Sequential Projected Newton method for regularization of nonlinear least squares problems

To cite this article: J Cornelis and W Vanroose 2021 J. Phys. Commun. 5 095009

View the article online for updates and enhancements.
Sequential Projected Newton method for regularization of nonlinear least squares problems

J Cornelis* and W Vanroose
Department of Mathematics, University of Antwerp, 2020 Antwerp, Belgium
E-mail: jeffrey.cornelis@uantwerp.be

Keywords: projected Newton method, inverse problem, regularization, constrained optimization, Talbot-Lau phase contrast imaging

Abstract
We develop a computationally efficient algorithm for the automatic regularization of nonlinear inverse problems based on the discrepancy principle. We formulate the problem as an equality constrained optimization problem, where the constraint is given by a least squares data fidelity term and expresses the discrepancy principle. The objective function is a convex regularization function that incorporates some prior knowledge, such as the total variation regularization function. Using the Jacobian matrix of the nonlinear forward model, we consider a sequence of quadratically constrained optimization problems that can all be solved using the Projected Newton method. We show that the solution of such a quadratically constrained sub-problem results in a descent direction for an exact merit function. This merit function can then be used to describe a formal line-search method. We also formulate a slightly more heuristic approach that simplifies the algorithm and allows for an inexact solution of the sequence of sub-problems. We illustrate the behavior of the algorithm using a number of numerical experiments, with Talbot-Lau x-ray phase contrast imaging as the main application. The numerical experiments confirm that the quadratically constrained sub-problems need not be solved with high accuracy in early iterations to make sufficient progress towards the solution. In addition, we show that the proposed method is able to produce reconstructions of similar quality compared to other state-of-the-art approaches with a significant reduction in computational time.

1. Introduction
In this manuscript we consider the regularized nonlinear least squares problem

$$\min_{x \in \mathbb{R}^n} \Psi(x) \text{ subject to } \frac{1}{2} ||s(x) - b||^2 \leq \frac{\sigma^2}{2}$$

(1)

where $\Psi: \mathbb{R}^n \to \mathbb{R}$ is a convex twice continuously differentiable function. The function $s: \mathbb{R}^n \to \mathbb{R}^m$ with $m \geq n$ models some nonlinear forward operation and $b \in \mathbb{R}^m$ is some data contaminated with measurement errors or noise. Here, $\sigma$ is an estimate of the noise-level, i.e. $\sigma \approx ||b_{\text{ex}} - b||_2$, for some ‘exact’ data $b_{\text{ex}}$. In the following we assume that the unconstrained solution of the least squares problem is below the noise-level, i.e. that $\min_{x \in \mathbb{R}^n} ||s(x) - b|| \leq \sigma$. This implies that the constraint in (1) is feasible and thus there exists at least one local solution.

Furthermore, we assume $s(x)$ has component functions $s_i(x): \mathbb{R}^n \to \mathbb{R}$ for $i = 1, \ldots, m$ that are all continuously differentiable, such that the Jacobian matrix

$$J(x) = \left( \frac{\partial s_i(x)}{\partial x_j} \right)_{ij} = \begin{pmatrix} \frac{\partial s_1(x)}{\partial x_1} & \cdots & \frac{\partial s_1(x)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial s_m(x)}{\partial x_1} & \cdots & \frac{\partial s_m(x)}{\partial x_n} \end{pmatrix} \in \mathbb{R}^{m \times n}$$
is well-defined. Let us denote the constraint function as \( c(x) = \frac{1}{2} \| s(x) - b \|^2 - \frac{x^T z}{2} \). It is an easy calculation to verify that the gradient of the constraint function is given by \( \nabla c(x) = f(x)^T (s(x) - b) \). Note that the constraint function \( c(x) \) is not necessarily convex.

In most cases of interest we can show that the inequality constraint in (1) becomes active for the solution \( x^* \) and can thus be replaced by an equality constraint.

**Lemma 1.1.** Suppose \( x^* \) is a local solution of (1) with \( \nabla \Psi(x^*) \neq 0 \) and \( \nabla c(x^*) = 0 \). Then \( \| s(x^*) - b \| = \sigma \) and there exists a unique Lagrange multiplier \( \lambda^* > 0 \) such that

\[
\begin{align*}
\nabla \Psi(x^*) + \lambda^* \nabla c(x^*) &= 0, \\
\lambda^* c(x^*) &= 0, \\
c(x^*) &= 0.
\end{align*}
\]

**Proof.** The linear independence constraint qualification (LICQ) \([1]\) trivially holds for \( x^* \) since we only have one constraint and we assume that \( \nabla c(x^*) = 0 \).\(^1\) Hence, there exists a Lagrange multiplier \( \lambda^* \in \mathbb{R} \) such that the Karush–Kuhn–Tucker or KKT conditions are satisfied:

\[
\begin{align*}
\nabla \Psi(x^*) + \lambda^* \nabla c(x^*) &= 0, \\
\lambda^* c(x^*) &= 0, \\
c(x^*) &= 0.
\end{align*}
\]

Suppose that \( \| s(x^*) - b \| = \sigma \), i.e. \( c(x^*) = 0 \), then it follows from the complementarity condition that \( \lambda^* = 0 \). This in turn means that \( \nabla \Psi(x^*) = 0 \) which contradicts our assumption that \( \nabla \Psi(x^*) \neq 0 \). Hence, it follows that \( \lambda^* > 0 \) and \( \| s(x^*) - b \| = \sigma \) and that \( (x^*, \lambda^*) \) satisfies (2). Uniqueness of the Lagrange multiplier easily follows from the fact \( \nabla c(x^*) \neq 0 \).

The conditions in theorem 1.1 are very mild since \( \nabla \Psi(x^*) = 0 \) means that \( x^* \) is an unconstrained minimizer of the convex regularization function. Similarly we have that \( \nabla c(x^*) = 0 \) expresses the first order necessary optimality condition of the unconstrained optimization problem \( \arg \min_{x \in \mathbb{R}^n} |s(x) - b| \).

Note that the equations in (2) are precisely the KKT conditions of the equality constrained problem

\[
\min_{x \in \mathbb{R}^n} \Psi(x) \text{ subject to } \frac{1}{2} \| s(x) - b \|^2 = \frac{\sigma^2}{2}.
\]

(3)

Here, the constraint expresses the discrepancy principle \([2]\), which basically says that a point \( x \) that satisfies \( \| s(x) - b \| < \sigma \) is expected to be ‘over-fitted’ to the noisy data \( b \). Among all possible solutions \( x \) that fit data \( b \) ‘well enough’ without being over-fitted, the one that minimizes the regularization function \( \Psi(x) \) is chosen.

The equality constrained optimization problem (3) is closely related to the (unconstrained) regularized nonlinear least squares problem

\[
\min_{x \in \mathbb{R}^n} \frac{1}{2} \| s(x) - b \|^2 + \alpha \Psi(x),
\]

(4)

with fixed regularization parameter \( \alpha > 0 \). The first order optimality conditions of (4) are given by

\[
\frac{f(x^*)^T (s(x^*) - b) + \alpha \nabla \Psi(x^*)}{\nabla c(x^*)} = 0.
\]

Any point \( x^* \) that satisfies these equations also satisfies the first equation of the KKT conditions (2) of the equality constrained optimization problem (3) with \( \lambda^* = 1/\alpha \). Hence, solving (3) can be seen as an approach to simultaneously solve the regularized nonlinear least squares problem (4) and find the corresponding regularization parameter \( \alpha \) such that the discrepancy principle is satisfied. Other well-known parameter choice methods include generalized cross validation and the L-curve criterion. We refer to [2] for more general information on the different parameter choice methods.

In this work we consider a novel approach to compute a solution \((x^*, \lambda^*)\) to the nonlinear system of equations (2) by solving a sequence of sub-problems, by using the linear approximation \( s(x + p) \approx s(x) + f(x)p \) for the constraint in (3) for some \( x \in \mathbb{R}^n \). By Taylor’s theorem we have

\[
s(x + p) = s(x) + \int_0^1 f(x + tp)p \, dt.
\]

\( \alpha \) LICQ is an assumption that ensures that the Karush–Kuhn–Tucker conditions are necessary conditions for local optimality of a constrained optimization problem, see definition 12.4 and theorem 12.1 in [1].
Hence it follows that
\[
\|s(x) + J(x)p - s(x + p)\| = \|J(x)p - \int_0^1 J(x + tp)p dt\| \\
= \|\int_0^1 (J(x) - J(x + tp))p dt\| \\
\leq \|p\| \max_{t \in [0,1]} \|J(x) - J(x + tp)\| = o(\|p\|).
\] (5)

When \(f(x)\) is Lipschitz continuous the bound is replaced by \(O(\|p\|^2)\).

After some calculations it follows that we have the following equivalent expressions:
\[
\frac{1}{2} \|J(x)p + s(x) - b\|^2 = \frac{\sigma^2}{2} \iff c(x) + \nabla c(x)^T p + \frac{1}{2} p^T J(x)^T J(x)p = 0.
\]

Hence it is clear that taking a linear approximation of the forward model \(s(x + p)\) corresponds to taking a quadratic model of the constraint function \(c(x + p)\). Alternatively we could have immediately considered a quadratic model of \(c(x + p)\) using Taylor’s theorem:
\[
c(x + p) \approx c(x) + \nabla c(x)^T p + \frac{1}{2} p^T \nabla^2 c(x)p.
\]

However, since \(c(x)\) is in general non-convex we have that the Hessian is possibly indefinite, which is undesirable for optimization algorithms. Moreover, for a least squares function we know that the matrix \(J(x)^T J(x)\) is an approximation of the true Hessian \(\nabla^2 c(x)\), see for instance chapter 10 in [1]. This approximation is for instance used in the Gauss-Newton algorithm for solving a least squares minimization problem.

The main idea of the method proposed in this manuscript is to solve a sequence of problems of the form
\[
\min_{p \in \mathbb{R}^q} \Psi(x + p) \text{ subject to } \frac{1}{2} \|J(x)p + s(x) - b\|^2 = \frac{\sigma^2}{2} \tag{6}
\]
for a given iterate \(x \in \mathbb{R}^n\). The KKT conditions for this optimization problem are given by
\[
\begin{aligned}
\{ \nabla \Psi(x + p) + \lambda (\nabla c(x) + J(x)^T J(x)p) & = 0, \\
\{ c(x) + \nabla c(x)^T p + \frac{1}{2} p^T J(x)^T J(x)p & = 0,
\end{aligned} \tag{7a}
\]
\[
\begin{aligned}
\{ \nabla \Psi(x)^T p + \frac{1}{2} p^T \nabla^2 \Psi(x)p & \text{ subject to } \frac{1}{2} \|J(x)p + s(x) - b\|^2 = \frac{\sigma^2}{2}. \tag{8}
\end{aligned}
\]

This idea forms the basis of one of the reference methods that we consider in this manuscript. In this case we get an approach that closely resembles the one taken in [3], namely the Sequential Quadratically Constrained Quadratic programming (SQCQP) method. The SQCQP method is slightly more general in the sense that they work with possibly multiple constraint functions \(c_i(x) : \mathbb{R}^n \to \mathbb{R}\) for \(i = 1, \ldots, m\). However, they assume convexity of these functions and then use the true Hessians \(\nabla^2 c_i(x)\) in the quadratic model. With the numerical experiments, see section 5, it will become clear that in the context of the current manuscript it is better to solve a sequence of problems of the form (6). Hence, we focus our presentation on this case. However, many of the results presented here can be modified in case we solve a sequence of problems (8).

A second reference method that we consider for the numerical experiments is based on the approximation \(J(x)^T J(x) \approx \nabla^2 c(x)\). The algorithm can be seen as a type of Gauss-Newton method and is able to solve the regularized nonlinear inverse problem with fixed regularization parameter \(\alpha > 0\) shown in (4). The linear systems of equations that appear in each iteration can be solved using the Conjugate Gradient method [4].

Note that even when (2) has a solution, this does not necessarily imply that (7) has a solution for all \(x\).

However, using the implicit function theorem, we can show that this sub-problem has a solution for all \(x\) in a neighborhood of the solution \(x^*\). The implicit function theorem can be formulated as follows:

**Theorem 1.2.** Let \(\Phi : \mathbb{R}^n \times \mathbb{R}^q \to \mathbb{R}^p\) be a continuously differentiable function and let \(\mathbb{R}^n \times \mathbb{R}^q\) have coordinates \((x, y)\). Suppose \((x^*, y^*)\) is a point satisfying \(\Phi(x^*, y^*) = 0\). If the Jacobian matrix of \(\Phi\) with respect to \(y\) is nonsingular in \((x^*, y^*)\), i.e. if the matrix \(J_b(x^*, y^*) = \left( \frac{\partial \Phi(x^*, y^*)}{\partial y} \right)_{ij} \in \mathbb{R}^{p \times q}\) is invertible, then there exists some open set \(X \subset \mathbb{R}^n\) containing \(x^*\) such that there exists a unique continuously differentiable function \(g : X \to \mathbb{R}^q\) such that \(g(x^*) = y^*\) and \(\Phi(x, g(x)) = 0\) for all \(x \in X\).
Proof. See for instance [5, 6].

Lemma 1.3. Let \((x^*, \lambda^*)\) be a KKT point (2) with \(\nabla c(x^*) = 0\) and suppose the matrix \(\nabla^2 \Psi(x^*) + \lambda^* \nabla J(x^*)J(x^*)\) is positive definite. Then there exist an open open ball \(B_\rho(x^*)\) with center \(x^*\) and radius \(\rho > 0\) such that for all \(x \in B_\rho(x^*)\) there exists a solution \((p, \lambda)\) for (7) with \(\lambda > 0\).

Proof. To prove this claim we apply the implicit function theorem to the function \(\Phi: \mathbb{R}^n \times \mathbb{R}^{n+1} \to \mathbb{R}^{n+1}\) with

\[
\Phi(x, p, \lambda) = \left( \nabla \Psi(x) + p + \lambda \nabla J(x)J(x)p + \frac{1}{2} p^T J(x)^T J(x)p, \nabla c(x) \right).
\]

Obviously we have \(\Phi(x^*, 0, \lambda^*) = 0\) since \((x^*, \lambda^*)\) solves (2). Moreover we have that the Jacobian of \(\Phi(x, p, \lambda)\) with respect to \((p, \lambda)\) is given by

\[
J_\Phi(x, p, \lambda) = \begin{pmatrix}
\nabla^2 \Psi(x) + \lambda^* \nabla J(x)^T J(x) & 0 \\
\nabla c(x)^T J(x) + \nabla c(x)^T J(x) & 0
\end{pmatrix}
\]

Evaluating this matrix in \((x^*, 0, \lambda^*)\) gives us

\[
J_\Phi(x^*, 0, \lambda^*) = \begin{pmatrix}
\nabla^2 \Psi(x^*) + \lambda^* \nabla J(x^*)^T J(x^*) & \nabla c(x^*) \\
\nabla c(x^*)^T J(x^*) & 0
\end{pmatrix}
\]

which is non-singular since \(\nabla^2 \Psi(x^*) + \lambda^* \nabla J(x^*)^T J(x^*)\) is positive definite and \(\nabla c(x^*) = 0\). Hence, from theorem 1.2 it follows that there exists an open set \(X \subset \mathbb{R}^n\) containing \(x^*\) such that there exists a unique continuously differentiable function \(g: X \to \mathbb{R}^{n+1}\) such that \(g(x^*) = (0, \lambda^*)\) and \(\Phi(x, g(x)) = 0\) for all \(x \in X\). It is now clear that \(g(x) = (p, \lambda)\) is a solution for (7). Furthermore, by continuity of \(g(x)\) and the fact that \(\lambda^* > 0\) we can choose the any radius \(\rho > 0\) small enough such that \(B_\rho(x^*) \subset X\) and \(\lambda > 0\).

The rest of the paper is organized as follows. In section 2 we derive the Sequential Projected Newton method, which is based on (approximately) solving a sequence of problems of the form (6). In section 3 we give some comments on related work and describe two reference methods. Next, in section 4, we introduce the application that we consider for our numerical experiments, namely Talbot-Lau x-ray phase contrast imaging. In section 5 we perform a number of experiments with the proposed algorithm and compare it with the two reference methods described in section 3. First, we show that the sub-problems (6) do not need to be solved with a high accuracy to make reasonable progress towards the solution. Next, we illustrate the fact that the Sequential Projected Newton method is significantly less computationally expensive compared to the two reference methods, while producing solutions of similar quality. Lastly this work is concluded and an outlook is given in section 6.

2. Derivation of the sequential projected Newton method

In this section we describe a line-search strategy that can be used when solving a sequence of problems of the form (7). Let us first describe a good choice of initial point \(x_0\) to start our algorithm. We know that there exists a point \(x_0\) that satisfies \(\|s(x_0) - b\| \leq \sigma\), since we assumed that the unconstrained solution is below the noise-level. Moreover, such a point can be computed very easily by performing just a few Gauss-Newton iterations applied to \(argmin_{x \in \mathbb{R}^n} \|s(x) - b\|\). For such a point we know that the linearized inequality constraint \(\|J(x)p + s(x) - b\| \leq \sigma\) feasible, since \(p = 0\) satisfies the inequality. From this it follows that the equality constraint \(\|J(x)p + s(x) - b\| = \sigma\) is also feasible, since the constraint can be seen as a convex paraboloid. This means that the nonlinear system of equations (7) has a solution for \(x = x_0\) and hence we can use this as our initial point. See theorem 3.1 for more details on how we can compute such a point.

Let us denote \([\cdot]_+ = \max\{0, \cdot\}\). It is well known that the following function is an exact merit function for the optimization problem (1):

\[
E(x) = \Psi(x) + r[c(x)]_+.
\]

The authors in [3] use this merit function in proving global convergence of the SQCQP method. The analysis in this section is inspired by this work.

In this section we show that if we choose the penalty parameter \(r > 0\) large enough, we can then always find a step-length \(\beta \in (0, 1]\) such that \(E(x + \beta p) < E(x)\). We start by proving the following lemma:

---

\[\text{Proof.} \]
Lemma 2.1. For all $x \in B_{r}(x^*)$ there exists a constant $\phi > 0$ such that
\[
\Psi(x + \beta p) - \Psi(x) \leq -\frac{\beta \lambda}{2} p^T J(x^*) J(x) p + \lambda \beta c(x) + \phi \beta^2 \|p\|^2
\]
where $(p, \lambda)$ is the KKT pair (7).

Proof. By Taylor’s theorem we have that there exists some $\lambda \in [0, 1]$ such that
\[
\Psi(x + \beta p) = \Psi(x) + \beta \nabla \Psi(x)^T p + \frac{\beta^2}{2} p^T \nabla^2 \Psi(x + \lambda \beta p) p
\]
and thus we have for all $\beta$ that
\[
\Psi(x + \beta p) - \Psi(x) \leq \beta \nabla \Psi(x)^T p + \phi \beta^2 \|p\|^2
\]  
(9)
with $\phi = \max_{\lambda \in [0, 1]} \frac{1}{2} \|\nabla^2 \Psi(x + \lambda p)\|$. Moreover, we also have
\[
\nabla \Psi(x + p) = \nabla \Psi(x) + \int_0^1 \nabla^2 \Psi(x + \lambda p) p \, dt
\]
which implies $\nabla^2 \Psi(x)^T p \leq \nabla^2 \Psi(x) p$ since $\nabla^2 \Psi(x)$ is positive semi-definite. Using this inequality together with (9) and the KKT conditions (7), we get
\[
\Psi(x + \beta p) - \Psi(x) \leq \beta \nabla \Psi(x + \lambda \beta p)^T p + \phi \beta^2 \|p\|^2
\]
\[
= \lambda \beta \left( -\nabla c(x)^T p - p^T J(x)^T J(x) p \right) + \phi \beta^2 \|p\|^2
\]
\[
= \left( -\frac{\lambda \beta}{2} p^T J(x)^T J(x) p \right) + \phi \beta^2 \|p\|^2.
\]

\[\boxdot\]

Lemma 2.2. Let $x \in B_{r}(x^*)$ with corresponding KKT pair $(p, \lambda)$ that satisfies (7) and $r > \lambda$. Let $0 < \eta < 1$, then we have for all $\beta$ sufficiently small
\[
E_i(x + \beta p) - E_i(x) \leq -\frac{\eta \lambda \beta}{2} p^T J(x)^T J(x) p.
\]  
(10)

Proof. From theorem 2.1 we have that there exists a constant $\phi_1 > 0$ such that
\[
\Psi(x + \beta p) - \Psi(x) \leq -\frac{\beta \lambda}{2} p^T J(x)^T J(x) p + \lambda \beta c(x) + \phi \beta^2 \|p\|^2.
\]
From the second equation in (7) it follows that $c(x) + \nabla c(x)^T p \leq 0$. Hence, for all $\beta \in [0, 1]$ we have
\[
c(x) + \beta \nabla c(x)^T p \leq (1 - \beta) c(x).
\]
From this it also follows that
\[
[c(x) + \beta \nabla c(x)^T p]_+ - [c(x)]_+ \leq -\beta [c(x)]_+.
\]
Again using Taylor’s theorem we have for some $t_3 \in (0, 1)$ that
\[
[c(x + \beta p)]_+ = \left[ c(x) + \beta \nabla c(x)^T p + \frac{\beta^2}{2} p^T \nabla c(x + t_3 \beta p) p \right]_+
\]
\[
\leq \left[ c(x) + \beta \nabla c(x)^T p + \frac{\beta^2}{2} p^T \nabla c(x + t_3 \beta p) p \right]_+
\]
\[
\leq \left[ c(x) + \beta \nabla c(x)^T p + \phi_2 \beta^2 \|p\|^2 \right]_+
\]
with $\phi_2 = \max_{\lambda \in [0, 1]} \frac{1}{2} \|\nabla^2 c(x + \lambda p)\|$. As a consequence we also have
\[
[c(x + \beta p)]_+ - [c(x)]_+ \leq [c(x) + \beta \nabla c(x)^T p + [c(x)]_+ + \phi_2 \beta^2 \|p\|^2
\]
\[
\leq -\beta [c(x)]_+ + \phi_2 \beta^2 \|p\|^2.
\]
Let us now write
\[ F(x + \beta p) - F(x) = \Psi(x + \beta p) - \Psi(x) + r[(c(x + \beta p))_+ - [c(x)]_+] \leq -\frac{\beta \lambda}{2} p^T J(x) J(x) p + \lambda \beta(c(x) + \phi_1 \beta^2 \|p\|^2 - r\beta[c(x)]_+ + r \phi_2 \beta^2 \|p\|^2 \]

\[ \leq -\frac{\beta \lambda}{2} p^T J(x) J(x) p + \beta(\lambda - r)[c(x)]_+ + (\phi_1 + r \phi_2) \beta^2 \|p\|^2 \]

\[ \leq -\frac{\beta \lambda}{2} p^T J(x) J(x) p + (\phi_1 + r \phi_2) \beta^2 \|p\|^2. \]

The final inequality follows from the fact that \( r > \lambda \). It is clear that (10) holds for all values of \( \beta \) that satisfy
\[ -\frac{\beta \lambda}{2} p^T J(x) J(x) p + (\phi_1 + r \phi_2) \beta^2 \|p\|^2 \leq -\eta \beta \]

By rearranging the terms it follows that the above equality holds for all \( \beta \) with

\[ \beta \leq \frac{(1 - \eta) \lambda p^T J(x) J(x) p}{2(\phi_1 + r \phi_2) \|p\|^2}. \]

This concludes the proof since the right-hand side in this equality is strictly positive.

**Theorem 2.3.** Suppose we generate an infinite sequence of iterates \( x_k = x_{k-1} + \beta_k p_k \) that all satisfy descent condition (10) for penalty parameter \( r_2 = \max(2 \lambda_2, \eta_2 \lambda_3) \), with \( (p_2, \lambda_2) \) the KKT pair (7) corresponding with \( x_{k-1} \) and \( r_2 \), an initial penalty parameter. Suppose \( J(x_{k-1}) J(x_{k-1}) \geq \mu 1 \) for some constant \( \mu > 0 \) independent of \( k \). Suppose in addition that the iterates \( (x_k, \lambda_k) \) remain bounded and that there exists a constant \( \gamma \) independent of \( k \) such that \( \beta_k \lambda_k > \gamma > 0 \). Then there exists a sub-sequence \( \mathcal{K} \subset \mathbb{N} \) such that \( (x_k, \lambda_k) \) converges to a solution of (2).

**Proof.** By boundedness of \( (x_k, \lambda_k) \) we know, using the Bolzano-Weierstrass theorem, that there exist a convergent sub-sequence \( (x_{k_l}, \lambda_{k_l}) \to (x, \lambda) \) for some \( \mathcal{K} \subset \mathbb{N} \). Hence, we can assume that \( r_k = r \) for some constant \( r \) for all \( k \in \mathcal{K} \) sufficiently large. Since the descent condition (10) holds and \( \lambda_k \beta_k \) grows for \( k \) sufficiently large

\[ F(x_k) - F(x_{k-1}) \leq -\eta \beta_k \lambda_k \frac{p_k^T J(x_{k-1}) J(x_{k-1}) p_k}{2}. \]

In addition we have that the sequence \( F(x_k) \) is decreasing and bounded below and thus it must converge. Hence, it also follows that \( F(x_{k-1}) - F(x_k) \to 0 \). Combining this with the fact that \( F(x_{k-1}) - F(x_k) \geq \eta \beta_k \lambda_k \frac{p_k^T p_k}{2} \), we can conclude that \( p_k \to 0 \). Writing out the KKT conditions (7) for \( (x_k, \lambda_k) \) we get

\[ \begin{cases} \nabla \Psi(x_k + p_k) + \lambda_k (\nabla c(x_k) + J(x_k) J(x_k) p_k) = 0, \\ c(x_k) + \nabla c(x_k) y_p + \frac{1}{2} p_k^T J(x_k) J(x_k) p_k = 0. \end{cases} \]

Now by taking the limit for \( k \in \mathcal{K} \) and using the fact that \( p_k \to 0 \) we have

\[ \begin{cases} \nabla \Psi(x) + \lambda \nabla c(x) = 0, \\ c(x) = 0, \end{cases} \]

which concludes the proof.

**Algorithm 1.** ‘Exact’ Sequential Projected Newton method (exact SPN).

1: Initialize \( k = 0, m > 0 \) and \( \eta, \theta \in (0, 1) \) and choose tolerance \( r \).
2: Compute initial point \( x_0 \) with \( \|s(x_0) - b\| \leq \sigma \) and set \( \lambda_0 = -\frac{\nabla \Psi(x_0) \nabla c(x_0)}{\nabla \Psi(x_0) \nabla c(x_0)} \).
3: while \( \|F(x_k, \lambda_k)\| \geq r \) do
4: Solve (7) for \( x = x_k \) and denote the solution as \( (p_k, \lambda_k) \).
5: Update penalty parameter \( r_{k+1} = \max(2 \lambda_{k+1}, \eta) \).
6: Choose \( \lambda_{k+1} \) the largest value in \( \{1, \theta, \theta^2, \ldots\} \) such that
\[ E_{k+1}(x_k + \beta_k p_k, \lambda_{k+1}) - E_{k+1}(x_k) \leq -\frac{\eta \beta_k \lambda_{k+1} \gamma}{2} \|
7: Update \( \lambda_{k+1} = \max(\lambda_k, \lambda_{k+1} - \lambda_k) \).
8: end while

9: \( k = k + 1 \)
Remark 2.4. Boundedness of the Lagrange multiplier $\lambda_k$ is the most important assumption in theorem 2.3. Otherwise, the descent property (10) is meaningless, since this would imply $\eta \rightarrow \infty$. Boundedness of $x_k$ is not necessary to prove that $p_k \rightarrow 0$, which can be seen by carefully reading the proof. By (5) we know that this implies that the sub-problem (6) becomes a better and better approximation of (3). Hence, it might be possible to remove the boundedness assumption of $x_k$. However, since for practical purposes (which will become clear in the rest of the manuscript) we need to consider a more heuristic approach, we believe there is little added value in explicitly proving this.

Theorem 2.3 allows us to formulate a line-search method for solving (2), see algorithm 1. Convergence of the algorithm can be monitored using the nonlinear function

$$F(x_k, \lambda) = \left( \nabla \Psi(x_k) + \lambda \nabla c(x_k) \right) c(x_k)$$

that expresses the KKT conditions (2) of the equality constrained optimization problem (3) (not to be confused with the merit function $F(x_k)$ used in the line-search). In section 5 we also investigate other possible convergence metrics.

Note that the definition of $\lambda_0$ on line 2 can be seen as the least squares approximation of the Lagrange multiplier corresponding with $x_0$, see for instance [7]. First of all by theorem 2.2 we have that the backtracking line-search on line 6 is well-defined and terminates with some $\beta_{k+1} > 0$. Furthermore we know by theorem 2.3 that this algorithm converges towards the solution of (2) under certain assumptions.

To improve the practical performance of algorithm 1 we describe a slightly more heuristic approach. In principle we can use any constrained optimization algorithm to solve (6) on line 4. However, the Projected Newton (PN) method [8, 9] is particularly well suited to solve this problem, since it exploits some of the problem-specific properties. The Projected Newton method is itself an iterative method that uses Generalized Krylov subspaces to solve a sequence of projected problems. The number of floating point operations used in the Projected Newton method grows with the iteration index of the algorithm, so for performance it is important that the number of (inner) iterations required to converge remains relatively small. More specifically we have that the number of floating point operations in the Projected Newton method grows with $O(k_{inner}^2)$, where $k_{inner}$ is the inner iteration index. Hence, if we require that (6) is solved very accurately, then algorithm 1 will not be very efficient. In the numerical experiments section we illustrate that (6) does not have to be solved very accurately to be able to make progress towards the solution. Furthermore it turns out that (6) is a very accurate model of (3) in the sense that the descent property (10) is satisfied by $\beta = 1$ for all iterations in the experiments of section 5. Hence, to improve performance of the algorithm and remove the seemingly unnecessary line-search we describe a slightly more heuristic approach.

To formulate the computationally efficient variant of algorithm 1 we define

$$F^{(k)}(p_k, \lambda) = \left( \nabla \Psi(x_k + p_k) + \lambda \nabla c(x_k) + J(x_k)^T f(x_k) p_k \right) c(x_k) + \nabla c(x_k)^T p_k + \frac{1}{2} p_k^T J(x_k)^T J(x_k) p_k$$

(12)

the system of nonlinear equations that express the KKT conditions (7) of the sub-problem on line 4 of algorithm 1 for $x = x_k$. Algorithm 2 is a heuristic approach that works very well in practice. We illustrate this claim by a number of numerical experiments in section 5.

**Algorithm 2.** Sequential Projected Newton (SPN) method.

1. Initialize $k = 0$ and $\zeta > 1$ and choose tolerance $\tau$.
2. Compute initial point $x_0$ with $\|x_0 - b\| \leq \sigma$ and set $\lambda_0 = -\frac{\nabla \Psi(x_0)^T v(x_0)}{\nabla \Psi(x_0) v(x_0)}$.
3. while $\|F(x_k, \lambda_k)\| \geq \tau$ do
4. Apply Projected Newton method [9] to
   $$\min_{p \in \mathbb{R}^n} \Psi(s_k + p) \quad \text{subject to} \quad \frac{1}{2} \|J(x_k) p + s(x_k) - b\|^2 = \frac{\sigma^2}{2}$$
   with stopping criterion $\|F^{(k)}(p_{k+1}, x_{k+1}, \lambda_{k+1})\| \leq \frac{\max \left\{ F^{(k)}(x_{k}, \lambda_{k}), \frac{1}{\zeta} \right\}}{\tau}$.
5. Update $x_{k+1} = x_k + p_{k+1}$.
6. $k = k + 1$.
7. end while.

Remark 2.5. The introduction of the parameter $\zeta$ in algorithm 2 is inspired by the class of inexact Newton methods [10], which is a generalization of Newton's method for solving a nonlinear system of equations $F(v): \mathbb{R}^n \rightarrow \mathbb{R}^n$. In stead of solving the Jacobian linear system $F'(v_k) \Delta v_{k+1} = -F(v_k)$ exactly for some iterate $v_k \in \mathbb{R}^n$, an approximate solution is computed that satisfies $\|F'(v_k) \Delta v_{k+1} + F(v_k)\| \leq \eta_k \|F(v_k)\|$, with 'forcing
term \( n_k \in (0, 1) \). Choosing a suitable forcing sequence can significantly improve practical performance \([11, 12]\).

3. Reference methods and related work

As a first reference method we consider an algorithm that is closely related to the Sequential Quadratically Constrained Quadratic Programming (SQCP) method developed in \([3]\). It can be seen as a modified version of algorithm \(1\), where we replace sub-problem \((6)\) on line 4 with sub-problem \((8)\) that uses a quadratic model of \(\Psi(x + p)\). Moreover, we also allow an approximate solution of the sub-problem using the \(\zeta\) parameter, similarly as in algorithm \(2\). Accuracy of the sub-problem is monitored using

\[
\hat{F}^{(k)}(p, \lambda) = \left\{ \begin{array}{l}
\nabla \Psi(x_k) + \nabla^2 \Psi(x_k)p + \lambda (\nabla \xi(x_k) + f(x_k)^T f(x_k)p) \\
\xi(x_k) + \nabla \xi(x_k)^T p + \frac{1}{2} p^T f(x_k)^T f(x_k)p
\end{array} \right. 
\]

The resulting algorithm is shown in algorithm \(3\) and is referred to as SPN-Q (where Q stands for ‘quadratic’). As previously mentioned, we can readily modify the analysis in section 2 to show similar results for this modified version. In fact, the authors in \([3]\) provide a local and global convergence analysis of the closely related SQCP method.

Algorithm 3. SPN-Q method.

1: Initialize \( k = 0, \eta_0 > 0 \) and \( \eta, \theta \in (0, 1) \) and choose tolerance \( \tau \).
2: Compute initial point \( x_0 \) with \( \|\xi(x_0) - b\| \leq \sigma \) and set \( \lambda_0 = -\frac{\nabla \Psi(x_0)}{\nabla \xi(x_0)} \).
3: while \( \|F(x_k, \lambda_k)\| \geq \tau \) do
4: Apply Projected Newton method \([9]\) to
5: \[
\min_{p \in \mathbb{R}^n} F^{(k)}(x_k + \frac{1}{2} \nabla^2 \Psi(x_k)p) \text{ subject to } \frac{1}{2} \|f(x_k)p + \xi(x_k) - b\|^2 = \frac{\sigma^2}{2} \] (14)
6: Update penalty parameter \( n_{k+1} = \max(2n_{k+1}, \eta) \).
7: Choose \( \beta_{k+1} \) the largest value in \( \{1, \theta, \theta^2, \theta^3, \ldots\} \) such that
8: \[
\beta_{k+1} \|x_{k+1} + p_{k+1}\| - E_{k+1}(x_k) \leq -\frac{n_{k+1} \beta_{k+1}^2 \|f(x_k)p_{k+1}\|^2}{2}
\]
9: Update \( x_{k+1} = x_k + \beta_{k+1}p_{k+1} \) and \( \lambda_{k+1} = \lambda_k + \beta_{k+1}(\lambda_{k+1} - \lambda_k) \).
10: \( k = k + 1 \)
end while

To describe the second reference method, we consider the unconstrained reformulation of the regularized nonlinear inverse problem with fixed regularization parameter given by \((4)\). The main difficulty with solving \((4)\) is the fact that the fit-to-data term \( \|\xi(x) - b\| \) is in general a non-convex function. This means that the solution is not necessarily unique, but more importantly, this also means that the Hessian \( \nabla^2 f(x) \) of the objective function \( f(x) = \frac{1}{2}\|\xi(x) - b\|^2 + \alpha \Psi(x) \) can possibly be indefinite. This in turn implies that the standard Newton direction \( p = -\nabla f(x)^{-1} \nabla f(x) \) is not necessarily a descent direction. However, this difficulty can be circumvented by in stead computing the Gauss-Newton direction

\[
(f(x)^T f(x) + \alpha \nabla^2 \Psi(x))p = -f(x)^T (\xi(x) - b) - \alpha \nabla \Psi(x).
\] (15)

A similar approach is for instance taken in \([13]\), where the authors consider nonlinear inverse problems where the forward model is described by a system of partial differential equations. The right-hand side in \((15)\) is precisely \(-\nabla f(x)\) (i.e. the same as in Newton’s method) and \(f(x)^T f(x)\) is a positive-definite approximation of the true Hessian of the nonlinear least squares term \(\frac{1}{2}\|\xi(x) - b\|^2\). It is then easy to show that \( p \in \mathbb{R}^n \) is a descent direction for \( f(x) \), i.e. that \( p^T \nabla f(x) \leq 0 \).

This means that we can compute a suitable step-length such that we have a decrease in the objective function \( f(x + \beta p) < f(x) \). The ideas above form the basis of the Gauss-Newton method, see algorithm \(4\). The linear system of equations \((15)\) can be solved (approximately) used the Conjugate Gradient method \([4]\), which is one of the most well-known and efficient Krylov subspace methods \([14, 15]\). The stopping criterion for the linear system of equations is chosen to be \(1/\zeta\) in correspondence to the choice of tolerance used for the Projected Newton method in algorithm \(2\), see also theorem \(2.5\).

Unfortunately, due to the fact that the fit-to-data term \((4)\) is non-convex we can not use specialized algorithms such as the split Bregman method \([16]\) or ADMM \([17]\). However, SPN-Q and the Gauss-Newton method can be used regardless of which forward model \(f(x)\) is used. Hence, we will use algorithms \(4, 3\) to compare the Sequential Project Newton method with in our numerical experiments.
For linear inverse problems there also exist a wide range of different algorithms that can be used, see for instance [18–24] and references therein. Hence, many of these algorithms could also be used to solve sub-problem (6) in the Sequential Projected Newton method. However, efficiency of the Projected Newton method was already illustrated by a number of numerical experiments in [9], so we do not consider alternative algorithms for the inner solver in our experiments.

Algorithm 4. Gauss-Newton (GN) method.

1: Initialize $x_0 = 0$, $k = 0$, $\zeta > 1$, $\theta \in (0, 1)$ and choose tolerance $\tau$.
2: while $\|N(x_k) + \alpha \nabla \Psi(x_k)\| \geq \tau$ do
3: Apply the Conjugate Gradient method [4] to
$$J(x_k)^T (x_k + \alpha \nabla \Psi(x_k)) = -(\nabla \Psi(x_k) + \alpha \nabla \Psi(x_k))$$ (16) with stopping criterion
$$\|J(x_k)^T (x_k + \alpha \nabla \Psi(x_k))\| \leq 1 / \zeta$$
4: Choose $\beta_{k+1}$ the largest value in $\{1, \theta^2, \theta^3, \ldots\}$ such that
$$\frac{1}{2} \|s(x_k + \beta_{k+1} \psi_{k+1}) - b\|^2 + \alpha \Psi(x_k + \beta_{k+1} \psi_{k+1}) < \frac{1}{2} \|s(x_k) - b\|^2 + \alpha \Psi(x_k),$$
5: Update $x_{k+1} = x_k + \beta_{k+1} \psi_{k+1}$.
6: $k = k + 1$
7: end while

Remark 3.1. We can use the Gauss-Newton method (algorithm 4) with $\alpha = 0$ to compute the initial point $x_0$ used in algorithms 1, 2 and 3. In this case, the Gauss-Newton method converges towards the unregularized solution $\operatorname{argmin}_{s} \|s(x) - b\|$. We terminate the algorithm when $\|s(x) - b\| < \sigma$, which typically happens already after only a few iterations. Hence, computing this initial point is relatively cheap. We use a fixed tolerance of $10^{-10}$ (i.e. $\zeta = 10^{10}$) for (16) in all our experiments (to make sure all algorithms start from the same initial point, regardless of other parameter choices).

4. Test problem: Talbot-Lau x-ray phase contrast imaging

For our numerical experiments we consider a test-problem from Talbot-Lau phase contrast imaging [25]. The forward model of the Talbot-Lau phase contrast projection operation in the discretized setting can be described with:

$$s(\mu, \epsilon, \delta) = s_0 e^{-\mu} [1 + v_0 e^{-\epsilon} \cos(\phi_0 + A_0 \delta)]$$ (17)

where $s_0 \in \mathbb{R}$ is the incoming intensity, $v_0 \in (0, 1)$ the visibility and $\phi_0 \in \mathbb{R}^m$ the phase shift without the object. The variables $\mu$, $\epsilon$ and $\delta$ are the absorption, dark field and phase contrast, respectively. In the current manuscript we consider two-dimensional examples where the three components $\mu, \epsilon$ and $\delta$ can be visualized as pixel images. Similarly we could also consider three dimensional examples where the different components contain a large number of voxels. If we denote $n$ the total number of pixels (or voxels) then we have $\mu, \epsilon, \delta \in \mathbb{R}^n$. Now let $m$ denote the total number of projections, i.e. the number of projection angles times the number of detectors. Then we have $s(\mu, \epsilon, \delta) \in \mathbb{R}^m$. In addition, we denote $b$ the vector of length $m$ containing the acquired (noisy) projections.

The matrix $A \in \mathbb{R}^{m \times 3}$ is the conventional projection matrix and $A_0 \in \mathbb{R}^{m \times n}$ is the system matrix for differential phase contrast: $A_0 = DA$, where $D \in \mathbb{R}^{m \times m}$ is a matrix performing the finite difference operation.

We generate the matrix $A$ using the ASTRA toolbox [26, 27] based on a simple parallel beam geometry with line kernel, see appendix. Let us denote $x = [\mu^T, \epsilon^T, \delta^T] \in \mathbb{R}^n$ with $n = 3n$ the vector obtained by stacking the three components. Note that $s(x)$ is a function $\mathbb{R}^n \rightarrow \mathbb{R}$ and $s(x)$ is a function $\mathbb{R}^m \rightarrow \mathbb{R}^m$.

To use the techniques developed in this manuscript we first need to compute the Jacobian matrix $J(x)$ of the nonlinear forward model $s(x)$. To do so it is convenient to write $s(x) = \tilde{s}(\tilde{x})$, with $
abla \tilde{x} = A \tilde{x} = [\mu^T, \nu^T, \nu^T] \in \mathbb{R}^{3n}$

$$\tilde{A} = \begin{pmatrix} A & 0 & 0 \\ 0 & A & 0 \\ 0 & 0 & A_0 \end{pmatrix}$$

and $\tilde{s}(\tilde{x}) = s_0 e^{-\mu} [1 + v_0 e^{-\epsilon} \cos(\phi_0 + \nu_1)]$, $(i = 1, \ldots, m)$.

Let us calculate the Jacobian $J_i(\tilde{x})$ of the function $\tilde{s}(\tilde{x})$. For $i \neq j$ we have that $\frac{\partial \tilde{s}}{\partial \tilde{u}_i} = \frac{\partial \tilde{s}}{\partial \tilde{u}_j} = \frac{\partial \tilde{s}}{\partial \tilde{v}_i} = \frac{\partial \tilde{s}}{\partial \tilde{v}_j} = 0$ and for $i = j$ we have

$$\frac{\partial \tilde{s}_i}{\partial \tilde{u}_i} = -\tilde{s}_i, \frac{\partial \tilde{s}_i}{\partial \tilde{u}_j} = -s_0 v_0 e^{-\epsilon} e^{-\mu} \cos(\phi_0 + \nu_1) and \frac{\partial \tilde{s}_i}{\partial \tilde{v}_i} = -s_0 v_0 e^{-\epsilon} e^{-\mu} \sin(\phi_0 + \nu_1).$$
Hence we can write \( f_j(\mathbf{x}) = (-D_1(\mathbf{x}) - D_2(\mathbf{x}) - D_3(\mathbf{x})) \in \mathbb{R}^{m \times 3m} \) with
\[
D_1(\mathbf{x}) = \text{diag}(\mathbf{x})_{i=1,...,m},
\]
\[
D_2(\mathbf{x}) = \text{diag}(s_0 N_0 e^{-i\mathbf{V}_1 e^{-u}} \cos(\phi_0 + \mathbf{V}_1))_{i=1,...,m},
\]
\[
D_3(\mathbf{x}) = \text{diag}(s_0 N_0 e^{-i\mathbf{V}_1 e^{-u}} \sin(\phi_0 + \mathbf{V}_1))_{i=1,...,m}.
\]
Using the chain rule for the Jacobian \( s(x) = s(\tilde{\mathbf{x}}) \) we immediately get
\[
f(x) = f_j(\tilde{\mathbf{x}}) \tilde{\mathbf{A}} = (-D_1(\tilde{\mathbf{x}})A - D_2(\tilde{\mathbf{x}})A - D_3(\tilde{\mathbf{x}})A\phi).
\]

### 4.1. Total variation regularization

Total variation regularization is a popular regularization technique that preserves edges in images. Let \( y \in \mathbb{R}^d \) be a vector obtained by stacking all columns of a pixel image \( Y \in \mathbb{R}^{N \times N} \) with \( n = N^2 \). The anisotropic total variation function is defined as
\[
\text{TV}(y) = \sum_{i,j=1}^{N} |\partial_i^{(x,j)}(y)| + |\partial_{iv}^{(x,j)}(y)|.
\]
with finite difference operators in the horizontal and vertical direction given by
\[
\partial_i^{(x,j)}(y) = \begin{cases} Y_{i,j+1} - Y_{i,j} & j < N \\ 0 & j = N \end{cases} \quad \text{and} \quad \partial_{iv}^{(x,j)}(y) = \begin{cases} Y_{i+1,j} - Y_{i,j} & i < N \\ 0 & i = N \end{cases}
\]
We can rewrite this in a more convenient way by first writing
\[
\tilde{D} = \begin{pmatrix} 1 & -1 & \cdots & -1 \\ \vdots & \ddots & \ddots & \vdots \\ -1 & \cdots & 1 & -1 \end{pmatrix} \in \mathbb{R}^{(N-1) \times N}
\]
which represents a finite difference approximation of the derivative operator in one dimension. Let \( \otimes \) denote the Kronecker product. We can compactly write \( \text{TV}(y) = \|Dy\|_1 \) with
\[
\tilde{L} = \begin{pmatrix} D_h & D_v \end{pmatrix} \in \mathbb{R}^{(2\tilde{n} - 2N\times \tilde{n})} \text{ and } D_h = \tilde{D} \otimes I_N \in \mathbb{R}^{(\tilde{n} - N) \times \tilde{n}}, \\
D_v = I_N \otimes \tilde{D} \in \mathbb{R}^{(\tilde{n} - N) \times \tilde{n}}.
\]
The matrices \( D_h \) and \( D_v \) represent the two dimensional finite difference approximation of the derivative operator in the horizontal and the vertical directions respectively. Now because we have three different images \( \mu, \epsilon \) and \( \delta \) we consider the regularization term \( \|Lx\|_1 \) with
\[
L = \begin{pmatrix} \tilde{L} & 0 & 0 \\ 0 & \tilde{L} & 0 \\ 0 & 0 & \tilde{L} \end{pmatrix} \in \mathbb{R}^{3\tilde{n} \times \tilde{n}}
\]
and \( \tilde{m} = 3(2\tilde{n} - 2N) \). The \( \ell_1 \) norm is of course non-differentiable, however, it is easy to formulate a smooth approximation \( \Psi(x) \approx \|Lx\|_1 \), where \( \Psi : \mathbb{R}^m \rightarrow \mathbb{R} \) is a twice continuously differentiable convex function. More precisely we define
\[
\Psi(x) = \sum_{i=1}^{\tilde{m}} \sqrt{|Lx_{3i}^T| + \xi}
\]
where \( \xi > 0 \) is a small scalar that ensures smoothness. This is the same technique as used in [9].

### 5. Numerical experiments

In this section we perform a number of numerical experiments using the Talbot-Lau test-problem described in section 4. The aim is to illustrate that algorithm 2 is a robust and computationally efficient method for solving the nonlinear system of equations \( f(x, \lambda) = 0 \) defined by the KKT equations (11) for the equality constrained optimization problem (3). This function is also used to monitor convergence of the algorithm. All experiments in this section are performed using MATLAB on a laptop computer with Intel(R) Core(TM) i7-7700HQ CPU @ 2.80 GHz.

For all our experiments we set \( s_0 = 1 \) in the forward model (17), which can be seen as a normalization of the projection data. Unless explicitly stated otherwise, the value \( s_0 = 0.75 \) is used for the visibility and \( \phi_0 \) is generated using three phase steps. We refer to appendix for more details on how the test-problems are generated. For a certain exact solution \( x_{ex} = (\mu_{ex}, \epsilon_{ex}, \delta_{ex}) \) we generate exact data \( b_{ex} = s x_{ex} \) and then add Gaussian distributed white noise to obtain noisy data \( b \). We refer to the value \( \sigma = \|b - b_{ex}\| \) as the noise-level and to \( \sigma / \|b_{ex}\| \) as the...
relative noise-level. We denote npix the number of pixels in each direction (we only consider square images) and nangles the number of projection angles used and we set the number of detectors equal to npix. For these choices we have \( n = 3 \times \text{npix}^2 \) unknowns and \( m = \text{npix} \times \text{nangles} \) measurements. We always choose nangles to be a multiple (\( \geq 3 \)) of npix, which implies \( m \geq n \). In each iteration of the SPN algorithm we start the Projected Newton method with initial Lagrange multiplier \( 10^5 \), which works well in practice.

**Experiment 1.** We start by performing a small experiment with some exact images \( \mu_\text{ex} \), \( \epsilon_\text{ex} \) and \( \delta_\text{ex} \) of size \( 10 \times 10 \), i.e. npix = 10, and nangles = 4 \times npix, resulting in a projection matrix \( A \) of size 400 \times 100 (\( m = 400, n = 300 \)). Next we generate data \( b \) with relative noise-level 0.01 in MATLAB as follows.

```plaintext
1 npix = 10; nangles = 4*npix; nstep = 3; v0 = 0.75; nn = npix^2;
2 x_ex = randn(3*nn, 1)/10;
3 [A, D, phi0] = generate_testproblem(npix, nangles, nstep); %AppendixA.
4 b_ex = exp(-A*x_ex(1:nn));...
5 noiselevel = 0.01; noise = randn(m, 1);
6 b = b_ex + noise/norm(noise);
```

We apply the 'exact' Sequential Projected Newton method, algorithm 1, with \( \Psi(x) = \sum_{i=1}^{n} \sqrt{x_i^2 + \zeta} \) and parameters \( r_0 = 10^6, \eta = 10^{-4}, \theta = 0.5, \tau = 10^{-4} \) and \( \xi = 10^{-6} \). Note that this choice of regularization term can be seen as a smooth approximation of the \( \ell_1 \)-norm, which is not a good choice for the random exact solution that we generated above. However, with the current experiment we are not yet concerned with the quality of the obtained solution, but only with the convergence behavior. We use the Projected Newton method [9] to solve sub-problem and set the tolerance for this solver to \( \tau/10 \), i.e. we stop the algorithm when we have found a point that satisfies \( \|F(k)(p_{k+1}, \lambda_{k+1})\| \leq \tau/10 \), where \( F(k)(p, \lambda) \) is defined by (12). For comparison we also apply SPN-Q (algorithm 3) using the same parameters and with \( \zeta = \infty \).

The result of this experiment is given by figure 1. We compare the value \( \|F(x_0, \lambda_0)\| \) for both methods in terms of the outer SPN iterations \( k \). In addition we also report the number of inner Projected Newton iterations required to converge for the sub-problems. First of all it is important to note that the decrease condition on line 6 of algorithm 1 is satisfied by the step-length \( \beta_k = 1 \) for all iterations \( k \) in case we solve sub-problem (6). In contrast, in case we use a quadratic model (8), the line-search terminates with \( \beta_k < 1 \) for a large number of iterations. This is the reason that the convergence in the latter case is much slower. It is also interesting to observe that the quadratic model requires more Projected Newton iterations on average to satisfy the stopping criterion. Hence it is clear from this experiment why choosing sub-problem (6) over (8) is the better choice.

**Experiment 2.** In the next experiment we illustrate the fact that the sub-problem (6) does not have to be solved to a high accuracy to be able to make progress towards the solution, i.e. to decrease \( \|F(x, \lambda)\| \). We keep the same set-up as before and consider two different relative noise-levels, namely 0.1 and 0.01. We apply algorithm 2 with \( \zeta = \infty \) and \( \tau = 10^{-4}, 10^{-6}, 10^{-8} \) and \( 10^{-10} \). This can be seen as applying algorithm 1, i.e. taking a fixed tolerance for the sub-problems, but always taking a full step-length (which satisfy the decrease condition on line 6 anyway). To clearly observe the effect of the tolerance of the Projected Newton method on convergence, we let the algorithm run for a fixed number of iterations. The result of this experiment is given by figure 2. The final
achievable accuracy in terms of $\| F(x, \lambda) \|$ is completely determined by the tolerance of the inner-solver. Furthermore, it can be observed that convergence (up to stagnation of the method) does not differ much in case we use a lower tolerance for the sub-problems. This implies that in early iterations it is not necessary to compute a very accurate solution. Hence an adaptive tolerance as implemented using the $\zeta$ coefficient in algorithm 2 seems like a good idea to improve performance of the algorithm. This is especially important since the computational cost of the Projected Newton method grows with the number of iterations. With the next experiment we investigate the trade-off between keeping the number of inner iterations as low as possible without compromising the convergence in terms of the number of outer iterations needed to converge.

Experiment 3. In this experiment we study the effect of the parameter $\zeta > 1$ in algorithm 2. If this value is large, then we require a relatively accurate solution of the sub-problem and we can expect that we need a large number of inner iterations to converge. However, we can also expect to make significant progress towards the solution in terms of $\| F(x_k, \lambda_k) \|$ for the outer iterations. To confirm this we apply the Sequential Projected Newton method, algorithm 2, with different values of the $\zeta$ parameter to a Talbot-Lau test-problem with npix = 16 and nangles = 5 × npix, i.e. $m = 1280$, $n = 768$. Furthermore we take relative noise-level 0.01, visibility $v_0 = 0.5$, tolerance $\tau = 10^{-6}$ and smoothing parameter $\xi = 10^{-6}$.

In figure 3 we show convergence of the method for $\zeta = 2, 3, 10$ and $\infty$. When we take a small value such as $\zeta = 2$ we can observe that the number of inner iterations needed to satisfy the stopping criterion is relatively small. However, convergence in terms of the outer iterations is also quite slow. For the choice $\zeta = \infty$ we get very rapid convergence in terms of outer iterations since we require an accurate solution ($\tau = 10^{-6}$) of the sub-problem, but we need a lot inner iterations to satisfy the stopping criterion of the inner-solver. It is also interesting to observe that in this case the number of Projected Newton iteration decreases towards convergence. This is because the Projected Newton method starts from an initial point containing all zeroes and near the solution of $F(x, \lambda) = 0$ we have that this is indeed a good initial ’guess’ for $p_0$.

To investigate this trade-off in a bit more detail we apply algorithm 2 again with a linearly spaced range of 100 values for $\zeta$ from 1.1 to 30. The number of outer iterations required to converge are reported in figure 4 (left). If we choose $\zeta$ close to one, then we need to perform a lot of outer iterations. However, from a certain point onward, the number of outer iterations hardly changes, which implies that there is almost no benefit to solving the sub-problem very accurately.

In figure 4 (right) we show the total number of inner (Projected Newton) iterations. Again, when $\zeta$ is close to one, we need to perform a lot of inner iterations, which is due to the fact that the number of outer iterations is quite large. A value of $\zeta$ is a little less than 5 seems to result in the fewest number of inner iterations for this particular experiment. From that point onwards, the number of inner iterations starts to increase again, since we require a more accurate solution of the sub-problem, without requiring fewer outer iterations to converge.

Experiment 4. Note that since the cost of the Projected Newton method grows (quadratically) with the inner iteration index, it does not necessarily mean that more inner iterations implies a longer total runtime. Fewer inner iterations on average, but more outer iterations might be preferable to few outer iterations that require a lot of inner iterations each. Hence in this experiment we take a closer look at the average number of inner

![Figure 2. Experiment 2. Influence of the tolerance of the Projected Newton inner-solver on the convergence history in terms of outer iterations of the SPN method (algorithm 2) for two test-problems with relative noise-level 0.1 (left) and 0.01 (right).](image-url)
iterations and the overall runtime of the method. To study the performance of the Sequential Projected Newton method (algorithm 2) for different choices of $\zeta$ we report some actual timings in table 1. We use the same set-up as in experiment 3 but in addition to the choice $\text{npix} = 16$ ($m = 1280, n = 768$), we now also take $\text{npix} = 32$ ($m = 5120, n = 3072$) and $\text{npix} = 64$ ($m = 20480, n = 12228$). The timings reported in table 1 are averaged over 10 consecutive runs of the algorithm. We do not change the noise in the data for the different runs. Hence, the average number of inner iterations and the number of outer iterations do not change.

It is clear that the choice $\zeta = \infty$ results in the fewest number of outer iterations. However, it also requires the most inner (Projected Newton) iterations on average. Recall that the number of floating point operations used in the Projected Newton method increases with the number of iterations, so it is important to keep this number as low as possible. Hence, this choice for $\zeta$ performs very badly in terms of total runtime. The timings for $\zeta$ between 1.1 and 10 are all comparable, except for the case $\text{npix} = 64$ and $\zeta = 1.1$, which seems to perform a bit worse. The choice $\zeta = 10$ seems to keep the number of outer iterations close to that of the choice $\zeta = \infty$, while also keeping the average number of inner iterations modest. This value is used in all subsequent experiments.

**Experiment 5.** For the next experiment we consider total variation for the regularization function, as implemented in (19). We use the exact images for $\mu_{ex}$, $\epsilon_{ex}$ and $\delta_{ex}$ with $\text{npix} = 64$ as shown in figure 5 and we take $\text{nangles} = 5 \times \text{npix}$ ($m = 20480, n = 12228$). For the other parameters we take tolerance $\tau = 10^{-6}$, relative noise-level 0.01, $\zeta = 10$ and smoothing parameter $\xi = 10^{-8}$. Note that it is important that the
smoothing parameter is relatively small, otherwise (19) is not a good approximation of the true total regularization function (18).

To study the performance of the Sequential Projected Newton method (algorithm 2) we also apply the two reference methods in section 3, namely SPN-Q (algorithm 3) and Gauss-Newton (algorithm 4) to the same test-problem. All parameters (such as \( \zeta \) or \( \tau \)) for the different methods are kept the same (when applicable). The regularization parameter \( \alpha \) for the Gauss-Newton method is chosen to be \( 1/\lambda^* \), where \( \lambda^* \) is the final Lagrange multiplier computed using the Sequential Projected Newton method. In table 2 we report the total runtime (in seconds) as well as the number of outer iterations needed to converge. We perform 5 different runs with 5 different instances of noise. It can be observed that algorithm 2 always performs best, both in number of iterations as well as runtime. Moreover it is clear that SPN-Q is not competitive compared to the other two

---

Table 1. Experiment 4. The number of outer iterations (outer its), average number of inner iterations per outer iteration (avg. inner) and the total runtime (in seconds) of the SPN method (algorithm 2) applied to a Talbot-Lau test-problem with npix=16, 32 and 64. Reported timings are averaged over 10 runs with the same instance of noise. The number of outer and inner iterations do not change over the different runs.

| \( \zeta \) | 1.1 | 1.5 | 2 | 3 | 4 | 5 | 10 | \( \infty \) |
|---|---|---|---|---|---|---|---|---|
| npix=16 | outer its | 113 | 41 | 26 | 16 | 13 | 12 | 8 | 5 |
|        | avg. inner | 4.67 | 10.63 | 16.85 | 23.69 | 28.08 | 32.50 | 44.75 | 169.80 |
|        | runtime | 0.39 | 0.50 | 0.50 | 0.45 | 0.45 | 0.48 | 0.47 | 3.29 |
| npix=32 | outer its | 132 | 45 | 27 | 17 | 14 | 12 | 9 | 5 |
|        | avg. inner | 5.64 | 14.00 | 20.44 | 29.65 | 37.00 | 41.00 | 55.44 | 254.80 |
|        | runtime | 3.75 | 3.05 | 2.67 | 2.70 | 3.00 | 2.93 | 3.46 | 29.43 |
| npix=64 | outer its | 157 | 47 | 28 | 18 | 15 | 13 | 9 | 6 |
|        | avg. inner | 8.38 | 19.68 | 29.71 | 41.44 | 52.00 | 59.92 | 75.44 | 337.33 |
|        | runtime | 40.40 | 28.06 | 26.21 | 25.10 | 26.67 | 28.47 | 28.98 | 241.25 |

Figure 5. Experiment 5. The exact solution used in the experiment together with the solutions obtained by algorithm 2 (SPN), algorithm 3 (SPN-Q) and algorithm 4 (GN) with total variation regularization.
The reconstructed solutions are shown in figure 5. Note that all methods (SPN, SPN-Q and GN) produce a solution of similar quality, which is not surprising since they are all essentially solving the same regularized nonlinear inverse problem using the total variation regularization function. We can clearly see that the reconstructed solutions are quite good and that the total variation regularization function nicely removes noise in the reconstruction, while still preserving edges in the images.

In figure 6 (left) we report some different convergence metrics for the SPN method (algorithm 2) applied to a particular instance of the test-problem. We show convergence of the values $\|F(x_k, \lambda_k)\|$ as always, as well as the relative difference of the iterates $x_k$ and $\lambda_k$. In addition we also show the convergence of the constraint by plotting $\|s(x_k) - b\| - \sigma$. These first three metrics seem to converge in a similar fashion, while the latter value converges much more quickly.

In the figure on the right, we show the Structural Similarity Index (SSIM) of the reconstructed solution compared with the exact solution. The closer this value is to 1 the more similar the two images are. The actual formula for the SSIM is quite involved so we do not explicitly write it down here but rather refer to [28] where it was first introduced. We compute this value using the MATLAB function ssim. It is interesting to see that the Structural Similarity Index stabilizes very quickly, even when $\|F(x_k, \lambda_k)\|$ is not yet very small. Hence, it might be appropriate for practical problem to use a stopping criterion based on $\|s(x_k) - b\| - \sigma$, which decreases much more quickly. Similar observations were made by the authors in [9].

**Experiment 6.** For our final experiment we use entirely the same set-up as before, but now use a larger test-problem with $\text{n pix} = 128$ ($m = 81920, n = 49152$) as shown in figure 7. Furthermore, we now choose a moderate tolerance $\tau = 10^{-3}$. Results of this experiment can be found in figures 7, 8. The same conclusions as in the previous experiment can be made here. Results comparing algorithm 2 with Gauss-Newton are presented in table 3. We do not include the SPN-Q method in this table since a single run took more than one hour to complete. Again we can conclude that the Sequential Projected Newton method converges much more quickly than the Gauss-Newton method, while producing a solution of similar quality.

| Table 2. Experiment 5. The number of outer iterations (outer its) and total runtime (in seconds) of the SPN method (algorithm 2), SPN-Q (algorithm 3) and Gauss-Newton (algorithm 4) for 5 different runs (with different instances of noise) applied to the Talbot-Lau test-problem with $\text{n pix} = 64$. The standard deviation (Std dev) over the different runs is also reported for the different quantities. |
|---|---|---|---|---|---|
| run | 1 | 2 | 3 | 4 | 5 | Std dev |
| SPN | outer its | 10 | 10 | 10 | 10 | 10 | 0 |
| | runtime | 27.17 | 30.01 | 29.75 | 29.06 | 30.03 | 1.20 |
| SPN-Q | outer its | 87 | 91 | 152 | 89 | 93 | 27.82 |
| | runtime | 482.79 | 329.73 | 721.85 | 343.79 | 354.73 | 165.64 |
| Gauss-Newton | outer its | 88 | 74 | 63 | 88 | 65 | 12.05 |
| | runtime | 79.96 | 61.54 | 56.77 | 82.38 | 56.16 | 12.80 |
Figure 7. Experiment 6. The exact solution used in the experiment together with the solutions obtained by algorithm 2 (SPN), algorithm 3 (SPN-Q) and algorithm 4 (GN) with total variation regularization.

Figure 8. Experiment 6. (Left) Convergence metrics for the SPN method (algorithm 2) of a Talbot-Lau experiment with \( npix = 128 \) and \( n\text{angles} = 5 \times npix \). Tolerance is set to \( \tau = 10^{-3} \) and noise-level 0.01. (Right) Structural Similarity index for the different components \( \mu_k \), \( \delta_k \) and \( \sigma_k \) for the same experiment.

Table 3. Experiment 6. The number of outer iterations and total runtime (in seconds) of the SPN method (algorithm 2) and Gauss-Newton (algorithm 4) for 5 different runs (with different instances of noise) applied to the Tablot Lau test-problem with \( npix = 128 \). The standard deviation (Std dev) over the different runs is also reported for the different quantities.

| run | outer its | runtime  | outer its | runtime  | outer its | runtime  | outer its | runtime  | outer its | runtime  |
|-----|-----------|----------|-----------|----------|-----------|----------|-----------|----------|-----------|----------|
| SPN | 8         | 145.15   | 8         | 148.30   | 8         | 155.66   | 8         | 144.43   | 166.80   | 9.36     |
|     | 8         | 144.33   | 8         | 155.66   | 8         | 144.43   | 8         | 166.80   | 9.36     |          |
|     | 8         | 144.33   | 8         | 155.66   | 8         | 144.43   | 8         | 166.80   | 9.36     |          |
|     | 8         | 144.33   | 8         | 155.66   | 8         | 166.80   | 9.36     |          |          |          |
|     | 8         | 144.33   | 8         | 155.66   | 8         | 166.80   | 9.36     |          |          |          |
|     | 8         | 144.33   | 8         | 155.66   | 8         | 166.80   | 9.36     |          |          |          |
|     | 8         | 144.33   | 8         | 155.66   | 8         | 166.80   | 9.36     |          |          |          |
|     | 8         | 144.33   | 8         | 155.66   | 8         | 166.80   | 9.36     |          |          |          |
|     | 8         | 144.33   | 8         | 155.66   | 8         | 166.80   | 9.36     |          |          |          |
|     | 8         | 144.33   | 8         | 155.66   | 8         | 166.80   | 9.36     |          |          |          |
|     | 8         | 144.33   | 8         | 155.66   | 8         | 166.80   | 9.36     |          |          |          |
6. Conclusions and outlook

In this work we have developed an algorithm for the automatic regularization of nonlinear inverse problems based on the discrepancy principle. By considering a linear approximation of the forward model using the Jacobian matrix, we obtain a quadratically constrained optimization problem which can be solved using the Projected Newton method. We prove that the solution of this equality constrained optimization problem results in a descent direction for a certain merit function, which can be used to formulate a formal line-search method for solving the inverse problem. We also formulate a slightly more heuristic version of the algorithm by loosening the tolerance of the solution of the sub-problems and removing the (seemingly unnecessary) line-search. We illustrate using the numerical experiments that these simplifications lead to a robust and computationally efficient method (in terms of runtime) for solving the regularized nonlinear inverse problem compared to two reference methods. Moreover, we show that we can obtain high quality reconstructions for Talbot-Lau x-ray phase contrast imaging using the proposed algorithm.

Since the Sequential Projected Newton method uses the discrepancy principle, it can only be applied in case we have a good estimate of the noise-level available. It might be worth investigating if similar techniques as presented in this manuscript can also be used in combination with other parameter choice methods, such as generalized cross validation or the L-curve criterion. Moreover, we have assumed throughout this entire work that the nonlinear inverse problem is overdetermined. However, in many applications (including Talbot-Lau phase contrast imaging) this is not always the case. Hence, a generalization of the Sequential Projected Newton in case of an underdetermined inverse problem would certainly be valuable. Preliminary experiments (not reported in this work) indicate that the Sequential Projected Newton method also often converges nicely in the underdetermined case, but can also diverge on some test-problems. Hence, further research and safeguards to ensure convergence are required.

Acknowledgments

This work was funded in part by the IOF-SBO project entitled High performance iterative reconstruction methods for Talbot-Lau grating interferometry based phase contrast tomography.

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.

Appendix. MATLAB code for generating Talbot-Lau test-problem

The following MATLAB code is based on the ASTRA toolbox [26, 27].

```matlab
function [A, D, phi0] = generate_testproblem(npix, nangles, nstep)
% Matlab function to generate Talbot-Lau test-problem.
% Input : -npix = number of pixels in each direction
% -nangles = number of projection angles
% -nstep = number of phase steps
% Output: -m x n matrix A, with m = npix*nangles and n = npix^2
% -m x m finite difference matrix D
% -m x 1 vector phi0: phase shift without the object
angles = linspace2(0, pi, nangles);
vol_geom = astra_create_vol_geom(npix, npix);
proj_geom = astra_create_proj_geom('parallel', 1.0, npix, angles);
proj_id = astra_create_projector('line', proj_geom, vol_geom);
matrix_id = astra_mex_projector('matrix', proj_id);
A = astra_mex_matrix('get', matrix_id);
astra_mex_projector('delete', proj_id);
astra_mex_matrix('delete', matrix_id);
D1 = spdiags([-ones(npix, 1), +ones(npix, 1)], [0 1], npix, npix);
I = spspeye(nangles); D = kron(I, D1);
phi_list = mod(0: (nangles-1), nstep) + 1;
PHI_0 = linspace2(0, pi, nstep);
```
for ii = 1:length(phi0), phi0(ii) = PHI_i; end