Newton's method

Dominik Koszkul
Michał Oleszczyk

May 23, 2015

1 Introduce

1.1 Formulating optimization problem

Problem, which needs to be solved, is to find minimum point in set of non-linear, multidimensional function. If function has more than one minimum, algorithm is looking for the nearest local minimum from the initial point. In this project, method to find these special points is *Newton's method*. This algorithm allows to find local minimum points, but this particular method can be optimized by using other algorithm to find optimal step, which is used in the main optimization program. Thanks to this, whole algorithm can find the solution in a faster way. The objective function is presented in the following way

$$\min_{x \epsilon \mathbb{R}^n} f(x), \tag{1}$$

where $n \leqslant 5$. Newton's method is a gradient method without constraints. This means that minimum point is searched in $\mathbb{R}^n$.

1.2 Newton's method

Newton's method is a local optimization type algorithm. The algorithm is presented in the following steps:

1. compute $f(x^i)$ and gradient $\nabla f(x^i)$,
2. compute direction $d^i = -H^{-1} \nabla f(x^i)$,
3. compute step length in selected direction (use method which counts minimum in selected direction),
4. check stop criteria (if any criteria are fulfilled then stop and show results),
5. reply all from step 1 with new counted point x^{i+1}.

To implement the method, stop criteria should be also known. In this case we have three main stop criteria for Newton's method and one additional that ensures that program stops computing after exceeding maximum iterations number.

1.2.1 Stop criteria

1. $< gradf(x^i) \cdot gradf(x^i) > \leqslant \varepsilon_1$
   Scalar product of the squared gradient of the function in point x^i should be less or equal to ε_1,
2. $||x^i - x^{i-1}|| \leqslant \varepsilon_2$
   Euclidean norm of the distance between x^i and x^{i-1} points should be less or equal to ε_2,
3. $|f(x^i) - f(x^{i-1})| \leqslant \varepsilon_3$
   Difference between values of the function in x^i and x^{i-1} should be less or equal to ε_3,
4. maximum number of iterations.

If one from these four criteria is satisfied program stops computing and presents results.

1.2.2 Convergence

Newton's method has a quadratic convergence - convergence rank equals two. This means that, if assumptions are fulfilled, the error decreases in a quadratic way along with the iteration number. In fact the convergence not always is. When the starting point is too far from minimum point the method may be divergent.

1.2.3 Method constraints

Minimum in selected direction method
Pure Newton's method after computing the direction has set step length which is equal to 1.0. When the starting point is far from the local minimum the method will have huge amount of iterations and this is main reason of long computing time. In our implementation we used method, which can find the best step length in current iteration. This algorithm is *bisection method with Goldstein test*. We chose this algorithm because in our program we have to compute the gradient and this method also uses the gradient to compute step length. Moreover this is one of the best method for Newton's algorithm. We can also set precision and thank to this manipulate speed of whole program.

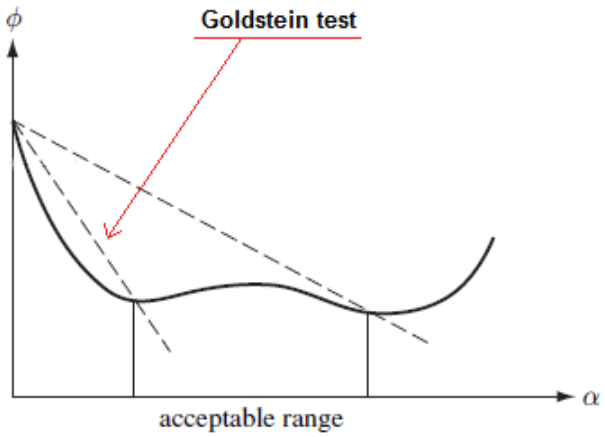

Figure 1: Goldstein test example [1]

Before starting algorithm we have to set a acceptable range $[\tau_l, \tau_r]$ and test coefficient β. For correct work of the algorithm it is crucial to find proper acceptable range. If it is too small whole algorithm will be computing very slowly due to small step in every iteration. Otherwise if the range is too wide, the method can set to big step and pass the minimum point every time. These values can be set by user before starting computations. *Bisection method with Goldstein test* is represented in the following steps:

1. compute direction derivative and step coefficient $\tau_r > 0$,
2. designate $\tau = \frac{1}{2}(\tau_l + \tau_r)$ and compute $f(x^i + \tau d)$, where d – selected direction,
3. if $f(x^i + \tau d) < f(x^i) + (1 - \beta)p\tau$ then $\tau_l \to \tau$,
4. if $f(x^i + \tau d) > f(x^i) + \beta p\tau$ then $\tau_r \to \tau$ and comeback to second step
5. if third and forth step are not fulfilled stop computing

In our implementation we decided that maximum iteration number in this algorithm will be 10. This constraint is sufficient and allows to find optimal step in each Newton's method iteration.

2 General information about the program

The program was written in *Matlab* environment. We use this program because we know its capabilities very well. Thanks to using *Matlab* we do not need use any external libraries to parse an input (function formula) or plot function / particular steps. To handle any input formula we used internal library, which is integrated with *Matlab* and can find predefined symbols in function – *symbolic math toolbox*. To design and create graphic user interface we used *GUIDE* library, which is also delivered with *Matlab*.

3 Rules for setting initial data

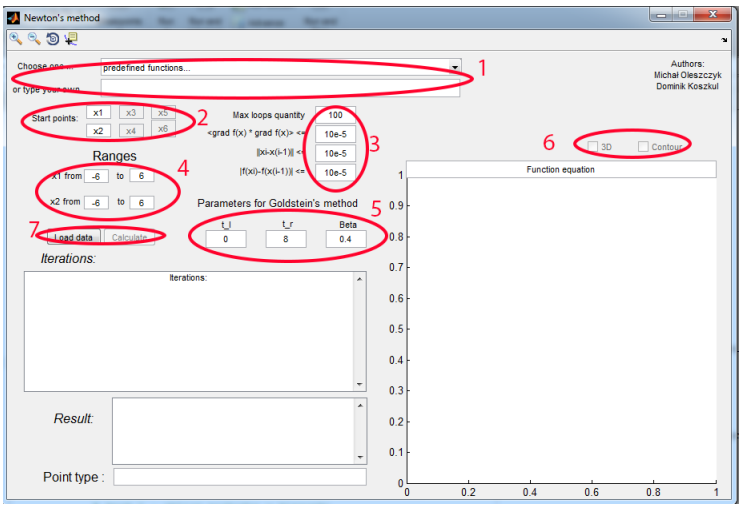

Figure 2: GUI of our program

4 Examples

4.1 Function with four local minima

Function equation:

$$y = x_1^4 + x_2^4 - 0.62x_1^2 - 0.62x_2^2 \tag{2}$$

Function figures:

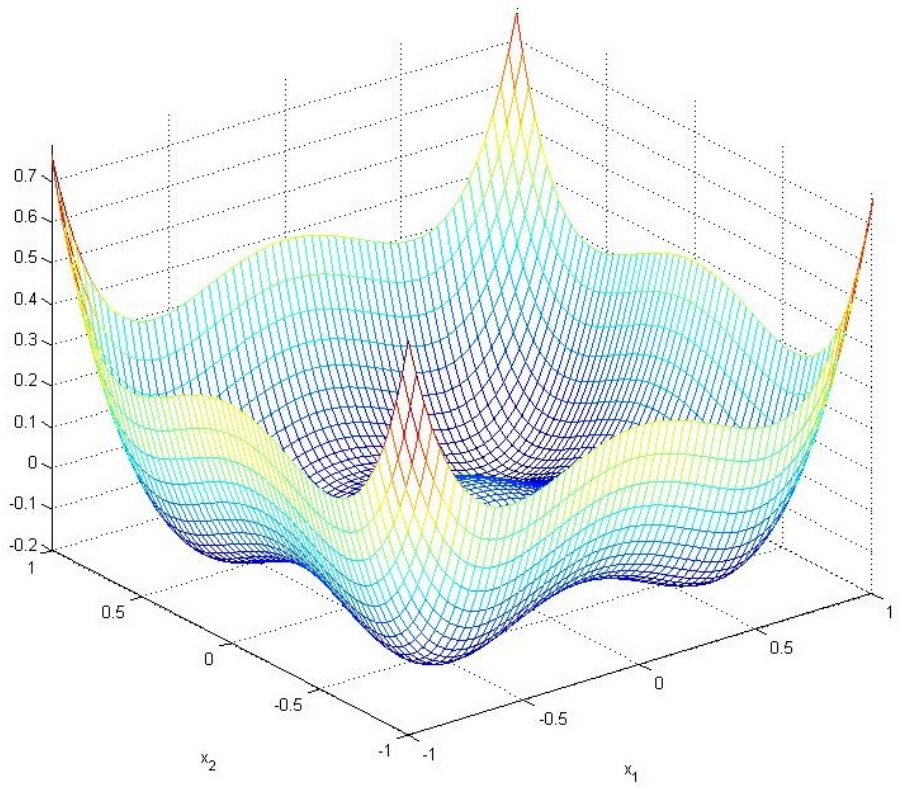

Figure 3: Analyzed function 3D view.

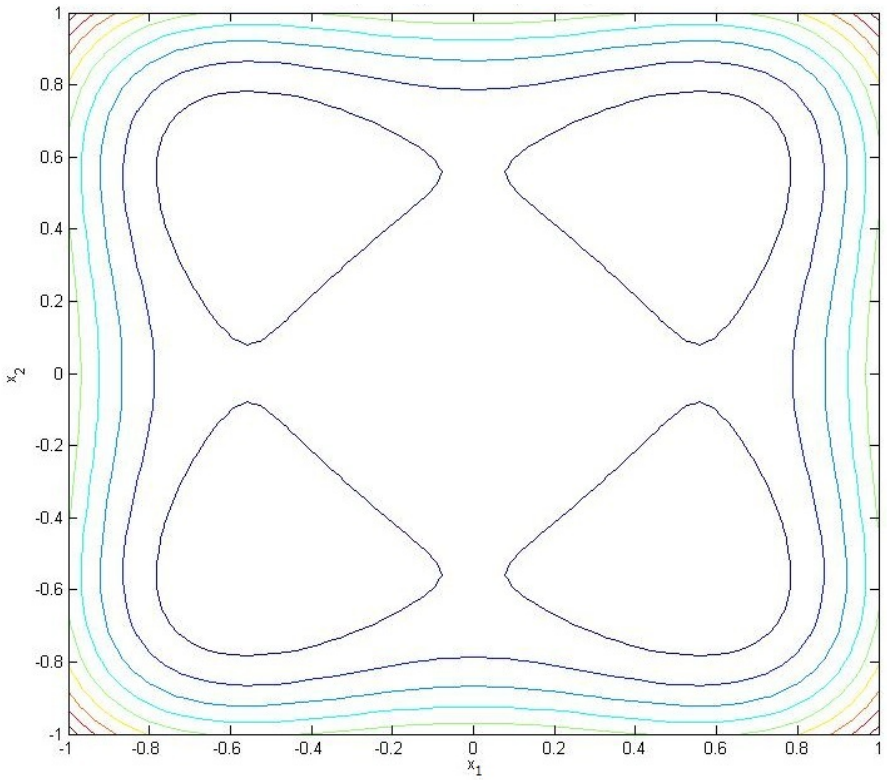

Figure 4: Analyzed function contour view.

Results:

iteration	point coordinates	function value	C_1 value	C_2 value	C_3 value
0	$1, 1$	0.76	0.132	-	-
1	$0.743, 0.743$	-0.0743	0.132	0.363	0.834
2	$0.61, 0.61$	-0.185	0.0358	0.189	0.11
3	$5.63 \cdot 10^{-1}, 5.63 \cdot 10^{-1}$	$-1.92 \cdot 10^{-1}$	$4.36 \cdot 10^{-3}$	$6.6 \cdot 10^{-2}$	$7.5 \cdot 10^{-3}$
4	$5.63 \cdot 10^{-1}, 5.63 \cdot 10^{-1}$	$-1.92 \cdot 10^{-1}$	$7.35 \cdot 10^{-5}$	$6.6 \cdot 10^{-2}$	$7.5 \cdot 10^{-3}$

iteration	point coordinates	function value	C_1 value	C_2 value	C_3 value
0	$-1, 1$	0.76	0.132	-	-
1	$-0.743, 0.743$	-0.0743	0.132	0.363	0.834
2	$-0.61, 0.61$	-0.185	0.0358	0.189	0.11
3	$-5.63 \cdot 10^{-1}, 5.63 \cdot 10^{-1}$	$-1.92 \cdot 10^{-1}$	$4.36 \cdot 10^{-3}$	$6.6 \cdot 10^{-2}$	$7.5 \cdot 10^{-3}$
4	$-5.63 \cdot 10^{-1}, 5.63 \cdot 10^{-1}$	$-1.92 \cdot 10^{-1}$	$7.35 \cdot 10^{-5}$	$6.6 \cdot 10^{-2}$	$7.5 \cdot 10^{-3}$

iteration	point coordinates	function value	C_1 value	C_2 value	C_3 value
0	$-1, -1$	0.76	0.132	-	-
1	$-0.743, -0.743$	-0.0743	0.132	0.363	0.834
2	$-0.61, -0.61$	-0.185	0.0358	0.189	0.11
3	$-5.63 \cdot 10^{-1}, -5.63 \cdot 10^{-1}$	$-1.92 \cdot 10^{-1}$	$4.36 \cdot 10^{-3}$	$6.6 \cdot 10^{-2}$	$7.5 \cdot 10^{-3}$
4	$-5.63 \cdot 10^{-1}, -5.63 \cdot 10^{-1}$	$-1.92 \cdot 10^{-1}$	$7.35 \cdot 10^{-5}$	$6.6 \cdot 10^{-2}$	$7.5 \cdot 10^{-3}$

iteration	point coordinates	function value	C_1 value	C_2 value	C_3 value
0	$1, -1$	0.76	0.132	-	-
1	$0.743, -0.743$	-0.0743	0.132	0.363	0.834
2	$0.61, -0.61$	-0.185	0.0358	0.189	0.11
3	$5.63 \cdot 10^{-1}, -5.63 \cdot 10^{-1}$	$-1.92 \cdot 10^{-1}$	$4.36 \cdot 10^{-3}$	$6.6 \cdot 10^{-2}$	$7.5 \cdot 10^{-3}$
4	$5.63 \cdot 10^{-1}, -5.63 \cdot 10^{-1}$	$-1.92 \cdot 10^{-1}$	$7.35 \cdot 10^{-5}$	$6.6 \cdot 10^{-2}$	$7.5 \cdot 10^{-3}$

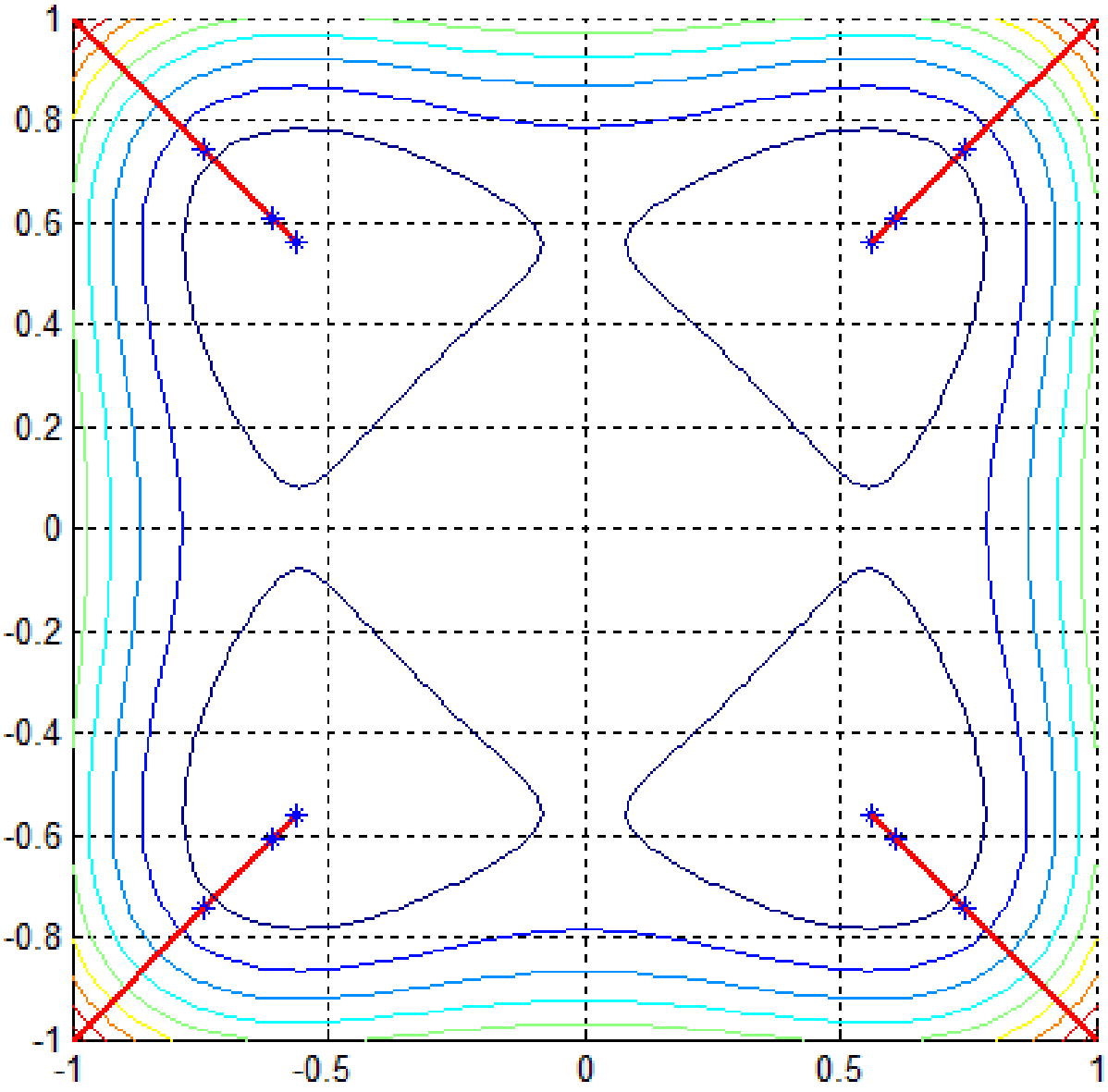

Figure 5: Location of four minima.

References

[1] http://math.stackexchange.com/questions/873467/goldstein-test-in-nonlinear-programming