Gradient Flow Method for Nonlinear Least Squares Minimization

Neculai Andrei[1]

Abstract

Solving the Nonlinear Least Squares Problem by means of integration of a first order ordinary differential equation is considered in this paper. The corresponding gradient flow algorithm and its variants, based on the approximation of the Hessian matrix associated to the residual functions of the problem, are presented. The ordinary differential equation is integrated by means of a two level implicit time discretization technique, with a splitting parameter $\theta \in [0,1]$. The Hessian matrices of residual functions of the problem are approximated using the function values and its gradient in two successive points along the trajectory of the differential equation. The convergence of the algorithms is analysed and it is shown that this is linear when $0 \le \theta < 1$ and quadratical when $\theta = 1$ and the integration step is sufficiently large. The main result of the paper shows that the best algorithm corresponds to the case when the approximations of the Hessian matrices of the residual functions of the problem are not considered in the algorithm. The obtained algorithm is quadratically convergent when the integration step is sufficiently large. In fact, this algorithm, with no second order information about the residual functions of the problem, is a new algebraic expression of the Levenberg-Marquardt algorithm in which the positive parameter is the reciprocal of the time discretization step. Some numerical examples illustrate the algorithms.

Key words: Nonlinear Least Squares Problem, gradient flow, Levenberg-Marquardt, Gauss-Newton, approximation of the Hessian.

AMS (MCS) subject classifications. 34G20, 65K05, 90C30,

1. Introduction

The problem we consider in this paper is the nonlinear least squares minimization:

$$\min \Phi(x) \tag{1}$$

where $\Phi(x) = \frac{1}{2}\|F(x)\|^2$ and $F(x) = [f_1(x), ..., f_m(x)] : R^n \to R^m$ is continuously differentiable. Often, each $f_i(x)$ is referred as a *residual*, and in the most practical applications, especially arising in data-fitting, $m \ge n$.

The most important feature that distinguishes least squares problems from the general unconstrained optimization problems is the structure of the Hessian matrix of Φ. The Jacobian and Hessian matrices of Φ are as follows:

$$\nabla\Phi(x) = \nabla F(x)^T F(x),$$

[1] Research Institute for Informatics, 8-10 Averescu Avenue, Bucharest 1, Romania, E-mail:nandrei@ici.ro

$$\nabla^2\Phi(x) = \nabla F(x)^T \nabla F(x) + \sum_{i=1}^{m} f_i(x)\nabla^2 f_i(x).$$

To calculate the $\nabla\Phi(x)$ and $\nabla^2\Phi(x)$ we must to calculate the Jacobian matrix of $F(x)$, i.e. $\nabla F(x)$. In many practical situations the first term from the Hessian, $\nabla F(x)^T \nabla F(x)$, is more important that the second one, especially when the residuals are small at the solution point.

For solving the nonlinear least squares problems (1) an algorithm which is very suitable when the residuals are small, is the Gauss-Newton algorithm. In this algorithm the Hessian is approximated by its first term. At the current point x_k the searching direction d_k is computed as solution of the following system of linear equations:

$$\left(\nabla F(x_k)^T \nabla F(x_k)\right) d_k = -\nabla F(x_k)^T F(x_k). \tag{2}$$

The next approximation of the solution x^* of (1) is computed as $x_{k+1} = x_k + d_k$. It is very simple to see that any solution of (2) is a descent direction for Φ, since as well as $\nabla\Phi(x_k) \neq 0$,

$$d_k^T \nabla\Phi(x_k) = -\left\|\nabla F(x_k) d_k\right\|^2 < 0.$$

In order to be convergent the Gauss-Newton method must be initialized in a point x_0 sufficiently close to a solution x^* of (1).

A modification of Gauss-Newton's method, designed to overcome this limitation is the Levenberg-Marquardt method [25,26]. In this method the search direction d_k is computed as a solution to the following system of linear equations:

$$\left(\nabla F(x_k)^T \nabla F(x_k) + \mu_k I\right) d_k = -\nabla F(x_k)^T F(x_k), \tag{3}$$

where μ_k is a positive parameter which controls both the magnitude and direction of d_k. A close inspection of (3) shows that the Levenberg-Marquardt algorithm is a blend of gradient descent and Gauss-Newton iterations. Since the matrix $\nabla F(x_k)^T \nabla F(x_k) + \mu_k I$ is always positive definite, it follows that (3) has a unique solution. It is very easy to prove that if $\nabla\Phi(x_k) \neq 0$, then the solution d_k of the linear system (3) satisfy $\nabla\Phi(x_k)^T d_k < 0$, proving that d_k is indeed a descent direction of Φ. Therefore the Levenberg-Marquardt method with, let say, Armijo's stepsize selection rule is globally convergent to a stationary point x^* of Φ. When μ_k is zero, then the direction d_k is identical to that of the Gauss-Newton algorithm. As μ_k tends to infinity, then d_k tends to a steepest descent direction. Therefore, for some sufficiently large μ_k, we have $\Phi(x_k + d_k) < \Phi(x_k)$.

For many problems the Levenberg-Marquardt algorithm is preferable to damped Gauss-Newton algorithm. This is because the Levenberg-Marquardt algorithm is well defined even when $\nabla F(x_k)$ at the current point x_k doesn't have full column rank. On the other hand, when the Gauss-Newton step is much too long, the Levenberg-Marquardt step is close to the steepest-descent direction $-\nabla F(x_k)^T F(x_k)$, which often is superior to the damped Gauss-Newton step [14, pp.228], [28], [29], [30].

The main difficulty of the Levenberg-Marquardt algorithm is the lack of an effective strategy for controlling the magnitude of μ_k at each iteration, in such a way that the corresponding algorithm to be efficient for a large spectrum of problems. Recently, Yamashita and Fukushima [38] proved that if $\|\nabla\Phi(x)\|$ provides a „local error bound“ for the problem $\nabla\Phi(x) = 0$ and the parameter μ is chosen as $\mu_k = \|\nabla\Phi(x_k)\|^2$, then the Levenberg-Marquardt algorithm retains the quadratic convergence property. Fan and Yuan [20] discuss some possible limitations of the choice $\mu_k = \|\nabla\Phi(x_k)\|^2$ in (3). If x_k is very close to the solution, then μ_k can be very small. Therefore, μ_k has no influence in (3). On the other hand, if x_k is far away from the solution, then μ_k would be very large and the solution of (3) may be very small. Taking this analysis into account, Fan and Yuan propose to use $\mu_k = \|\nabla\Phi(x_k)\|$ in (3), thus proving the local quadratic convergence of the corresponding algorithm. Some other improvements and convergence properties of the Levenberg-Marquardt algorithm and its inexact variant has been considered in [13,38].

In this paper we propose a gradient flow algorithm for solving the nonlinear least squares problem based on the integration of a first order ordinary differential equation. Section 2 presents the general flow algorithm, together with its properties, for solving nonlinear least squares problems. The main result of this paper is given in section 3. We show that a very simple modification of the gradient flow algorithm, consisting of rejection the second order terms, give the best algorithm of this approach. In this respect we consider some variants of the gradient flow algorithm in which the Hessian matrices of functions f_i are approximated by scalars, for which different formula are suggested. It is shown that the convergence of the resulting algorithms is quadratical when the splitting parameter has a unitar value and the integration step is sufficienly large. Rejecting the second order term, the corresponding algorithm is an equivalent algebraic expression of the Levenberg-Marquardt algorithm for which we prove its quadratic convergence. Thus, the gradient flow approach for solving least squares minimization problems gives a strong theoretical basis for the Levenberg-Marquardt algorithm. Section 4 gives some numerical experience with this approach of nonlinear least squares problems.

2. Gradient flow algorithm for nonlinear least squares minimization

As we know, a necessary condition for the point x^* be an optimal solution for (1) is:

$$\nabla\Phi(x^*) = 0. \tag{4}$$

In order to fulfill this optimality condition the following continuous gradient flow reformulation of the problem is considered: *solve the ordinary differential equation*:

$$\frac{dx(t)}{dt} = -\nabla\Phi(x(t)) \tag{5}$$

with the initial condition

$$x(0) = x_0. \tag{6}$$

Therefore, the minimization problem (1) has been reduced to the integration of the differential equation (5) with initial condition (6). Methods of this type, using the idea of following the trajectory of a system of ordinary differential equations, are not new and a number of authors have been proposed numerous ordinary differential equations and computational schemes for their integration. To have an idea about this subject let us shortly review them.

Let $y(t)$ be the displacement from the current point x. The initial condition for all equations is therefore $y(0) = 0$. The Courant's method [12], based on idea of Hadamard [21] is to solve the differential equation:

$$y'(t) = -\nabla\Phi(x_0 + y(t)). \tag{7}$$

Boggs [5] extended this idea of Courant by using a predictor-corrector method for solving (7) in connection with a quasi-Newton approximation of the Hessian.

Considering the local approximation of (7) along a sequence of points we get:

$$y_i'(t) = -\nabla\Phi(x_i) - \nabla^2\Phi(x_i)y_i(t). \tag{8}$$

When the Hessian is nonsingular, Botsaris and Jacobson [9] use (8) for solving (1). Otherwise, in order to bound the solution, they replace the Hessian by a matrix with the same eigenvectors, but with the absolute values of the eigenvalues. Vial and Zang [36], and Zang [39], use a quasi-Newton approximation of the Hessian in (8). Botsaris dedicated a number of paper for integration of (8) [6,7]. He considers an approximation of the Hessian which is updated at each step by means of the Sherman-Morrison formula.

Another equation, known as the continuous Newton equation is

$$y'(t) = -\nabla^2\Phi(x_0 + y(t))^{-1}\nabla\Phi(x_0 + y(t)). \tag{9}$$

This equation has been considered by Botsaris [8], where he considers an implicit ordinary differential equation solver with an approximation of the inverse of the Hessian matrix which is updated by means of Sherman-Morrison formula.

Finally, considering a mechanical interpretation of the problem, by following the trajectory of a point mass in the force field $-\nabla\Phi$ with dissipation, Aluffi-Pentini, Parisi and Zirilli [1,2], use the second order differential equation:

$$a(t)y''(t) + b(t)y'(t) + \nabla\Phi(x_0 + y(t)) = 0, \tag{10}$$

where $a(t)$ and $b(t)$ are positive, real-valued functions. For solving (10) they consider an implicit ordinary differential equation solver and a quasi-Newton approximation of the Hessian. Zirilli *et al*, [40,41] describe different practical procedures for choosing $a(t)$ and $b(t)$ during the integration of (10), proving that as $t \to \infty$, the solution trajectory is very close to that of Newton's method. They show that using the second order differential equations gives a larger domain of convergence than that corresponding to the first order system. More than this, (10) permits a greater control of the trajectory since at $t = 0$ we must specify not only the initial point $y(0)$ but also $y'(0)$. Different choices for $y'(0)$ may lead to different solution. A similar approach, based on second order differential equations, was considered by Snyman [34] by solving the differential equation $y''(t) = -\nabla\Phi(y(t))$.

Brown and Bartholomew-Biggs [10,11] experiment a number of methods based on all these differential equations (8)-(10) using specialized ordinary differential equations solvers. Their conclusion is that the most successful method is that based on (8) using the Hessian or a quasi-Newton approxination of it.

Behrman [4] in his Dissertation solves the problem (1) by an algorithm which basically calculates a curve that is an approximation to the integral curve of the vector field $-\nabla\Phi$. A searching procedure along this curve is initiated, determining a point that reduces the value of the objective function Φ.

For unconstrained optimization, the gradient flow method is presented by Andrei [3], where the ordinary differential equation (5) is integrated by means of a discretization scheme based on a two level implicit time discretization technique, with a splitting parameter $\theta \in [0,1]$. The convergence of the algorithm is linear when $0 \le \theta < 1$ and quadratic when $\theta = 1$ and the integration step is sufficiently large.

In a more general context refering to the constrained optimization, the gradient flow methods, known as stable barrier-projection and barrier-Newton methods have been considered by Evtushenko [15,16], and Evtushenko and Zhadan [17-19]. Convergence of these methods via Lyapunov functions has been considered by Smirnov [33]. Recently, for solving constrained optimization problems, improvements and some computational experience with these methods have been considered by Wang, Yang and Teo [37]. Basically, in this approach a constrained optimization problem is reformulated as an ordinary differential equation in such a way that its solution converges to an equillibrium point of the optimization problem as parameter t from this equation goes to ∞. We see that this approach based on reformulation of the optimization problem as a differential equation was and continue to be very attractive and promising. See also the book by Helmke and Moore [22].

In the following we shall present the main convergence results and the corresponding gradient flow algorithm for solving (1) by integration of the system (5) with initial condition (6) [3].

Theorem 2.1. *Consider that x^* is a point satisfying (4) and $\nabla^2\Phi(x^*)$ is positive definite. If x_0 is sufficiently close to x^*, then $x(t)$, the solution of (5) with initial condition x_0, tends to x^* as t goes to ∞.*

Proof. The system (5) can be written as

$$\dot{x} = \Psi(x),$$

where $\Psi(x) = -\nabla\Phi(x)$. To show that x^* is an asymptotically stable point for (5) we shall consider the Poincaré-Lyapunov theory [35]. According to this theory, x^* is an asymptotically stable point for the nonlinear differential equation system $\dot{x} = \Psi(x)$ if $\Psi(x)$ is continuously differentiable and the linearized system

$$\dot{y} = \nabla\Psi(x^*)y,$$

where $y = x - x^*$, is exponentially stable, i.e. all eigenvalues of $\nabla\Psi(x^*)$ are strictly negative. Considering the Taylor's expansion of $\Psi(x)$ around x^*, and using (4), we get:

$$\begin{aligned} \frac{dx}{dt} &\cong \Psi(x^*) + \nabla\Psi(x^*)(x - x^*) \\ &= -\left[\nabla\Phi(x^*) + \nabla^2\Phi(x^*)(x - x^*)\right] \\ &= -\nabla^2\Phi(x^*)(x - x^*). \end{aligned}$$

But, $\nabla^2\Phi(x^*)$ is positive definite by the assumption of the theorem. Therefore, its eigenvalues satisfy $\lambda_i > 0$, for all $i = 1, ..., n$. By the Poincaré-Lyapunov theory it follows that $\lim\limits_{t\to\infty} y(t) = 0$,or $x(t) \to x^*$ as $t \to \infty$. ■

The following theorem shows that $\Phi(x(t))$ is strictly decreasing along the trajectory solution of (5).

Theorem 2.2. *Let* $x(t)$ *be the solution of (5) with initial condition (6). For a fixed* $t_0 \geq 0$ *if* $\nabla\Phi(x(t)) \neq 0$ *for all* $t > t_0$, *then* $\Phi(x(t))$ *is strictly decreasing with respect to* t, *for all* $t > t_0$.

Proof. We have:

$$\frac{d\Phi(x(t))}{dt} = \nabla\Phi(x(t))^T \frac{dx(t)}{dt} = -\nabla\Phi(x(t))^T \nabla\Phi(x(t)) = -\left\|\nabla\Phi(x(t))\right\|_2^2 .$$

Since $\nabla\Phi(x(t)) \neq 0$ when $t > t_0$, it follows that $d\Phi(x(t))/dt < 0$, i.e. $\Phi(x(t))$ is strictly decreasing with respect to $t > t_0$. ■

Observe that the ordinary differential equation (5), associated to (1), is a gradient system [23, pp.199]. Gradient systems have special properties that make their flows very simple. For gradient system (5) at regular points x, characterized by the fact that $\nabla\Phi(x) \neq 0$, the trajectories cross level surfaces of the function $\Phi(x)$ orthogonally. Nonregular points are equillibria of the system, and if x^* is an isolated minimum of $\Phi(x)$, then x^* is an asymptotically stable equilibrium of the gradient system (5).

Therefore, solving the unconstrained optimization problem (1) has been reduced to that of integration of the ordinary differential equation (5) with initial condition (6). Now we shall consider a discretization of this equation as well as the corresponding integration scheme.

Let $0 = t_0 < t_1 < \cdots < t_k < \cdots$ be a sequence of time points for the time $t \geq t_0$. Consider $h_k = t_{k+1} - t_k$ the sequence of time distances between two successive time points. With these, let us consider the following time-steeping discretization of (5):

$$\frac{x_{k+1} - x_k}{h_k} = -\left[(1-\theta)\nabla\Phi(x_k) + \theta\nabla\Phi(x_{k+1})\right], \tag{11}$$

where $\theta \in [0,1]$ is a parameter. From this we get:

$$x_{k+1} = x_k - h_k\left[(1-\theta)\nabla\Phi(x_k) + \theta\nabla\Phi(x_{k+1})\right].$$

When $\theta = 0$ the above discretization is the explicit forward Euler's scheme. On the other hand, when $\theta = 1$ we have the implicit backward Euler's scheme. But,

$$\nabla\Phi(x_{k+1}) = \nabla\Phi(x_k) + \nabla^2\Phi(x_k)\delta x_k + \Gamma(\delta x_k),$$

where $\delta x_k = x_{k+1} - x_k$ and $\Gamma(\delta x_k)$ is the remainder satisfying $\|\Gamma(\delta x_k)\| = O\left(\|\delta x_k\|^2\right)$. Therefore

$$x_{k+1} = x_k - h_k\left[I + h_k\theta\nabla^2\Phi(x_k)\right]^{-1}\left[\nabla\Phi(x_k) + \theta\Gamma(\delta x_k)\right].$$

Omitting the higher order term $\Gamma(\delta x_k)$ we get:

$$x_{k+1} = x_k - h_k\left[I + h_k\theta\nabla^2\Phi(x_k)\right]^{-1}\nabla\Phi(x_k), \tag{12}$$

for any $\theta \in [0,1]$. Considering x_0 as the initial guess, then (12) defines a series $\{x_k\}$. Like in [37], the convergence of (12) is given by

Theorem 2.3. *Let $\{x_k\}$ be the sequence defined by (12) and x^* a solution of (1), such that $\nabla^2\Phi(x^*)$ is positive definite. If the initial point x_0 is sufficiently close to x^*, then:*

(i) If $\theta \in [0,1]$ and $h_k > 0$ is sufficiently small, then x_k converges linearly to x^*.

(ii) If $\theta = 1$ and $h_k \to \infty$, then x_k converges quadratically to x^*.

Proof. *(i)* From (12) we have:

$$\left[I + h_k\theta\nabla^2\Phi(x_k)\right](x_{k+1} - x_k) = -h_k\nabla\Phi(x_k)$$

hence:

$$x_{k+1} = x_k - h_k\left[\nabla\Phi(x_k) + \theta\nabla^2\Phi(x_k)(x_{k+1} - x_k)\right]. \tag{13}$$

Subtracting x^* from both sides of (13) and having in view that $e_k = x_k - x^*$, $x_{k+1} - x_k = e_{k+1} - e_k$ and $\nabla\Phi(x^*) = 0,$ we get:

$$e_{k+1} = e_k - h_k\left[\nabla\Phi(x_k) - \nabla\Phi(x^*) + \theta\nabla^2\Phi(x_k)(e_{k+1} - e_k)\right].$$

Now using the mean value theorem we have:

$$e_{k+1} = e_k - h_k\left[\nabla^2\Phi(\xi_k)e_k + \theta\nabla^2\Phi(x_k)(e_{k+1} - e_k)\right],$$

where $\xi_k \in [x_k, x^*]$. Solving for e_{k+1}, we have:

$$e_{k+1} = \left\{I - h_k\left[I + h_k\theta\nabla^2\Phi(x_k)\right]^{-1}\nabla^2\Phi(\xi_k)\right\}e_k. \tag{14}$$

Considering the norm of both sides of this equality we obtain:

$$\|e_{k+1}\| \le \varphi(x_k, \xi_k, \theta, h_k)\,\|e_k\|, \tag{15}$$

where

$$\varphi(x_k, \xi_k, \theta, h_k) = \left\|I - h_k\left[I + h_k\theta\nabla^2\Phi(x_k)\right]^{-1}\nabla^2\Phi(\xi_k)\right\|. \tag{16}$$

From (16) we see that if $\varphi(x_k, \xi_k, \theta, h_k) < 1$, then e_k converges to zero linearly. Using continuity and the fact that x_0 is close to x^* we can write:

$$\varphi(x_k, \xi_k, \theta, h_k) \le \left(1 - \frac{h_k\lambda^k_{\min}}{1 + h_k\theta\lambda^k_{\max}}\right) < 1, \tag{17}$$

where $\lambda^k_{\min}$ and $\lambda^k_{\max}$ represent the minimum and the maximum eigenvalues of $\nabla^2\Phi(x_k)$, respectively. Therefore, from (17) it follows that $\lim\limits_{k\to\infty} e_k = 0$ linearly, i.e. $x_k \to x^*$ linearly.

(ii) Consider $\theta = 1$ in (11), we get:

$$\frac{x_{k+1} - x_k}{h_k} = -\left[\nabla\Phi(x_k) + \nabla^2\Phi(x_k)\delta x_k\right],$$

where $\delta x_k = x_{k+1} - x_k$. When $h_k \to \infty$ the above relation is reduced to

$$\nabla\Phi(x_k) + \nabla^2\Phi(x_k)\delta x_k = 0$$

which is the Newton method applied to $\nabla\Phi(x) = 0$. When x_k is sufficiently close to x^*, as we know the Newton method is quadratically convergent, proving the theorem. ■

Remark 2.1. From (15) and (17), with $\theta = 1$, we have:

$$\|e_{k+1}\| \leq p_{k+1} \|e_0\|$$

where

$$p_{k+1} = \prod_{i=0}^{k} \left(1 - \frac{h_i \lambda_{\min}^i}{1 + h_i \lambda_{\max}^i} \right).$$

But, $\nabla^2 \Phi(x_i)$ is positive definite, therefore for all $i = 0, \dots k$,

$$0 < 1 - \frac{h_i \lambda_{\min}^i}{1 + h_i \lambda_{\max}^i} < 1.$$

So, p_k, for all k, is a decreasing sequence, from $(0, 1)$, i.e. it is convergent. If $h_i \to \infty$, then for all $i = 0, \dots k$,

$$1 - \frac{h_i \lambda_{\min}^i}{1 + h_i \lambda_{\max}^i} \to 1 - 1/\kappa \left(\nabla^2 \Phi(x_i) \right).$$

Clearly, if there is an i for which $\kappa \left(\nabla^2 \Phi(x_i) \right)$ is close to 1, then the convergence of the algorithm is very rapid. ■

As the theorem 2.3 recommends, the algorithm based on (12) is quadratically convergent if $\theta = 1$ and $h_k \to \infty$. The problem is how to choose the sequence h_k. The most direct idea is to choose h_k in such a way that the matrix

$$I + h_k \theta \nabla^2 \Phi(x_k)$$

to be positive definite. The following theorem suggests how to choose the value h_k of time distances between two successive time points.

Theorem 2.4. *If* $h_k > \max \left\{ -\frac{1}{\lambda_i^k}, i = 1, \dots, n \right\}$, *where* λ_i^k, $i = 1, \dots, n$, *are the eigenvalues of* $\nabla^2 \Phi(x_k)$, *then* $\left[I + h_k \nabla^2 \Phi(x_k) \right]$ *is positive definite.*

Proof. The matrix $\nabla^2 \Phi(x_k)$ is symmetric, i.e. it has real eigenvalues λ_i^k, $i = 1, \dots, n$. There exists a matrix P such that:

$$P^{-1} \nabla^2 \Phi(x_k) P = diag(\lambda_1^k, \dots, \lambda_n^k).$$

Therefore, $P^{-1} \left[I + h_k \nabla^2 \Phi(x_k) \right] P = I + h_k diag \left(\lambda_1^k, ..., \lambda_n^k \right)$, which is positive definite when $1 + h_k \lambda_i^k > 0$, for all $i = 1, ..., n$. ■

The above presented results are very general and can be applied to any function $\Phi(x)$ satisfying the conditions of the above theorems. Basically, the function Φ must have a positive definite Hessian at solution point. Now, in order to get an algorithm for solving the nonlinear least squares problem (1) we shall particularize the above gradient flow algorithm by considering the special structure of the Hessian of function $\Phi(x)$.

Proposition 2.1. If $f_i(x)$ is a convex function for all $i = 1, ..., m$, $f_i(x) \geq 0$, for all $i = 1, ..., m$ and $rank(\nabla F(x)) = n$, then $\nabla^2 \Phi(x)$ is positive definite.

Proof. For every $y \neq 0$,we see that $y^T (\nabla F(x)^T \nabla F(x)) y = \|\nabla F(x) y\|^2 > 0$ since $rank(\nabla F(x)) = n$. On the other hand, $f_i(x) \geq 0$, therefore $f_i(x) \nabla^2 f_i(x)$ is a positive definite matrix, since f_i is a convex function. ■

Therefore, in conditions of Proposition 2.1 all the above theorems remain true showing the convergence of the method when applied to this particular form of function $\Phi(x)$. Therefore, for solving (1) by integration of the ordinary differential equation (5), the following algorithm can be presented:

Algorithm GFA (Gradient Flow Algorithm)

Step 1. Consider the initial point $x_0 \in R^n$, a parameter $\theta \in [0,1]$, a sequence of time step sizes $\{h_k\}$ and an $\varepsilon > 0$ sufficiently smal. Set $k = 0$.

Step 2. Solve for d_k the system:

$$\left[I + h_k\theta\left(\nabla F(x_k)^T\nabla F(x_k) + \sum_{i=1}^{m} f_i(x_k)\nabla^2 f_i(x_k)\right)\right] d_k = -h_k\nabla F(x_k)^T F(x_k). \tag{18}$$

Step 3. Update the variables: $x_{k+1} = x_k + d_k$.

Step 4. Test for continuation of iterations. If $\|F(x_{k+1})\| \le \varepsilon$, stop; otherwise set $k = k+1$ and continue with step 2. ■

Therefore, when $f_i(x)$ are convex and positive for all $i = 1,...,m$; $rank(\nabla F(x)) = n$, $\theta = 1$ and $h_k \to \infty$, then the GFA is quadratically convergent to x^*. We see that the algorithm is very simple. The most difficult is step 2, requiring to solve a system of linear equations. But, this is a common step also for Newton, Gauss-Newton and Levenberg-Marquardt algorithms. On the other hand, it is necessary to evaluate the Hessians of functions $f_i(x), i = 1,...,m,$ which turns out to be a difficult task. Observe that, rejecting from (18) the second order terms, we get an equivalent algebraic expression of the Levenberg-Marquardt algorithm for solving least squares minimization problems. In the next section we consider some variants of GF algorithm and prove that the best algorithm corresponding to the gradient flow approach is that which ignore completely the second order information given by the Hessians $\nabla^2 f_i(x_k)$, $i = 1,...,m$.

3. Gradient flow algorithm for nonlinear least squares minimization without second order terms

It is well-known that, when the residuals are very small at the solution, the second-order terms do not contribute significantly to the efficiency of the Levenberg-Marquardt or Gauss-Newton algorithms for solving least squares minimization problems. In this section we prove that rejecting from (18) the second order information we get a more efficient algorithm which is quadratically convergent to a solution of (1). With other words, we prove that any (scalar) approximation of Hessians $\nabla^2 f_i(x_k)$, at point x_k, does not improve the convergence of the algorithm (18).

To see this we present a modification of GFA by considering some scalar approximations of the Hessian matrices $\nabla^2 f_i(x_k)$ of the residual functions $f_i(x), i = 1, ..., m,$ at point x_k. Many approximation schemes could be imagined, here we present one of them.

Suppose that the functions $f_i(x), i = 1, ..., m,$ are convex, and let us consider the point $x_{k+1} = x_k + d_k,$ where d_k is the searching direction. In this point we can write:

$$f_i(x_{k+1}) = f_i(x_k) + \nabla f_i(x_k)^T d_k + \frac{1}{2} d_k^T \nabla^2 f_i(z) d_k,$$

where $z \in [x_k, x_{k+1}]$. Having in view the local character of the searching procedure and that the distance between x_k and x_{k+1} is sufficiently small, we can choose $z = x_{k+1}$ and consider $\gamma_i^{k+1} I$ as an approximation of $\nabla^2 f_i(x),$ in point $x_{k+1},$ where $\gamma_i^{k+1} \in R.$ As we can see this is an anticipative viewpoint in which the approximation of the Hessian of function f_i in point x_{k+1} is computed using the local information from point x_k. Therefore we can write:

$$\gamma_i^{k+1} = \frac{2}{d_k^T d_k} \left[f_i(x_{k+1}) - f_i(x_k) - \nabla f_i(x_k)^T d_k \right]. \tag{19}$$

Since $f_i,\ i = 1, ..., m$ are convex functions, it follows that $\gamma_i^{k+1} \geq 0$ for all $i = 1, ..., m.$ In fact the following proposition can be proved.

Proposition 3.1. *Assume that* $f_i(x)$ *is continuously differentiable and* $\nabla f_i(x)$ *is Lipschitz continuous, with a positive constant* L_i. *Then at point* $x_{k+1}, \gamma_i^{k+1} \leq 2L_i.$

Proof. From (19) we have:

$$\gamma_i^{k+1} = \frac{2\left[f_i(x_k) + \nabla f_i(\xi_k)^T d_k - f_i(x_k) - \nabla f_i(x_k)^T d_k \right]}{\|d_k\|^2},$$

where $\xi_k \in [x_k, x_{k+1}]$. Therefore,

$$\gamma_i^{k+1} = \frac{2\left[\nabla f_i(\xi_k) - \nabla f_i(x_k)\right]^T d_k}{\|d_k\|^2}.$$

Using the inequality of Cauchy and the Lipschitz continuity it follows that

$$\gamma_i^{k+1} \leq \frac{2\|\nabla f_i(\xi_k) - \nabla f_i(x_k)\|}{\|d_k\|} \leq \frac{2L_i \|\xi_k - x_k\|}{\|d_k\|} \leq \frac{2L_i \|x_{k+1} - x_k\|}{\|x_{k+1} - x_k\|} = 2L_i. \blacksquare$$

Therefore, in the current point x_k the following approximation of the Hessian $\nabla^2 \Phi(x_k)$ can be considered:

$$\nabla F(x_k)^T \nabla F(x_k) + \delta_k I, \tag{20}$$

where

$$\delta_k = \sum_{i=1}^{m} f_i(x_k) \gamma_i^k. \tag{21}$$

Observe that if $rank \nabla F(x_k) = n$ and $f_i(x_k) \geq 0,$ then (20) represents a positive definite approximation of $\nabla^2 \Phi(x_k)$.

Using this approximation of Hessians $\nabla^2 f_i(x_k),\ i = 1, ..., m,$ in the current point $x_k,$ and (18) we get the following iterative process:

$$x_{k+1} = x_k - h_k \left[I + h_k\theta\left(\nabla F(x_k)^T\nabla F(x_k) + \delta_k I\right)\right]^{-1}\nabla F(x_k)^T F(x_k), \qquad (22)$$

where δ_k is given by (21).

However, even if $f_i(x)$ is a convex function, due to finite precision of the numerical computations, especially towards the final part of the iterative process, it is possible that γ_i^k be negative, but very small. On the other hand, for nonconvex function it is often possible that γ_i^k be negative. Therefore, in order to have a positive definite approximation of $\nabla^2\Phi(x_k)$ we can consider the following formula for δ_k computation:

$$\delta_k = \sum_{i=1}^{m} f_i(x_k)^2\left(\gamma_i^k\right)^2. \qquad (23)$$

This formula is a little too conservative. Our numerical evidence proved that a large percentage from the number of iterations are characterized by: $f_i(x_k) \geq 0$ and $\gamma_i^k > 0$. Therefore, for computation of δ_k we can consider the following less conservative procedure, which ensures a nonnegative value for δ_k :

Procedure δ (δ_k computation)

```
Set $\delta_k = 0$.
For $i = 1, ..., m$, do:
    Set $p = f_i(x_k)$, $q = \gamma_i^k$.
    If $p < 0$, then $p = f_i(x_k)^2$.
    If $q < 0$, then $q = \left(\gamma_i^k\right)^2$.
    Set $\delta_k = \delta_k + pq$.
End For.
```

Another approximation of the Hessians $\nabla^2 f_i(x_k)$, at point x_k, the simplest one, is to consider $\nabla^2 f_i(x_k) = f_i(x_k)I$. (See [14, pp.210].) Therefore, in this case we have:

$$\delta_k = \sum_{i=1}^{m} f_i(x_k)^2. \qquad (24)$$

We see that using a scalar approximation of the Hessians we get a family of algorithms, given by (22), parametrized by δ_k, where δ_k is computed as in (23), (24) or by means of procedure δ. With this the following algorithm can be presented:

Algorithm MGFA (Modified Gradient Flow Algorithm)

Step 1. Consider the initial point $x_0 \in R^n$, a parameter $\theta \in [0,1]$, a sequence of time step sizes $\{h_k\}$ and an $\varepsilon > 0$ sufficiently small. Compute: $F(x_0)$, $\nabla F(x_0)$ and $\delta_0 = \|F(x_0)\|$. Set $k = 0$.

Step 2. Solve the system of linear equations:

$$\left[I + h_k\theta\left(\nabla F(x_k)^T\nabla F(x_k) + \delta_k I\right)\right] d_k = -h_k\nabla F(x_k)^T F(x_k). \qquad (25)$$

Step 3. Update the variables: $x_{k+1} = x_k + d_k$.

Step 4. Test for continuation of iterations. If $\|F(x_{k+1})\| < \varepsilon$, stop; otherwise set $k = k+1$ and go to step 5.

Step 5. Compute δ_k using (23), procedure δ or (24), and go to step 2. ■

The convergence of MGFA is given by

Theorem 3.1. *Let $\{x_k\}$ be the sequence defined by (22) and x^* a solution of (1) such that: $F(x)$ is twice continuous differentiable, $\nabla F(x)$ is Lipschitz continuous and $rank \nabla F(x^*) = n$. If the initial point x_0 is sufficiently close to x^*, then:*

(i) If $\theta \in [0,1]$ and $h_k > 0$ is sufficiently small, then x_k converges linearly to x^*.

(ii) If $\theta = 1$ and $h_k \to \infty$, then x_k converges quadratically to x^*.

Proof. *(i)* From (22) we have

$$\left[(1 + h_k\theta\delta_k) I + h_k\theta\nabla F(x_k)^T\nabla F(x_k)\right] d_k = -h_k\nabla F(x_k)^T F(x_k).$$

After some algebra we get:

$$x_{k+1} = x_k - \rho_k \left[\nabla F(x_k)^T F(x_k) + \theta\nabla F(x_k)^T\nabla F(x_k)(x_{k+1} - x)\right], \tag{26}$$

where

$$\rho_k = \frac{h_k}{1 + h_k\theta\delta_k}. \tag{27}$$

Subtracting x^* from both sides of the above equality and using the mean value theorem, as in the proof of Theorem 2.3, we have:

$$e_{k+1} = \left\{I - \rho_k \left[I + \rho_k\theta\nabla F(x_k)^T\nabla F(x_k)\right]^{-1} \nabla F(x_k)^T\nabla F(\xi_k)\right\} e_k, \tag{28}$$

where $e_k = x_k - x^*$, $\xi_k \in [x_k, x^*]$. Taking the norm on both sides of this equality we obtain:

$$\|e_{k+1}\| \leq \varphi(x_k, \xi_k, \theta, \rho_k) \|e_k\|, \tag{29}$$

where

$$\varphi(x_k, \xi_k, \theta, \rho_k) = \left\|I - \rho_k \left[I + \rho_k\theta\nabla F(x_k)^T\nabla F(x_k)\right]^{-1} \nabla F(x_k)^T\nabla F(\xi_k)\right\|. \tag{30}$$

From the estimate (29) we see that if $\varphi(x_k, \xi_k, \theta, \rho_k) < 1$, then the first order terms in (29) show that the error e_k converges to zero linearly. Since $rank\nabla F(x^*) = n$ it follows that $\nabla F(x^*)^T\nabla F(x^*)$ is positive definite. Now, when x_k is sufficiently close to x^*, by continuity, (30) implies that if $\theta \in [0,1]$:

$$\varphi(x_k, \xi_k, \theta, \rho_k) \leq \left(1 - \frac{\rho_k\lambda^k_{\text{min}}}{1 + \rho_k\theta\lambda^k_{\text{max}}}\right), \tag{31}$$

where λ^k_{min} and λ^k_{max} are the minimum and maximum eigenvalues of $\nabla F(x_k)^T\nabla F(x_k)$, respectively. Using (27) we see that

$$1 - \frac{\rho_k\lambda^k_{\text{min}}}{1 + \rho_k\theta\lambda^k_{\text{max}}} = 1 - \frac{h_k\lambda^k_{\text{min}}}{1 + h_k\theta(\delta_k + \lambda^k_{\text{max}})} < 1. \tag{32}$$

Therefore, (29) implies that

$$\lim_{k\to\infty} e_k = 0$$

linearly, proving that x_k converges to x^* linearly.

(ii) Considering $\theta = 1$ in (26) we get:

$$\frac{x_{k+1} - x_k}{\rho_k} = -\nabla F(x_k)^T \left[F(x_k) + \nabla F(x_k)(x_{k+1} - x_k)\right]. \tag{33}$$

But

$$\lim_{h_k \to \infty} \frac{h_k}{1 + h_k \delta_k} = \frac{1}{\delta_k}$$

and using, for example, (23) we see that $\lim\limits_{k\to\infty} \delta_k = \lim\limits_{k\to\infty} \sum\limits_{i=1}^{m} f_i(x_k)^2 \left(\gamma_i^k\right)^2 = 0$. (The same is true for procedure δ or (24)) Therefore $\lim\limits_{k\to\infty} \rho_k = \infty$. Having in view that ∇F is of full column rank, the equation (33) reduces to

$$\nabla F(x_k)(x_{k+1} - x_k) + F(x_k) = 0.$$

This coincides with the Newton method applied to $F(x) = 0$, which we know that in conditions of the theorem is quadratically convergent to x^* if the initial point x_0 is sufficiently close to x^*. This completes the proof. ■

An interesting feature of the theorem 3.1 is that the algorithm allows sufficiently large values for h_k when the splitting parameter satisfies $\theta = 1$. In this case, x_k converges quadratically to a local solution of (1). On the other hand, we see that h_k is acting on the both members of (25) as a scaling parameter, this ensuring the numerical stability of the system (25).

Remark 3.1. From (29) and (31) with (32), for $\theta = 1$, we have:

$$\|e_{k+1}\| \le p_{k+1} \|e_0\|, \tag{34}$$

where

$$p_{k+1} = \prod_{i=0}^{k} \left(1 - \frac{h_i \lambda_{\min}^i}{1 + h_i(\delta_i + \lambda_{\max}^i)}\right) \tag{35}$$

But, $\nabla F(x_i)^T \nabla F(x_i)$ is a positive definite matrix, therefore for all $i = 0, 1, ..., k$,

$$0 < 1 - \frac{h_i \lambda_{\min}^i}{1 + h_i(\delta_i + \lambda_{\max}^i)} < 1.$$

So, p_k, for all k, is a decreasing sequence in $(0, 1)$, i.e. p_k is convergent to zero. ■

In order to see the complexity of the MGF algorithm let us denote:

$$a_i = \frac{h_i \lambda_{\min}^i}{1 + h_i(\delta_i + \lambda_{\max}^i)} \tag{36}$$

and consider $a_j = \min\{a_i : 0 \le i \le k\}$. Then

$$p_{k+1} = \prod_{i=0}^{k} (1 - a_i) \le (1 - a_j)^{k+1}, \tag{37}$$

i.e.

$$\|e_{k+1}\| \leq p_{k+1} \|e_0\| \leq (1-a_j)^{k+1} \|e_0\| .$$

Thus, the number of iterations required to obtain an accuracy $\|e_{k+1}\| = \|x_{k+1} - x^*\| \leq \varepsilon,$ starting from the initial point $x_0,$ is bounded by

$$\frac{\log(\varepsilon) - \log(\|e_0\|)}{\log(1-a_j)} - 1. \tag{38}$$

We see that this expression depends on the final accuracy, on the initial estimation x_0 of the solution, as well as on the distribution of the eigenvalues of the Hessians of the residual functions f_i, $i = 1, ..., m,$ along the trajectory of the ordinary differential equation (5).

Remark 3.2. Considering $\delta_k = 0$ in MGFA (22) algorithm we get *another algorithm* which is very close to that of Levenberg-Marquardt:

$$x_{k+1} = x_k - h_k \left[I + h_k\theta\nabla F(x_k)^T \nabla F(x_k)\right]^{-1} \nabla F(x_k)^T F(x_k), \tag{39}$$

for which, like in theorem 3.1, for $\theta = 1,$ we can prove that $\|e_{k+1}\| \leq \bar{p}_{k+1} \|e_0\|$, where

$$\bar{p}_{k+1} = \prod_{i=0}^{k} \left(1 - \frac{h_i\lambda_{\min}^i}{1 + h_i\lambda_{\max}^i}\right),$$

and, as above, for $i = 0, ..., k,$ $\lambda_{\min}^i$ and $\lambda_{\max}^i$ are the minimum and maximum eigenvalues of $\nabla F(x_i)^T \nabla F(x_i),$ respectively. ■

Theorem 3.2. *In the family of algorithms given by (22), the Levenberg-Marquardt algorithm, which correspond to* $\theta = 1$ and $\delta_k = 0,$ *is the best one.*

Proof. Having in view that $h_k > 0$, it is very easy to see that (39), with $\theta = 1,$ can be written as:

$$\left[\frac{1}{h_k} I + \nabla F(x_k)^T \nabla F(x_k)\right] d_k = -\nabla F(x_k)^T F(x_k),$$

which is the Levenberg-Marquardt algorithm (3), with $\mu_k = 1/h_k$. Now, since $\nabla F(x_i)^T \nabla F(x_i)$ is a positive definite matrix and $\delta_i \geq 0$ in (22), it follows that for all $i = 0, 1, ..., k,$

$$\frac{h_i\lambda_{\min}^i}{1 + h_i(\delta_i + \lambda_{\max}^i)} \leq \frac{h_i\lambda_{\min}^i}{1 + h_i\lambda_{\max}^i}.$$

Therefore, using (35) we see that $p_{k+1} \geq \bar{p}_{k+1}$. Hence, for $\delta_k = 0,$ the convergence of (22) with $\theta = 1$, i.e. the convergence of (39) with $\theta = 1,$ is more rapid. ■

Therefore in this interpretation we get the Levenberg-Marquardt algorithm as a simple particularization of MGF algorithm.

4. Numerical examples

In order to see the performances of the MGFA, in the sequel, we present some numerical experiments obtained with a Fortran implementation of the MGFA. In this respect the algorithm MGFA has been implemented with the following values for δ_k :

a) $\delta_k = \sum_{i=1}^{m} f_i(x_k)^2 \left(\gamma_i^k\right)^2$, b) δ_k given by procedure δ,

c) $\delta_k = \sum_{i=1}^{m} f_i(x_k)^2$, d) $\delta_k = 0$.

In all numerical experiments we have considered $\theta = 1$. The tolerance ε is chosen to be 10^{-7},(see step 4 of the algorithm). The time step size h_k is considered constant, the same for all k. At the same time, we have considered another set of experiments in which $h_k = 1/\left\|F(x_k)\right\|^2$.

Example 1. (*Equillibrium Combustion*) [27]

$$f_1(x) = x_1x_2 + x_1 - 3x_5,$$
$$f_2(x) = 2x_1x_2 + x_1 + 3r_{10}x_2^2 + x_2x_3^2 + r_7x_2x_3 + r_9x_2x_4 + r_8x_2 - rx_5,$$
$$f_3(x) = 2x_2x_3^2 + r_7x_2x_3 + 2r_5x_3^2 + r_6x_3 - 8x_5,$$
$$f_4(x) = r_9x_2x_4 + 2x_4^2 - 4rx_5,$$
$$f_5(x) = x_1x_2 + x_1 + r_{10}x_2^2 + x_2x_3^2 + r_7x_2x_3 + r_9x_2x_4 + r_8x_2 + r_5x_3^2 + r_6x_3 + x_4^2 - 1,$$

where

$r = 10$	$r_5 = 0.193$	$r_6 = 4.10622e-4$
$r_7 = 5.45177e-4$	$r_8 = 4.4975e-7$	$r_9 = 3.40735e-5$
$r_{10} = 9.615e-7$		

The following initial points have been considered:

x_0^1	x_0^2	x_0^3	x_0^4
1	1	1	21
0	1	1	1
10.15	10.15	10.15	10.15
5.5	0.5	0.5	1.5
0.05	0.05	10.05	1.05

.

The following tables give the number of iterations necessary to get a solution corresponding to different selections of δ_k and h_k, starting the algorithm from different initial points.

Table 1a ($\delta_k = \sum_{i=1}^{m} f_i(x_k)^2 \left(\gamma_i^k\right)^2$)

	$h_k = 10^6$	$h_k = 10^7$	$h_k = 10^8$	$h_k = 10^9$	$h_k = 10^{10}$	$h_k = 1/\left\|F(x_k)\right\|^2$
x_0^1	670	115	63	57	57	328
x_0^2	752	197	144	139	138	403
x_0^3	702	148	94	89	88	403
x_0^4	719	164	111	105	105	790

Table 1b (δ_k given by procedure δ)

	$h_k = 10^6$	$h_k = 10^7$	$h_k = 10^8$	$h_k = 10^9$	$h_k = 10^{10}$	$h_k = 1/\|F(x_k)\|^2$
x_0^1	653	100	48	43	41	314
x_0^2	658	105	52	48	47	310
x_0^3	657	104	51	46	46	362
x_0^4	809	133	68	62	59	724

Table 1c ($\delta_k = \sum_{i=1}^{m} f_i(x_k)^2$)

	$h_k = 10^6$	$h_k = 10^7$	$h_k = 10^8$	$h_k = 10^9$	$h_k = 10^{10}$	$h_k = 1/\|F(x_k)\|^2$
x_0^1	927	370	315	310	309	595
x_0^2	921	364	309	304	303	582
x_0^3	973	416	361	356	355	684
x_0^4	1784	809	712	702	701	1373

Table 1d ($\delta_k = 0$)

	$h_k = 10^6$	$h_k = 10^7$	$h_k = 10^8$	$h_k = 10^9$	$h_k = 10^{10}$	$h_k = 1/\|F(x_k)\|^2$
x_0^1	768	87	19	12	11	309
x_0^2	632	74	18	14	14	303
x_0^3	632	74	18	14	14	355
x_0^4	632	74	18	14	14	701

The following solutions have been obtained:

x^*	x^*	x^*
0.00311411	0.002471	0.0027567
34.592169	43.87876	39.248218
0.0650419	0.0577847	-0.0613849
0.859378	-0.860205	0.859724
0.0369518	0.0369655	0.0369851

.

Example 2. (*Steady-state solution for reaction rate equations*) [32]

$$\begin{aligned} f_1(x) &= 1 - x_1 - k_1 x_1 x_6 + r_1 x_4, \\ f_2(x) &= 1 - x_2 - k_2 x_2 x_6 + r_2 x_5, \\ f_3(x) &= -x_3 + 2k_3 x_4 x_5, \\ f_4(x) &= k_1 x_1 x_6 - r_1 x_4 - k_3 x_4 x_5, \\ f_5(x) &= 1.5(k_2 x_2 x_6 - r_2 x_5) - k_3 x_4 x_5, \\ f_6(x) &= 1 - x_4 - x_5 - x_6, \end{aligned}$$

where

$k_1 = 31.24$	$k_2 = 0.272$	$k_3 = 303.03$
$r_1 = 2.062$	$r_2 = 0.02$	

The following initial points have been considered:

x_0^1	x_0^2	x_0^3	x_0^4
1.09	1.19	2.19	0.05
1.05	1.15	3.15	0.99
0.05	0.05	0.05	0.05
0.99	0.99	0.99	0.99
0.05	0.05	0.05	0.05
0	0.09	1.09	0.09

.

Table 2a ($\delta_k = \sum_{i=1}^{m} f_i(x_k)^2 \left(\gamma_i^k\right)^2$)

	$h_k = 10$	$h_k = 10^2$	$h_k = 10^3$	$h_k = 10^4$	$h_k = 10^5$	$h_k = 1/\left\|F(x_k)\right\|^2$
x_0^1	21	12	10	10	10	10
x_0^2	21	12	10	9	9	10
x_0^3	244	234	232	232	232	237
x_0^4	137	127	125	125	125	126

Table 2b (δ_k given by procedure δ)

	$h_k = 10$	$h_k = 10^2$	$h_k = 10^3$	$h_k = 10^4$	$h_k = 10^5$	$h_k = 1/\left\|F(x_k)\right\|^2$
x_0^1	19	10	9	8	8	8
x_0^2	19	10	8	8	8	8
x_0^3	154	145	143	143	143	148
x_0^4	192	181	179	179	179	188

Table 2c ($\delta_k = \sum_{i=1}^{m} f_i(x_k)^2$)

	$h_k = 10$	$h_k = 10^2$	$h_k = 10^3$	$h_k = 10^4$	$h_k = 10^5$	$h_k = 1/\left\|F(x_k)\right\|^2$
x_0^1	16	7	5	5	5	5
x_0^2	17	7	6	5	5	6
x_0^3	24	14	12	12	12	16
x_0^4	21	11	9	9	9	9

Table 2d ($\delta_k = 0$)

	$h_k = 10$	$h_k = 10^2$	$h_k = 10^3$	$h_k = 10^4$	$h_k = 10^5$	$h_k = 1/\left\|F(x_k)\right\|^2$
x_0^1	14	5	3	3	3	5
x_0^2	15	5	4	4	4	5
x_0^3	18	6	5	5	5	12
x_0^4	17	6	5	5	5	9

The following solution has been obtained:

x^*
0.974243
0.982829
0.0515124
0.935671
0.90839e-4
0.06423807

.

Example 3 (*Circuit design problem*) [31]

$$f_k(x) = (1 - x_1x_2)x_3 \left\{\exp\left[x_5\left(g_{1k} - g_{3k}x_7 10^{-3} - g_{5k}x_8 10^{-3}\right)\right] - 1\right\} - g_{5k} + g_{4k}x_2,\ k = 1, ..., 4,$$
$$f_{k+4}(x) = (1 - x_1x_2)x_4 \left\{\exp\left[x_6\left(g_{1k} - g_{2k} - g_{3k}x_7 10^{-3} - g_{4k}x_9 10^{-3}\right)\right] - 1\right\} - g_{5k}x_1 + g_{4k},\ k = 1, ..., 4,$$
$$f_9(x) = x_1x_3 - x_2x_4,$$

where

$$g = \begin{bmatrix} 0.4850 & 0.7520 & 0.8690 & 0.9820 \\ 0.3690 & 1.2540 & 0.7030 & 1.4550 \\ 5.2095 & 10.0677 & 22.9274 & 20.2153 \\ 23.3037 & 101.7790 & 111.4610 & 191.2670 \\ 28.5132 & 111.8467 & 134.3884 & 211.4823 \end{bmatrix}.$$

The following initial points have been considered:

x_0^1	x_0^2	x_0^3	x_0^4
0.7	0.65	0.75	0.75
0.5	0.45	0.45	0.45
0.9	0.8	0.9	0.9
1.9	1.8	1.77	1.77
8.1	8.5	8.5	8.9
8.1	8.5	7.5	7.9
5.9	5.9	5.5	5.5
1	1.1	1.25	1.35
1.9	1.5	1.88	1.88

.

Table 3a ($\delta_k = \sum_{i=1}^{m} f_i(x_k)^2 \left(\gamma_i^k\right)^2$)

	$h_k = 10$	$h_k = 10^2$	$h_k = 10^3$	$h_k = 10^4$	$h_k = 10^5$	$h_k = 1/\left\|F(x_k)\right\|^2$
x_0^1	142	50	40	38	38	27
x_0^2	173	60	47	45	45	56
x_0^3	256	146	133	131	131	132
x_0^4	600	500	489	487	487	488

Table 3b (δ_k given by procedure δ)

	$h_k = 10$	$h_k = 10^2$	$h_k = 10^3$	$h_k = 10^4$	$h_k = 10^5$	$h_k = 1/\|F(x_k)\|^2$
x_0^1	123	32	22	20	20	21
x_0^2	151	39	26	24	24	26
x_0^3	218	108	96	94	94	94
x_0^4	497	397	386	384	384	385

Table 3c ($\delta_k = \sum_{i=1}^{m} f_i(x_k)^2$)

	$h_k = 10$	$h_k = 10^2$	$h_k = 10^3$	$h_k = 10^4$	$h_k = 10^5$	$h_k = 1/\|F(x_k)\|^2$
x_0^1	113	22	12	10	10	11
x_0^2	140	27	15	12	12	14
x_0^3	135	25	13	11	11	12
x_0^4	124	24	14	12	11	13

Table 3d ($\delta_k = 0$)

	$h_k = 10$	$h_k = 10^2$	$h_k = 10^3$	$h_k = 10^4$	$h_k = 10^5$	$h_k = 1/\|F(x_k)\|^2$
x_0^1	108	10	6	4	4	10
x_0^2	132	16	7	5	4	12
x_0^3	129	19	6	5	5	11
x_0^4	46	15	6	5	5	11

The following solution has been obtained:

x^*
0.8999999
0.4499875
1.000006
2.00006
7.99997
7.99969
5.00003
0.99998
2.00005

.

Example 4 (*Robot kinematics problem*) [24]

$$f_1(x) = 0.004731x_1x_3 - 0.3578x_2x_3 - 0.1238x_1 + x_7 - 0.001637x_2 - 0.9338x_4 - 0.3571,$$
$$f_2(x) = 0.2238x_1x_3 + 0.7623x_2x_3 + 0.2638x_1 - x_7 - 0.07745x_2 - 0.6734x_4 - 0.6022,$$
$$f_3(x) = x_6x_8 + 0.3578x_1 + 0.004731x_2,$$
$$f_4(x) = -0.7623x_1 + 0.2238x_2 + 0.3461,$$
$$f_5(x) = x_1^2 + x_2^2 - 1,$$
$$f_6(x) = x_3^2 + x_4^2 - 1,$$
$$f_7(x) = x_5^2 + x_6^2 - 1,$$

$$f_8(x) = x_7^2 + x_8^2 - 1.$$

The following initial points have been considered:

x_0^1	x_0^2	x_0^3	x_0^4
0.164	0.14	-0.15	-1
-0.98	0.98	0.98	1
-0.94	0.94	-0.94	-1
-0.32	0.32	0.32	1
-0.99	0.99	-0.97	-1
-0.056	0.056	0.056	1
0.41	0.41	-0.44	-1
-0.91	-0.91	0.99	1

.

Table 4a ($\delta_k = \sum_{i=1}^{m} f_i(x_k)^2 \left(\gamma_i^k\right)^2$)

	$h_k = 10$	$h_k = 10^2$	$h_k = 10^3$	$h_k = 10^4$	$h_k = 10^5$	$h_k = 1/\left\|F(x_k)\right\|^2$
x_0^1	9	4	3	3	3	3
x_0^2	10	6	5	5	5	5
x_0^3	13	8	7	7	6	7
x_0^4	15	11	10	10	9	12

Table 4b (δ_k given by procedure δ)

	$h_k = 10$	$h_k = 10^2$	$h_k = 10^3$	$h_k = 10^4$	$h_k = 10^5$	$h_k = 1/\left\|F(x_k)\right\|^2$
x_0^1	9	5	4	3	3	4
x_0^2	11	7	6	6	6	6
x_0^3	13	9	8	8	8	8
x_0^4	16	11	10	10	10	14

Table 4c ($\delta_k = \sum_{i=1}^{m} f_i(x_k)^2$)

	$h_k = 10$	$h_k = 10^2$	$h_k = 10^3$	$h_k = 10^4$	$h_k = 10^5$	$h_k = 1/\left\|F(x_k)\right\|^2$
x_0^1	9	4	3	3	3	3
x_0^2	11	6	5	5	5	6
x_0^3	13	8	7	7	7	8
x_0^4	18	13	12	12	12	16

Table 4d ($\delta_k = 0$)

	$h_k = 10$	$h_k = 10^2$	$h_k = 10^3$	$h_k = 10^4$	$h_k = 10^5$	$h_k = 1/\left\Vert F(x_k)\right\Vert^2$
x_0^1	9	4	3	3	3	3
x_0^2	10	6	5	5	5	5
x_0^3	11	7	6	6	6	7
x_0^4	14	9	9	9	9	12

The following solutions have been obtained:

x^*	x^*	x^*	x^*
0.164431	0.671554	0.671563	0.671554
-0.986388	0.740955	0.741005	0.740955
-0.947063	0.951893	-0.651582	-0.651590
-0.321045	-0.306431	-0.758578	-0.758578
-0.998233	0.963810	-0.962545	0.962793
0.059418	0.266587	-0.271124	0.271124
0.411033	0.404641	-0.437592	-0.437592
-0.911620	-0.914475	0.899181	-0.899181

.

Example 5. (*Quadratic residual functions*)

$$f_1(x) = x_1^2 - 1,$$

$$f_i(x) = (x_{i-1} + x_i)^2 - i, \, i = 2, ..., n.$$

Considering the initial point $x_0 = [1, ..., 1]^T$, the following results are obtained.

Table 5a ($\delta_k = \sum_{i=1}^{m} f_i(x_k)^2 \left(\gamma_i^k\right)^2$)

n	$h_k = 10$	$h_k = 10^2$	$h_k = 10^3$	$h_k = 10^4$	$h_k = 10^5$	$h_k = 1/\left\Vert F(x_k)\right\Vert^2$
100	176	43	28	25	25	613
150	274	57	34	30	28	1600
200	378	71	38	34	32	3152

Table 5b (δ_k given by procedure δ)

n	$h_k = 10$	$h_k = 10^2$	$h_k = 10^3$	$h_k = 10^4$	$h_k = 10^5$	$h_k = 1/\left\Vert F(x_k)\right\Vert^2$
100	246	114	99	96	95	681
150	409	192	169	165	164	1732
200	586	279	247	242	241	3357

Table 5c ($\delta_k = \sum_{i=1}^{m} f_i(x_k)^2$)

n	$h_k = 10$	$h_k = 10^2$	$h_k = 10^3$	$h_k = 10^4$	$h_k = 10^5$	$h_k = 1/\|F(x_k)\|^2$
100	746	614	599	597	596	1179
150	1824	1607	1584	1581	1580	3145
200	3473	3166	3134	3130	3129	6242

Table 5d ($\delta_k = 0$)

n	$h_k = 10$	$h_k = 10^2$	$h_k = 10^3$	$h_k = 10^4$	$h_k = 10^5$	$h_k = 1/\|F(x_k)\|^2$
100	155	23	8	6	6	596
150	249	32	9	7	7	1580
200	350	42	11	7	7	3129

Example 6. (*Quadratic residual functions*)

$$f_1(x) = x_1^2 - 1,$$
$$f_i(x) = (x_i - \sum_{j=1}^{i-1} x_j)^2 - 1, i = 2, ..., n.$$

Considering the initial point $x_0 = [0.1, ..., 0.1]^T$, the following results are obtained.
Table 6b (δ_k given by procedure δ)

n	$h_k = 1$	$h_k = 10$	$h_k = 10^2$	$h_k = 10^3$	$h_k = 10^4$	$h_k = 1/\|F(x_k)\|^2$
100	20	16	16	16	16	18
200	30	26	25	25	25	27
300	27	23	23	23	23	26

Table 6c ($\delta_k = \sum_{i=1}^{m} f_i(x_k)^2$)

n	$h_k = 1$	$h_k = 10$	$h_k = 10^2$	$h_k = 10^3$	$h_k = 10^4$	$h_k = 1/\|F(x_k)\|^2$
100	15	12	11	11	11	12
200	16	14	13	13	13	14
300	18	15	14	14	14	16

Table 6d ($\delta_k = 0$)

n	$h_k = 1$	$h_k = 10$	$h_k = 10^2$	$h_k = 10^3$	$h_k = 10^4$	$h_k = 1/\|F(x_k)\|^2$
100	9	7	7	7	7	11
200	10	8	8	8	8	12
300	10	9	9	9	9	14

Example 7.

$$f_1(x) = x_1^2 - 1,$$
$$f_i(x) = (x_i^2 - x_{i-1})^2 - i, i = 2, ..., n.$$

Considering the initial point $x_0 = [3.2, ..., 3.2]^T$, the following results are obtained.

Table 7b (δ_k given by procedure δ)

n	$h_k = 1$	$h_k = 10$	$h_k = 10^2$	$h_k = 10^3$	$h_k = 10^4$	$h_k = 1/\left\|F(x_k)\right\|^2$
100	26	20	19	19	19	30
200	85	79	78	78	78	138
300	173	167	166	166	166	360

Table 7c ($\delta_k = \sum_{i=1}^{m} f_i(x_k)^2$)

n	$h_k = 1$	$h_k = 10$	$h_k = 10^2$	$h_k = 10^3$	$h_k = 10^4$	$h_k = 1/\left\|F(x_k)\right\|^2$
100	28	23	21	21	21	32
200	79	73	72	72	71	131
300	214	208	207	207	207	401

Table 7d ($\delta_k = 0$)

n	$h_k = 1$	$h_k = 10$	$h_k = 10^2$	$h_k = 10^3$	$h_k = 10^4$	$h_k = 1/\left\|F(x_k)\right\|^2$
100	13	8	7	7	7	21
200	13	8	7	7	7	71
300	13	8	7	7	7	207

5. Conclusion

In this paper we proposed a gradient flow approach for solving the nonlinear least squares problem. This is based on the integration of an ordinary differential equation for which a discretization thechique with a splitting parameter has been considered. It has been shown that the solution of the discretized problem converges to a local solution of the problem either linearly or quadratically as a function of the choice of the spliting parameter and the size of the discretization step. When the size of the discretization step tends to infinit, then the convergence is quadratic. The main result of the paper shows that when the second order information, given by the Hessian of the residual functions of the problem, is not considered into the algorithm, then we get the best algorithm based on gradient flow approach, which is an equivalent algebraic expression of the Levenberg-Marquardt algorithm, for which we prove its quadratic convergence. Numerical experiments with different strategies for scalar approximation of the Hessian matrices of the residual functions of the problem show that the most efficient variant of the algorithm is that correspondig to the case when the second order information is not considered into the algorithm.

While the well known Levenberg-Marquardt algorithm is in no way optimal motivated, but just only a heuristic (as a combination of the gradient descent and Gauss-Newton iterations, or as a trust-region algorithm), the paper represents a reasonable justification of the Levenberg-Marquardt as a result of the gradient flow method for nonlinear least squares problem.

References

1. F. Aluffi-Pentini, V. Parisi and F. Zirilli, ”A differential equations algorithm for nonlinear equations”. ACM Trans. Math. Software, vol.10, pp. 299-316, 1984.

2. F. Aluffi-Pentini, V. Parisi and F. Zirilli, ”Algorithm 617, DAFNE: a differential-equations algorithm for nonlinear equations”. ACM Trans. Math. Software, vol.10, pp. 317-324, 1984.

3. N. Andrei, ”Gradient flow algorithm for unconstrained optimization”. ICI Technical Report, April, 2004.

4. W. Behrman, ”An efficient gradient flow method for unconstrained optimization”. Dissertation, Stanford University, June 1998.

5. P.T. Boggs, ”An algorithm, based on singular perturbation theory, for illconditioned minimization problems”. SIAM J. Numer. Anal., vol.14, pp.830-843, 1977.

6. C.A. Botsaris, ”Differential gradient methods”. J. Math. Anal. Applics., vol.63, pp.177-198, 1978

7. C.A. Botsaris, ”A curvilinear optimisation method based upon iterative estimation of the eigensystem of the Hessian matrix”. J. Math. Anal., Applics., vol.63, pp.396-411, 1978.

8. C.A. Botsaris, ”A class of methods for unconstrained minimization based on stable numerical integration techniques”. J. Math. Anal. Applics., vol.63, pp.729-749, 1978.

9. C.A. Botsaris and D.H. Jacobson, ”A Newton-type curvilinear search method for optimization”. J. Math. Anal. Applics., vol.54, pp.217-229, 1976.

10. A.A. Brown and M.C. Bartholomew-Biggs, ”ODE vs SQP methods for constrained optimization”. Technical Report No. 179, The Hatfield Polytechnic, Numercial Optimisation Centre, June 1987.

11. A.A. Brown and M.C. Bartholomew-Biggs, ”Some effective methods for unconstrained optimization based on the solution of systems of ordinary differential equations”. J. Optim. Theory Appl., vol.62, pp.211-224, 1989.

12. R. Courant, ”Variational methods for the solution of problems of equilibrium and vibrations”. Bull. Amer. Math. Soc., vol.49, pp.1-23, 1943.

13. H. Dan, N. Yamashita and M. Fukushima, ”Convergence properties of the inexact Levenberg-Marquardt method under local error bound conditions”. Technical Report, Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Kyoto, Japan, January 16, 2001.

14. J.E. Dennis and R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

15. Y.G. Evtushenko, ”Two numerical methods of solving nonlinear programming problems”. Sov. Math. Dokl., vol.15, pp.420-423, 1974.

16. Y.G. Evtushenko, Numerical Optimization Techniques. Optimization Software, Inc., New York, 1985.

17. Y.G. Evtushenko and V.G. Zhadan, ”The space tranformation techniques in mathematical programming”. in Lecture Notes in Control and Information Science, 180, *System Modeling and Optimizatio*n. Proc. of the 15th IFIP Conference, P. Kall (Ed.), Zurich, Springer-Verlag, pp.292-300, 1992.

18. Y.G. Evtushenko and V.G. Zhadan, ”Stable barrier-projection and barrier Newton

methods in linear programming". Computational Optimization and Applications, vol.3, pp.289-303, 1994.

19. Y.G. Evtushenko and V.G. Zhadan, "Stable barrier-projection and barrier Newton methods in nonlinear programming".Optimization Methods and Software, vol.3, pp.237-256, 1994.

20. J.Y. Fan and Y.X. Yuan, "On the convergence of a new Levenberg-Marquardt method". Thechnical Report, AMSS, Chinese Academy of Sciences, 2001.

21. J. Hadamard, "Mémoire sur le problème d'analyse relatif à l'équilibre des plaques élastiques encastrées". Mémoires présenté par divers savants à l'Académie des Sciences de l'Institut National de France, Series 2, vol.33, pp.1-128, 1908.

22. U. Helmke and J.B. Moore, Optimization and Dynamical Systems. Springer Verlag, 1994.

23. M.W. Hirsch and S. Smale, Differential Equations, Dynamical Systems, and Linear Algebra. Academic Press, New York, 1974.

24. R. Kearfott and M. Novoa, "INTBIS, a portable interval Newton bisection package". ACM Trans. Math. Soft., vol.16, pp.152-157, 1990.

25. K. Levenberg, "A method for the solution of certain problems in least squares". Quart. Appl. Math. vol.2, pp.164-168, 1944.

26. D. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters". SIAM J. Appl. Math. vol.11, pp.431-441, 1963.

27. K. Meintjes and A.P. Morgan, "Chemical-equilibrium systems as numerical test problems". ACM Trans. Math. Soft., vol.16, pp.143-151, 1990.

28. J.J. Moré, "The Levenberg-Marquardt algorithm: implementation and theory". in *Numerical Analysis*, G.A. Watson (Ed.) Lecture Notes in Mathematics, vol. 630, pp.105-116, Springer Verlag, 1977.

29. M.R. Osborne, "Nonlinear least squares - the Levenberg algorithm revisited". J. Austral. Math. Soc. vol.19, pp.343-357, 1976.

30. M.J.D. Powell, "Convergence properties of a class of minimization algorithms". in *Nonlinear Programming 2*, O. Mangasarian, O., R. Meyer and S. Robinson, (Eds.) Academic Press, pp.1-27, 1975.

31. H. Ratschek and J. Rokne, "A circuit design problem". J. Global Optimization, vol.3, pp.501, 1993.

32. M. Shacham, "Numerical solution of constrained nonlinear algebraic equations". Int. J. Numer. Meth. in Eng, vol.23, pp.1455-1481, 1986.

33. G.V. Smirnov, "Convergence of barrier-projection methods of optimization via vector Lyapunov functions". Optimization Methods and Software, vol.3, pp.153-162, 1994.

34. J.A. Snyman, "A new and dynamic method for unconstrained optimisation". Appl. Math. Modelling., vol.6, pp.449-462, 1982.

35. F. Verhulst, Nonlinear Differential Equations and Dynamical Systems. Springer Verlag, Berlin, 1990.

36. J-Ph. Vial. and I. Zang. "Unconstrained optimization by approxiamtion of the gradient path". Math. Oper. Res., vol.2, pp. 253-265, 1977.

37. S. Wang, X.Q. Yang and K.L. Teo, "A unified gradient flow approach to constrained nonlinear optimization problems". Computational Optimization and Applications, vol.25,

pp.251-268, 2003.

38. N. Yamashita and M. Fukushima, ”On the rate of convergence of the Levenberg-Marquardt method”. Technical Report, Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Kyoto, Japan, October 28, 2000.

39. I. Zang, ”A new arc algorithm for unconstrained optimization”. Math. Programming, vol.15, pp.36-52, 1978.

40. F. Zirilli, S. Incerti and V. Parisi, ”A new method for solving nonlinear simultaneous equations”. SIAM J. Numer. Anal., vol.16, pp.779-789, 1979.

41. F. Zirilli, S. Incerti and F. Aluffi-Pentini, ”Systems of equations and a stable integration of second order ODE's”. in *Numerical Optimisation and Dynamical systems* (Eds. L.C.W. Dixon and G.P. Szego), Elsevier North Hollanmd, pp.289-307, 1987.

July 5, 2004