Gradient Flow Algorithm for Unconstrained Optimization

Neculai Andrei[1]

Abstract. The gradient flow approach for unconstrained optimization is presented. The idea of this approach is that for an unconstrained optimization problem an ordinary differential equation to associate. Fundamentally this is a gradient system based on the first order optimality conditions of the problem. Using a discretization scheme, based on a two level implicit time discretization technique, with a splitting parameter $\theta \in [0,1]$, an algorithm is obtained. The convergence of this algorithm is analyzed and it is shown that this is linearly convergent when $0 \leq \theta < 1$ and quadratically convergent when $\theta = 1$ and the integration step is sufficiently large. At every step a linear algebraic system is solved, but no linear search is required. Numerical experiments on a limited number of test functions show that this approach based on integration of a gradient system is more robust and even superior to the Newton's method.

Key words. Unconstrained optimization, gradient flow method, quadratic convergence.

1. Introduction

Unconstrained optimization, expressed as $\min f(x)$, where $f(x)$ is a real function, is one of the most active area in optimization community, virtually appearing in every human activity. For solving these problems many efficient methods have been suggested. Excellent presentations of these methods can be found, for example, in (Refs. 1-5). The most useful algorithms classify in: the conjugate gradient method and its variants, the Newton method and its extensions, the BFGS variable metric method and its limited memory variants, the truncated Newton method and the Nelder-Mead simplex method. All these methods consider iterations of the form

$$x_{k+1} = x_k + \lambda_k d_k,$$

where d_k is a descent search direction and λ_k is a steplength obtained by a one-dimensional search. The conjugate - gradient methods consider the search direction as $d_k = -\nabla f(x_k) + \beta_k d_{k-1}$, where the scalar β_k is chosen in such a manner that the method reduces to the linear conjugate gradient when the function is quadratic and the line search is exact. The rest of methods defines the search direction by $d_k = -B_k^{-1}\nabla f(x_k)$, where B_k is a nonsingular symmetric matrix. Mainly, the matrix B_k is selected as: $B_k = I$ (the steepest descent method), $B_k = \nabla^2 f(x_k)$ (the Newton's method) or an approximation of the Hessian $\nabla^2 f(x_k)$ (BFGS, DFP, SR1 etc.).

[1] Research Institute for Informatics, 8-10, Averescu Avenue, Bucharest, Romania, E-mail: nandrei@u3.ici.ro

Other methods, different from those above mentioned, are the gradient flow methods, known as stable barrier-projection and barrier-Newton methods, which have been introduced for the first time by Evtushenko (Refs. 6,7) and by Evtushenko and Zhadan (Refs. 8-10). Convergence of these methods via Lyapunov functions has been considered by Smirnov (Ref. 10). Recently, improvements and some computational experience with these methods have been considered by Wang, Yang and Teo (Ref. 12). Basically, in this approach a constrained optimization problem is reformulated as an ordinary differential equation (ODE) in such a way that the solution of this ODE converges to an equilibrium point of the optimization problem as parameter t from the ODE goes to ∞.

In this paper, a gradient flow approach for unconstrained optimization is proposed. To a general unconstrained optimization problem an ODE is associated. Basically, this ODE is a gradient system. For integration of this ODE a discretization scheme, based on a two level implicit time discretization scheme with a splitting parameter θ is proposed. It is shown that the solution of this discretized gradient flow equation converges to a local minimum of the original problem, either linearly or quadratically according to the choice of θ and of the time step size of the series of the discretization time points. When $\theta = 1$ and the time step size tends to ∞, then the corresponding algorithm is quadratically convergent.

The structure of the paper is as follows. Section 2 is dedicated to present the idea of the gradient flow approach for unconstrained optimization. In section 3 we present the corresponding gradient flow algorithm, as well as its convergence analysis. Some numerical experiments and comparisons with the Newton method with backtracking are given in section 4.

2. The gradient flow approach for unconstrained optimization

Consider the following unconstrained optimization problem

$$\min f(x), \tag{1}$$

where $x \in R^n$ and $f : R^n \to R$ is a function assumed to be twice continuous differentiable. As we know, a necessary condition for the point x^* be an optimal solution for (1) is:

$$\nabla f(x^*) = 0. \tag{2}$$

This is a system of n nonlinear equations, which must be solved to get the optimal solution x^*. In order to fulfill this optimality condition the following continuous gradient flow reformulation of the problem is suggested: solve the following ordinary differential equation:

$$\frac{dx(t)}{dt} = -\nabla f(x(t)) \tag{3}$$

with the initial condition

$$x(0) = x_0. \tag{4}$$

The following theorem gives the convergence result of the system (3) initialized with (4).

Theorem 2.1. *Consider that x^* is a point satisfying (2). Suppose that $\nabla^2 f(x^*)$ is positive definite. If x_0 is close enough to x^*, then $x(t)$, solution of (3), tends to x^* as t goes to ∞.*

Proof. The system (3) can be written as

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

where $\Phi(x) = -\nabla f(x)$. To show that x^* is an assymptotically stable point for (3) we shall consider the Poincaré - Lyapunov theory (Ref. 13). According to this theory, x^* is an asymptotically stable point for the nonlinear differential equation system $\dot{x} = \Phi(x)$ if $\Phi(x)$ is continuously differentiable and the linearized system

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

where $y = x - x^*$, is exponentially stable, i.e. all eigenvalues of $\nabla\Phi(x^*)$ are strictly negative. Considering the Taylor's expansion of $\Phi(x)$ around x^*,and having in view that $\nabla f(x^*) = 0$, we get:

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

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

The following theorem shows that $f(x(t))$ is strictly decreasing along the solution of (3).

Theorem 2.2. *Let $x(t)$ be the solution of (3). For a fixed $t_0 \geq 0$ if $\nabla f(x(t)) \neq 0$ for all $t > t_0$, then $f(x(t))$ is strictly decreasing with respect to t, for all $t > t_0$.*

Proof. We have:

$$\begin{aligned} \frac{df(x(t))}{dt} &= \nabla f(x(t))^T \frac{dx(t)}{dt} \\ &= -\nabla f(x(t))^T \nabla f(x(t)) \\ &= -\left\|\nabla f(x(t))\right\|_2^2. \end{aligned}$$

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

Observe that the ordinary differential equation (3), associated to (1), is a gradient system (Ref. 14, pp.199). Gradient systems have special properties that make their flows very simple. For gradient system (3) at regular points x, characterized by the fact that $\nabla f(x) \neq 0$, the trajectories cross level surfaces of the function $f(x)$ orthogonally. Non-regular points are equilibria of the system, and if x^* is an isolated minimum of $f(x)$, then

x^* is an asymptotically stable equilibrium of the gradient system (3).

3. The gradient flow algorithm for unconstrained optimization

As we have seen, solving the unconstrained optimization problem (1) has been reduced to that of integration of the ordinary differential equation (3) with initial condition (4). Now we shall consider a discretization of this ODE 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 (3):

$$\frac{x_{k+1} - x_k}{h_k} = -\left[(1-\theta)\nabla f(x_k) + \theta \nabla f(x_{k+1})\right], \tag{5}$$

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

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

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,

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

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

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

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

$$x_{k+1} = x_k - h_k\left[I + h_k\theta\nabla^2 f(x_k)\right]^{-1}\nabla f(x_k), \tag{6}$$

for any $\theta \in [0,1]$. Considering x_0 as the initial guess, then (6) defines a series $\{x_k\}$. The convergence of (6) is given by

Theorem 3.1. *Let $\{x_k\}$ be the sequence defined by (6) and x^* a solution of (1), such that $\nabla^2 f(x^*)$ is positive definite. If the initial point x_0 is close enough 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 (6) we have:

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

hence:

$$x_{k+1} = x_k - h_k\left[\nabla f(x_k) + \theta\nabla^2 f(x_k)(x_{k+1} - x_k)\right]. \tag{7}$$

Substracting x^* from both sides of (7) and having in view that $e_k = x_k - x^*$, $x_{k+1} - x_k = e_{k+1} - e_k$ and $\nabla f(x^*) = 0$, we get:

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

Now using the mean value theorem we have:

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

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

$$e_{k+1} = \left\{I - h_k \left[I + h_k \theta \nabla^2 f(x_k)\right]^{-1} \nabla^2 f(\xi_k)\right\} e_k. \tag{8}$$

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

$$\|e_{k+1}\| \leq \varphi(x_k, \xi_k, \theta, h_k) \|e_k\|, \tag{9}$$

where

$$\varphi(x_k, \xi_k, \theta, h_k) = \left\| I - h_k \left[I + h_k \theta \nabla^2 f(x_k)\right]^{-1} \nabla^2 f(\xi_k) \right\|. \tag{10}$$

From (9) we see that if $\varphi(x_k, \xi_k, \theta, h_k) < 1$, then e_k converges to zero linearly. With these, selecting $\xi_k = x_k$, we can write:

$$\varphi(x_k, \xi_k, \theta, h_k) \leq \left(1 - \frac{h_k \lambda_{\min}^k}{1 + h_k \theta \lambda_{\max}^k}\right) < 1, \tag{11}$$

where $\lambda_{\min}^k$ and $\lambda_{\max}^k$ represents the minimum and the maximum eigenvalues of $\nabla^2 f(x_k)$, respectively. Therefore, from (9) it follows that $\lim_{k\to\infty} e_k = 0$ linearly, i.e. $x_k \to x^*$ linearly.

(ii) Consider $\theta = 1$ in (7), we get:

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

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

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

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

Remark 3.1. From (9) and (11), 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, $\nabla^2 f(x_i)$ is positive definite, therefore for all $i = 1, \ldots k$,

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

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 $i = 1, \ldots k$,

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

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

Based on (6), for solving (1), the following algorithm can be presented:

Algorithm GFUO (Gradient Flow Unconstrained Optimization)

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

Step 2. Solve for δx_k the system

$$\left[I + h_k\theta\nabla^2 f(x_k)\right]\delta x_k = -h_k\nabla f(x_k). \tag{12}$$

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

Step 4. Test for continuation of iterations. If $\|\nabla f(x_k)\| \le \varepsilon$, stop; otherwise set $k = k+1$ and continue with step 2.

As the theorem 3.1 recommends, the GFUO algorithm 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 of the system (12) to be positive definite. The following theorem suggests how to choose the value h_k of time distances between two successive time points.

Theorem 3.2. *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* $\nabla^2 f(x_k)$, *then* $\left[I + h_k\nabla^2 f(x_k)\right]$ *is positive definite.*

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

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

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

In order to see the complexity of the GFUO algorithm let us denote

$$a_i = \frac{h_i\lambda_{\min}^i}{1+h_i\lambda_{\max}^i}$$

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

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

i.e. $\|e_{k+1}\| \le p_{k+1}\|e_0\| \le (1-a_j)^{k+1}\|e_0\|$. Thus, the number of iterations required to obtain an accuracy $\|e_{k+1}\| = \|x_{k+1} - x^*\| \le \varepsilon$, starting at point x_0, is bounded by

$$\frac{\log_2 \varepsilon - \log_2 \|e_0\|}{\log_2(1-a_j)} - 1.$$

This expression depends on the final accuracy, on the initial estimation of the optimal point, as well as on the distribution of eigenvalues of the Hessian matrices of function f along the iterations.

4. Numerical experiments

In order to see the performances of the algorithm GFUO we present some numerical experiments obtained with a Fortran implementation of it. In all experiments we have considered $\theta = 1$. The tolerance ε is chosen to be 10^{-7}. The time step size h_k is considered the same for all k, equal with h. The GFUO algorithm is compared with the Newton algorithm with backtracking:

$$x_{k+1} = x_k - \lambda_k \nabla^2 f(x_k)^{-1} \nabla f(x_k),$$

where the steplength λ_k is chosen to satisfy the first Wolfe condition:

$$f(x_k + \lambda_k d_k) \leq f(x_k) + \alpha \lambda_k \nabla f(x_k)^T d_k,$$

by backtracking, starting with $\lambda_k = 1$ and reducing this value as $\lambda_k = \lambda_k s$, where $\alpha = 0.0001$, $s = 0.8$ and $d_k = -\nabla^2 f(x_k)^{-1} \nabla f(x_k)$.

Example 1. (Full matrix)

$$f(x) = \sum_{i=1}^{n} i x_i^2 + \frac{1}{100} \left(\sum_{i=1}^{n} x_i \right)^2.$$

Considering $n = 1000$, table 1 gives the number of iterations subject to different initial points and values for time step size h.

Table 1. Number of iterations for different initial points and h (GFUO and Newton).

$\mathbf{n = 1000}$	$\mathbf{h = 1}$	$\mathbf{h = 10}$	$\mathbf{h = 10^2}$	$\mathbf{h = 10^3}$	**Newton**
$x_0 = [0.5, \ldots, 0.5]$	15	6	4	3	2
$x_0 = [10.5, \ldots, 10.5]$	18	7	4	3	2
$x_0 = [100.5, \ldots, 100.5]$	20	7	5	3	2
$x_0 = [1000.5, \ldots, 1000.5]$	22	8	5	4	2

Example 2. (Diagonal)

$$f(x) = \sum_{i=1}^{n} \frac{i}{10} \left(\exp(x_i) - x_i \right).$$

Table 2. Number of iterations for different initial points and h (GFUO and Newton).

$\mathbf{n = 1000}$	$\mathbf{h = 1}$	$\mathbf{h = 10}$	$\mathbf{h = 10^2}$	$\mathbf{h = 10^3}$	**Newton**
$x_0 = [1, \ldots, 1]$	141	21	7	5	5
$x_0 = [10, \ldots, 10]$	154	30	17	15	15
$x_0 = [100, \ldots, 100]$	244	120	107	105	105

Example 3. (Tridiagonal)

$$\begin{aligned} f(x) \quad = \quad & \left((5 - 3x_1 - x_1^2)x_1 - 3x_2 + 1\right)^2 + \\ & \sum_{i=2}^{n-1} \left((5 - 3x_i - x_i^2)x_i - x_{i-1} - 3x_{i+1} + 1\right)^2 + \\ & \left((5 - 3x_n - x_n^2)x_n - x_{n-1} + 1\right)^2, \end{aligned}$$

Table 3. Number of iterations for different initial points and h (GFUO and Newton).

$\mathbf{n = 1000}$	$\mathbf{h = 1}$	$\mathbf{h = 10}$	$\mathbf{h = 10^2}$	$\mathbf{h = 10^3}$	**Newton**
$x_0 = [-1, \ldots, -1]$	8	6	5	5	5
$x_0 = [-10, \ldots, -10]$	18	15	14	14	14
$x_0 = [-100, \ldots, -100]$	28	26	25	25	25
$x_0 = [1, \ldots, 1]$	11	8	7	6	6
$x_0 = [10, \ldots, 10]$	20	17	16	15	15
$x_0 = [100, \ldots, 100]$	30	27	26	25	25

Example 4. (Full matrix)

$$f(x) = \sum_{i=1}^{n-1} (x_i - 1)^2 + \left(\sum_{j=1}^{n} x_j^2 - 0.25\right)^2 .$$

For $n = 100$ and $n = 200$ the algorithms GFUO and Newton with backtracking give the results in table 4a and 4b.

Table 4a. Number of iterations for different initial points and h (GFUO and Newton).

$\mathbf{n = 100}$	$\mathbf{h = 1}$	$\mathbf{h = 10}$	$\mathbf{h = 10^2}$	$\mathbf{h = 10^3}$	**Newton**
$x_0 = [1, 2, \ldots, n]$	23	20	19	19	19
$x_0 = [1/10, 2/10, \ldots, n/10]$	17	14	14	13	13
$x_0 = [1 * 10, 2 * 10, \ldots, n * 10]$	29	26	25	25	25

Table 4b. Number of iterations for different initial points and h (GFUO and Newton).

$\mathbf{n = 200}$	$\mathbf{h = 1}$	$\mathbf{h = 10}$	$\mathbf{h = 10^2}$	$\mathbf{h = 10^3}$	**Newton**
$x_0 = [1, 2, \ldots, n]$	25	22	22	21	21
$x_0 = [1/10, 2/10, \ldots, n/10]$	19	17	16	16	16
$x_0 = [1 * 10, 2 * 10, \ldots, n * 10]$	30	28	27	27	27

Example 5. (Full matrix)

$$f(x) = (x_1 - 3)^2 + \sum_{i=2}^{n} \left(x_1 - 3 - 2\left(x_1 + x_2 + \ldots x_i\right)^2 \right)^2 .$$

Table 5a. Number of iterations for different initial points and h (GFUO and Newton).

$\mathbf{n = 50}$	$\mathbf{h = 10}$	$\mathbf{h = 10^2}$	$\mathbf{h = 10^3}$	$\mathbf{h = 10^4}$	**Newton**
$x_0 = [0.001, \ldots, 0.001]$	2348	398	140	18	8
$x_0 = [0.01, \ldots, 0.01]$	6390	327	86	78	14
$x_0 = [0.1, \ldots, 0.1]$	4406	320	68	48	25
$x_0 = [1, \ldots, 1]$	1043	166	74	65	41
$x_0 = [10, \ldots, 10]$	1055	177	85	76	52

Table 5b. Number of iterations for different initial points and h (GFUO and Newton).

$\mathbf{n = 100}$	$\mathbf{h = 10}$	$\mathbf{h = 10^2}$	$\mathbf{h = 10^3}$	$\mathbf{h = 10^4}$	**Newton**
$x_0 = [0.001, \ldots, 0.001]$	4668	1690	85	111	11
$x_0 = [0.01, \ldots, 0.01]$	11873	999	106	35	16
$x_0 = [0.1, \ldots, 0.1]$	1049	158	72	61	38
$x_0 = [1, \ldots, 1]$	1069	179	93	82	50
$x_0 = [10, \ldots, 10]$	1081	189	103	91	62

Example 6. (Bidiagonal)

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

Considering the initial point $x_0 = [-1.2, 1, \cdots, -1.2, 1]$, the following results are obtained:

Table 6. Number of iterations for different values of h (GFUO and Newton).

$\mathbf{n}$	$\mathbf{h = 1}$	$\mathbf{h = 10}$	$\mathbf{h = 10^2}$	$\mathbf{h = 10^3}$	**Newton**
50	48	15	14	14	9
100	48	15	14	14	9
1000	48	15	14	14	11
2000	48	15	14	14	11
3000	48	15	14	14	12
5000	48	15	14	14	12

Example 7. (Powel function)

$$\begin{aligned} f(x) \quad = \quad & \sum_{i=1}^{n/4} \left(x_{4i-3} + 10x_{4i-2}\right)^2 + 5\left(x_{4i-1} - x_{4i}\right)^2 + \\ & \left(x_{4i-2} - 2x_{4i-1}\right)^4 + 10\left(x_{4i-3} - x_{4i}\right)^4 . \end{aligned}$$

Considering the initial point $x_0 = [3, -1, 0, 1, \ldots, 3, -1, 0, 1]$, the following results are obtained:

Table 7. Number of iterations for different values of h (GFUO and Newton).

n	$\mathbf{h = 10^2}$	$\mathbf{h = 10^3}$	$\mathbf{h = 10^4}$	$\mathbf{h = 10^5}$	**Newton**
400	715	87	25	21	17
800	897	105	27	21	18
1200	1024	118	29	21	18
1600	1126	129	30	21	18
2000	1211	137	31	22	18

Example 8. (Extended Rosenbrock function)

$$f(x) = \sum_{i=1}^{n/2} c\left(x_{2i} - x_{2i-1}^2\right)^2 + \left(1 - x_{2i-1}\right)^2 .$$

For the initial point $x_0 = [-1.2, 1, \ldots, -1.2, 1]$, and $c = 100$ the following results are obtained:

Table 8a. Number of iterations for different values of h (GFUO and Newton).

n	$\mathbf{h = 10}$	$\mathbf{h = 10^2}$	$\mathbf{h = 10^3}$	$\mathbf{h = 10^4}$	**Newton**
2	15	9	7	7	20
100	16	9	7	7	20
1000	17	10	8	7	20
2000	17	10	8	7	20

For $c = 1000$ we get the following results:

Table 8b. Number of iterations for different values of h (GFUO and Newton).

n	$\mathbf{h = 10}$	$\mathbf{h = 10^2}$	$\mathbf{h = 10^3}$	$\mathbf{h = 10^4}$	**Newton**
2	16	9	7	7	38
100	17	9	7	7	39
1000	18	10	7	7	39
2000	18	10	7	7	39

For $c = 10000$ we get the results:

Table 8c. Number of iterations for different values of h (GFUO and Newton).

n	$\mathbf{h = 10}$	$\mathbf{h = 10^2}$	$\mathbf{h = 10^3}$	$\mathbf{h = 10^4}$	**Newton**
2	16	9	7	6	78
100	18	10	7	6	79
1000	18	10	7	7	79
2000	19	10	7	7	79

Example 9. (White and Holst function)

$$f(x) = \sum_{i=1}^{n/2} c\left(x_{2i} - x_{2i-1}^3\right)^2 + \left(1 - x_{2i-1}\right)^2.$$

For the initial point $x_0 = [-1.2, 1, \ldots, -1.2, 1]$, and $c = 100$ the following results are obtained:

Table 9a. Number of iterations for different values of h (GFUO and Newton).

$\mathbf{n}$	$\mathbf{h = 10}$	$\mathbf{h = 10^2}$	$\mathbf{h = 10^3}$	$\mathbf{h = 10^4}$	**Newton**
2	19	11	7	6	26
100	21	11	8	7	26
1000	22	12	8	7	26
2000	22	12	8	7	26

For $c = 1000$ we get the following results:

Table 9b. Number of iterations for different values of h (GFUO and Newton).

$\mathbf{n}$	$\mathbf{h = 10}$	$\mathbf{h = 10^2}$	$\mathbf{h = 10^3}$	$\mathbf{h = 10^4}$	**Newton**
2	20	11	8	7	51
100	21	12	8	7	52
1000	22	12	8	7	52
2000	23	12	8	7	52

For $c = 10000$ we get the results:

Table 9c. Number of iterations for different values of h (GFUO and Newton).

$\mathbf{n}$	$\mathbf{h = 10}$	$\mathbf{h = 10^2}$	$\mathbf{h = 10^3}$	$\mathbf{h = 10^4}$	**Newton**
2	21	12	8	7	108
100	22	12	8	7	108
1000	23	13	9	7	108
2000	24	13	9	7	108

Example 10. (Arrowhead function)

$$f(x) = \sum_{i=1}^{n-1} \left(x_i^2 + x_n^2\right)^2 - 4x_i + 3.$$

For the initial point $x_0 = [1, 1, \ldots, 1, 1]$ the following results are obtained:

Table 10. Number of iterations for different values of h (GFUO and Newton).

n	h = 10	h = 10^2	h = 10^3	h = 10^4	Newton
10	7	6	6	6	7
100	7	6	6	6	7
1000	7	6	6	6	7
2000	7	6	6	6	7

Example 11. (Arrowhead-Bidiagonal function)

$$f(x) = (x_1 - x_2)^2 + \sum_{i=2}^{n-1} (x_{i-1} + x_i + x_n)^4 + (x_{n-1} - x_n)^2.$$

For the initial point $x_0 = [1, -1, \ldots, 1, -1]$ the following results are obtained:

Table 11. Number of iterations for different values of h (GFUO and Newton).

n	h = 10^4	h = 10^5	h = 10^6	h = 10^7	Newton
10	17	17	17	17	18
100	19	19	19	19	20
1000	27	20	21	21	22
2000	29	21	21	21	22

Example 12. (Engval function)

$$f(x) = \sum_{i=2}^{n-1} \left\{ (x_{i-1}^2 + x_i^2)^2 - 4x_{i-1} + 3 \right\}.$$

For the initial point $x_0 = [2, \ldots, 2]$ the following results are obtained:

Table 12. Number of iterations for different values of h (GFUO and Newton).

n	h = 10	h = 10^2	h = 10^3	h = 10^4	Newton
10	10	8	8	8	9
100	10	8	8	8	9
1000	10	8	8	8	9
5000	10	8	8	8	9

5. Conclusion

In this paper we proposed a gradient flow approach of an unconstrained optimization problem, which basically is a particularization of the gradient flow approach of the nonlinear equality constrained optimization problem by Wang, Yang and Teo (Ref. 12). The corresponding algorithm is based on a two level implicit time discretization scheme with a splitting parameter θ of an ordinary differential equation associated to the original unconstrained optimization problem. It is shown that the algorithm converges to a local minimum of the optimization problem either linearly or quadratically, depending on the value of the splitting parameter. Numerical experiments show that the algorithm is comparable with that of Newton's, for some ill-conditioned problems being superior. At every step it is necesarry

to solve an $n-$dimensional algebraic system of linear equations, but unlike the Newton's method no linear search is required.

References.

1. Andrei, N., *Advanced Mathematical Programming, Theory, Computational Methods, Applications,* Technical Press, Bucharest, Romania, 1999.

2. Gill, Ph.E., Murray, W. and Wright, M.H., *Practical Optimization.* Academic Press, London, New York, 1981.

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

4. Nocedal, J., *Theory of algorithms for unconstrained optimization*. Acta Numerica, pp.199-242, 1992.

5. Boyd, S. and Vandenberghe, L., *Convex Optimization*. Cambridge University Press, 2003.

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

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

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

9. Evtushenko, Y.G. and Zhadan, V.G., *Stable barrier-projection and barrier Newton methods in linear programming.* Computational Optimization and Applications, vol.3, pp.289-303, 1994.

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

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

12. Wang, S., Yang, X.Q. and Teo, K.L., *A unified gradient flow approach to constrained nonlinear optimization problems.* ComputationalOptimization and Applications, vol.25, pp.251-268, 2003.

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

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