188 (2010)
21. Plessix, R.E.: A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. Geophysical Journal International **167**(2), 495–503 (2006)
22. Powell, M.J.: A new algorithm for unconstrained optimization. Nonlinear programming pp. 31–65 (1970)
23. Rheinboldt, W.C.: University of Pittsburgh, lecture notes: Quasi-Newton methods. Url: https://www-m2.ma.tum.de/foswiki/pub/M2/Allgemeines/SemWs09/quasi-newt.pdf. Last visited on 2018/01/07
24. Ruiz, D.: A scaling algorithm to equilibrate both rows and columns norms in matrices. Tech. rep., CM-P00040415 (2001)
25. Schnabel, R.B.: Quasi-Newton methods using multiple secant equations. Tech. rep., DTIC Document (1983)
26. Zhang, J., Xu, C.: Properties and numerical performance of quasi-Newton methods with modified quasi-Newton equations. Journal of Computational and Applied Mathematics **137**(2), 269–278 (2001)