A two-dimensional search for a Gauss-Newton algorithm

A.B. Forbes
National Physical Laboratory, Teddington, United Kingdom
M.C. Bartholomew-Biggs
University of Hertfordshire, Hatfield, United Kingdom

Abstract
This paper describes a fall-back procedure for use with the Gauss-Newton method for non-linear least-squares problems. While the basic Gauss-Newton algorithm is often successful, it is well-known that it can sometimes generate poor search directions and exhibit slow convergence. For dealing with such situations we suggest a new two-dimensional search strategy. Numerical experiments indicate that the proposed technique can be effective.

1 Introduction

This report describes a method for non-linear least-squares calculations whose main ideas are explained in detail in section 2 below. Basically it uses a standard Gauss-Newton algorithm with the additional distinctive feature that a two-dimensional search procedure is used on any iteration when the Gauss-Newton step proves unsatisfactory. The effectiveness of this fall-back 2D search strategy is illustrated by numerical results presented in section 3.

The approach is implemented as a fortran90 code `GN_solver` which is intended to be suitable for a wide range of least-squares problems (such as those encountered on a day-to-day basis at the National Physical Laboratory). A convenient and flexible user-interface is provided by means of a *harness* routine [1] which is not prescriptive about the form of the function evaluation or the algebraic details of the calculation of the Gauss-Newton search direction. Hence a user is free to exploit special features of a particular problem such as sparsity or structure.

When an iterative algorithm is implemented in software intended for general use an important issue is the robustness of the termination tests. It is desirable, as far as possible, that the performance of the algorithm should be insensitive to changes of scaling on the variables or the function (such as might occur due to a change of physical units). The stopping rules in `GN_solver` are intended to fulfil this requirement – although they do require some tolerance parameters to be specified which reflect the user's own insight into acceptable solution accuracy. Section 3.2 gives some experimental evidence about changes in behaviour of `GN_solver` due to rescaling of problems.

The concluding section outlines some further developments of the algorithmic ideas in `GN_solver` with a view to future versions of the software.

2 GN_solver

`GN_solver` is an implementation of a Gauss-Newton algorithm for nonlinear least-squares problems of the form

$$\text{Minimize} \quad F(x) = \sum_{i=1}^{m} f_i(x)^2. \tag{1}$$

`GN_solver` requires a user-supplied subroutine which, for any values of the optimization variables $x_1,..,x_n$ evaluates the subfunctions $f_1,..,f_m$ and also the Jacobian matrix J with elements $J_{ij} = \partial f_i / \partial x_j$.

The interface between the user-routine and `GN_solver` is a *harness* subroutine (as described in [1]). This calls the user's routine and – according to a flag set by `GN_solver` – it returns either
(i) the values of the subfunctions f_i
or (ii) the values $f_1,..,f_m$ and the elements of gradient $g = 2J^T f$
or (iii) the values $f_1,..,f_m,\ g_1,..,g_n$ and (an approximation to) the Gauss-Newton direction $p = (J^T J)^{-1} J^T f$.
The interface harness routine allows p to be calculated in a way that is efficient and appropriate to the features of a particular problem (see [1]).

The method implemented in `GN_solver` is basically a straightforward Gauss-Newton algorithm. The k-th iteration begins with a current solution estimate x_k and calculates the Gauss-Newton direction $p = (J_k^T J_k)^{-1} J_k^T f_k$. A convergence test is then performed which terminates the process if

$$||f_k||_2 < m\varepsilon \tag{2}$$

or

$$||g_k||_2 < \frac{n}{\sqrt{m}} \sqrt{(}\varepsilon ||f_k||_2) \tag{3}$$

or

$$||p||_2 < (\tau_a + \varepsilon)(n + ||x_k||_2) \text{ and } ||g_k||_2 < \frac{n}{m} \varepsilon^{0.3} (m + ||f_k||_2)$$
$$\text{and } |(||f_{k-1}||_2 - ||f_k||_2)| < m\tau_f. \tag{4}$$

In these tests, ε denotes machine precision while τ_a and τ_f are small positive tolerances to be specified by the user. The tests are based on suggestions made by Gill et al [2].

If convergence does not occur then a new point $x_{k+1} = x_k + sp$ is usually obtained by a line search. The case when this does not happen is when the calculated Gauss-Newton direction does not satisfy a descent property

$$-p^T g_k < \sqrt{\varepsilon} ||p||_2 ||g_k||_2.$$

In this case, an alternative direction p is computed using a two-dimensional search procedure outlined below in sections 2.3 and 2.4.

Two kinds of line search are available in `GN_solver`. The first is an Armijo-type search which uses only function values while the second combines the Armijo search with the secant method and involves calculating both the function and the gradient at every trial point. The first alternative is a *weak* search which merely ensures an acceptable decrease in the function value. The second method, however, is capable of performing a *perfect* search to find the one-dimensional minimum of F along the direction p. For a broader discussion of both types of line search see, for instance, [2] or [3].

Both line search algorithms are given below in sections 2.1 and 2.2. For the moment we note that the algorithm can specify a value s_{min} such that the search is *clearly successful* if

$$||x_{k+1} - x_k||_2 \geq s_{min}||p||_2$$

which implies that the change in the variables is not less than some chosen fraction of the Gauss-Newton step. Similarly, a line search is is *clearly unsuccessful* if the relative change in the variables is comparable with machine precision – i.e.

$$||x_{k+1} - x_k||_\infty < \varepsilon(||x_k||_\infty + \varepsilon).$$

After a clearly unsuccessful search the iterative process terminates while a clearly successful search is followed by the start of a new iteration. If, however, the line search yields a new point which lies between these alternatives, the algorithm has the option of performing a *two-dimensional search* in the plane of p and g_k in order to find a new point which is better than x_{k+1}. This search, and the conditions which trigger it, are described in sections 2.3 and 2.4 below.

Some authors [4],[5] have suggested using a quasi-Newton search direction as a fallback option when a Gauss-Newton step is unsuccessful. Our reason for not using this option in `GN_solver` relates to the harness interface. Its purpose is to allow a user to compute p *efficiently*, taking into account any structure in the Jacobian. Some nonlinear least-squares applications have thousands of parameters but very exploitable structure in the Jacobian which can make the Gauss-Newton direction quite cheap to compute. However it may be much less straightforward for any quasi-Newton update to take advantage of the problem structure and this could mean that the use of a quasi-Newton direction could be relatively inefficient.

2.1 A weak line search algorithm

The following is an Armijo algorithm for a weak line search to yield an acceptable decrease in the function F, defined by

$$\eta p^T \nabla F(x_k) \leq F(x_{k+1}) - F(x_k) \leq (1-\eta) p^T \nabla F(x_k).$$

It is used in `GN_solver` for $0 < \eta \leq 0.25$ and involves parameters $c(< 1)$, $C(> 1)$ and ε.

```
Given $x_k$, $p$, $F_k = F(x_k)$, $\bar{g}_k = p^T \nabla F(x_k)$, $s$ and $\eta$
Set $s_l = 0,\ D_l = 1,\ i_{fail} = 0$
Start extrapolation loop:
**Repeat**
 Set
    $$x^+ = x_k + sp,\ \ F^+ = F(x^+),\ \ D = \frac{(F^+ - F_k)}{s\bar{g}_k}$$
 If $\eta \leq D \leq 1 - \eta$ then $s_r = s$ and exit loop
 If $D < \eta$ then $s_r = s,\ D_r = D$ and exit loop
 Set
    $$s_l = s,\ D_l = D,\ s = Min[Cs,\ \frac{0.5s}{(1-D)}]$$
**end repeat**
Start interpolation loop:
**Repeat** If $\eta \leq D \leq 1 - \eta$ then $s_r = s$ and exit loop
 If $D < \eta$ then $s_r = s,\ D_r = D$
 If $D > 1 - \eta$ then $s_l = s,\ D_l = D$
 Set $s_{lb} = cs_r + (1-c)s_l,\ s_{ub} = cs_l + (1-c)s_r$
 Set
    $$s_q = s_r + \frac{(0.5 - D_r)}{(D_l - D_r)}(s_l - s_r),\ s = Min[s_{ub}, Max[s_{lb},\ s_q]]$$
 Set
    $$x^+ = x_k + sp,\ F^+ = F(x^+),\ D = \frac{(F^+ - F_k)}{s\bar{g}_k}$$
 If $||x^+ - x||_\infty < \varepsilon ||x_0||_\infty + \varepsilon$ set $i_{fail} = 1$ and exit loop
**end repeat**
If $i_{fail} = 0$ then $x_{k+1} = x^+,\ F_{k+1} = F^+$; otherwise search has failed.
```

2.2 A perfect line search algorithm

The following is a combined Armijo/secant algorithm for a strong or perfect line search to yield a decrease in the magnitude of the projected gradient, given by

$$|p^T \nabla F(x_{k+1})| \leq \mu |p^T \nabla F(x_k)|.$$

This is used in `GN_solver` with $\mu = 1 - 2\eta$ and $0.25 < \eta < 0.5$.

```
Given $x_0$, $p$, $F_0 = F(x_0)$, $g_0 = \nabla F(x_0)$, $s$ and $\mu$
Run the Armijo algorithm with $\eta = 0.25$ to obtain values $s_l$ and $s_r = s$
```

```
Set $x_l = x_k + s_l p,\ x_r = x_k + s_r p$
Evaluate $F_l = F(x_l),\ g_l = p^T \nabla F(x_l),\ F_r = F(x_r),\ g_r = p^T \nabla F(x_r)$
If $|g_r| \le \mu|\bar{g}_k|$ then return $s_r, x_r,\ F_r$
If $g_r \le \bar{g}_k$ then set $i_{fail} = 1$ and return $s_r,\ x_r,\ F_r$ (not a minimum)
Repeat
  Set
      $s = s_l - \frac{g_l s_l}{(g_r - g_l)},\ \ x^+ = x_k + sp,\ \ F^+ = F(x^+),\ \ g^+ = p^T \nabla F(x^+)$
  If $|g^+| \le \mu|\bar{g}_k|$ then exit loop
  If $F^+ > F_r$ then set $i_{fail} = 1$ and exit loop
  If $|g_r| < |g_l|$ then set $s_l = s_r,\ F_l = F_r,\ g_l = g_r$
  Set $s_r = s^+,\ F_r = F^+,\ g_r = g^+$
end repeat
If $i_{fail} = 0$ return $x_{k+1} = x^+,\ F_{k+1} = F^+$ as the line minimum
Otherwise return $x_{k+1} = x^+,\ F_{k+1} = F^+$ as an improved point (not a minimum).
```

2.3 The two-dimensional search algorithm

In the event that the Gauss-Newton step proves unsatisfactory for one of the reasons specified in section 2.4, `GN_solver` seeks a new point x^+ by performing a two-dimensional search in the plane spanned by the Gauss-Newton direction p and the negative gradient $-g_k$. Specifically, this search looks for the least value of F on a circular arc centred on x_k with radius ρ and can be outlined as follows:

```
Given $p$ and $g_k = \nabla F(x_k)$ (assumed not parallel) and a radius $\rho(> 0)$
Construct the unit vector $\hat{g} = g_k/||g_k||_2$
Set $\underline{u} = p - \hat{g}^T p \hat{g}$ (so that $\hat{g}^T u = 0$) and set $\hat{u} = u/||u||_2$.
Set $\bar{\theta} = \cos^{-1}[-p^T \hat{g}/||p||_2]$
Find $\theta^*$ to minimize $\phi(\theta) = F(x_k - (\rho\cos\theta)\hat{g} + (\rho\sin\theta)\hat{u})$ for $\bar{\theta} \ge \theta \ge 0$.
Return $x^+ = x_k - (\rho\cos\theta^*)\hat{g} + (\rho\sin\theta^*)\hat{u}$
```

In `GN_solver` the minimization of $\phi(\theta)$ is done by seven iterations of the bisection method. This is chosen on the expectation that $\bar{\theta}$ will be about 90^o and so seven iterations will locate θ^* to a precision of less than 1^o.

The next section shows how this 2D search is incorporated into `GN_solver`.

2.4 Triggering a two-dimensional search

There are two situations in which a two-dimensional search may be needed.

(a) If a (weak or perfect) search along p terminates with a step size $s < s_{min}$.

In this case, if the new point $x_{k+1} = x_k + sp$ and $F_{k+1} = F(x_{k+1})$ then the following procedure is used to determine whether a two-dimensional search is appropriate.

```
Evaluate $D = (F_{k+1} - F_k)/(sp^T \nabla F(x_k))$.
If $D < 1$ and $s/(1-D) < s_{min}$
  Evaluate $F_s = F(x_k + s_{min} p)$. If $F_s \geq F_k$ perform a 2D search with $\rho = s||p||_2$
  If $F^+ < F_{k+1}$ set $x_{k+1} = x^+$
Start a new iteration from $x_{k+1}$.
```

The secondary test that must be satisfied before a 2D search is attempted is one which suggests that the steplength s_{min} would produce an increase in F. This test is included because a weak line search sometimes accepts a point with $s < s_{min}$ even though the Gauss-Newton step could have yielded a much better reduction. In this situation the additional computing cost of a 2D search is not usually justified.

(b) If the search direction p gives insufficient descent.

If $-p^T g < \varepsilon||p||_2||g||_2$ then a two dimensional search is used to find a better search direction, as follows.

```
Perform a 2D search with $\rho = 0.001||p||_2$ and obtain a new point $x^+$
Perform a line search along the direction $p = x^+ - x$.
```

3 Numerical results

We quote some trial results obtained when `GN_solver` is applied to the following problems.

Problem 1 is the Rosenbrock problem

$$F(x) = 100(x_2 - x_1^2)^2 + (1 - x - 1)^2$$

using the non-standard starting point $(-7,\ 49)$.

Problem 2 is the extended scaled Rosenbrock problem

$$F(x) = \sum_{k=1}^{4} 10^4 (x_{k+1} - x_k)^2 + (1 - x_k)^2$$

starting at $x = (-0.5,\ 0.25,\ 0.0625,\ 0.003906,\ 0.0000053)$

Problem 3 is

$$F(x) = \sum_{i=1}^{30} s_i^2 \quad \text{where} \quad s_i = e^{-i/10} + 1 - x_1 e^{ix_2} - x_3 e^{ix_4}$$

using the starting point $x = (0.5,\ 0.5,\ 0.5,\ 0.0)$

Problem 4 is

$$F(x) = \sum_{i=1}^{20} s_i^2 \quad \text{where} \quad s_i = e^{-i/10} + 5 + 0.05i - x_1 e^{ix_2} - x_3 e^{ix_4}$$

using the starting point $x = (5.67,\ -0.0083,\ 0.283,\ 0.0782)$

Problem 5 is

$$F(x) = \sum_{i=1}^{41} s_i^2 \quad \text{where} \quad s_i = x_1 + x_2|i/8 - x_3|^{x_4} - (1.77 - 0.15|i/8 + 0.737|^{3.56}) + (-0.1)^i$$

using the starting point $x = (1.0,\ -1.0,\ 1.1,\ 1.1)$

For all these problems the convergence tests (2) – (4) were used with $\varepsilon = 10^{-16}$ and $\tau_a = \tau_f = 10^{-5}$.

We now give some results which show the effect of using different values of η and s_{min}. Choosing $\eta = 0.1$ implies that the line search is quite weak, with a new point being accepted if it gives quite a modest reduction in the value of F. Increasing η causes x_{k+1} to be closer to a one-dimensional minimum along the direction p; and the choice $\eta = 0.499$ causes the line search to be very close to perfect. Increasing s_{min} away from zero raises the threshold for a Gauss-Newton step to be *clearly successful* and hence tends to increase the number of two-dimensional searches that are performed.

The entries in the tables below give the numbers of iterations and function evaluations needed to solve the test problems stated above. When $s_{min} > 0$ the figure in brackets shows how many two-dimensional searches were performed. The quoted values of s_{min} vary from problem to problem, according to the smallest value of the trigger threshold which produces any change from basic the $s_{min} = 0$ case.

η	0.499	0.45	0.25	0.1
s_{min} = 0.0	69 367	68 301	80 308	81 305
s_{min} = 0.04	64(1) 377	64(1) 308	80(0) 308	81(0) 305
s_{min} = 0.05	60(2) 384	61(2) 319	71(2) 317	77(1) 309

Table 1: Iterations and function calls for `GN_solver` on Problem 1

η	0.499	0.45	0.25	0.1
s_{min} = 0.0	158 905	158 739	203 789	203 786
s_{min} = 0.03	144(2) 887	144(2) 723	198(1) 797	202(1) 805
s_{min} = 0.04	135(6) 954	136(6) 787	174(4) 790	199(2) 815

Table 2: Iterations and function calls for `GN_solver` on Problem 2

The way in which performance is affected by the choice of $\eta$is shown most clearly when $s_{min} = 0$ so that there are no two-dimensional searches. The expectation is that as η increases the number of iterations will decrease while the number of function calls per iteration will increase. Results show that this is usually the case (and remains so when $s_{min} > 0$).

η	0.499	0.45	0.25	0.1
s_{min} = 0.0	119 662	124 617	119 580	212 1219
s_{min} = 0.02	119(0) 662	114(3) 586	104(2) 444	61(2) 270
s_{min} = 0.03	101(5) 686	85(8) 572	95(5) 478	74(4) 355

Table 3: Iterations and function calls for `GN_solver` on Problem 3

η	0.499	0.45	0.25	0.1
s_{min} = 0.0	173 879	175 847	168 787	173 802
s_{min} = 0.01	132(4) 879	132(4) 717	137(5) 710	127(5) 624
s_{min} = 0.015	110(15) 989	110(15 831	117(27) 1052	112(23) 928

Table 4: Iterations and function calls for `GN_solver` on Problem 4

η	0.499	0.45	0.25	0.1
s_{min} = 0.0	116 795	108 613	121 674	113 610
s_{min} = 0.01	57(5) 439	56(5) 362	75(10) 508	69(10) 471
s_{min} = 0.02	46(8) 446	47(8) 378	62(12) 486	66(11) 477

Table 5: Iterations and function calls for `GN_solver` on Problem 5

For all five problems there is evidence that it can be beneficial to use two-dimensional searches when the basic line search takes a step less than s_{min}. In Problems 1 and 2 we see modest reductions in iteration count accompanied by small increases in numbers of function calls. In Problems 3 and 4, for smaller values of η, a few two-dimensional searches produce a substantial decrease in both iteration count and numbers of function evaluations. For Problem 5 there are appreciable improvements in iteration count and function evaluations for all values of η.

The choice of s_{min}, the threshold for triggering two-dimensional searches, seems, unfortunately, to be rather problem-dependent. Below a certain level it will, of course, have no effect; but as it becomes large enough to cause a small number of 2D searches to occur then the consequences are initially quite beneficial. However, as s_{min} increases further, some sort of law of diminishing returns seems to operate. Thus it appears that the standard line search should not be superseded too readily by the 2D alternative.

3.1 The cost of the two-dimensional search

In order to assess the efficiency of the two-dimesnional search procedure we need to look more closely at what is meant by a *function call* in the results in Tables 1 – 5. The quoted figures simply record the number of times the harness routine is called.

A two-dimensional search involves seven iterations of the bisection method which employ 17 calls to the harness routine *each requiring only the evaluation of the subfunctions* f_i. For every Gauss-Newton iteration that is saved, however, we also save a harness-routine call involving the additional and expensive calculation of a Gauss-Newton direction. Properly to evaluate the benefits of the two-dimensional searches, therefore, we need to keep distinct counts of numbers of evaluations of residuals f, gradient g and search direction p. As a specific example, consider Problem 3 with $\eta = 0.499$. When $s_{min} = 0$ there are 119 ”expensive” calls to the harness routine to calculate f, g and the Gauss-Newton direction p. In addition there are 543 ”medium-cost” calls to the harness routine to supply only f and g. In contrast, when $s_{min} = 0.03$, the five two-dimensional searches take 85 of the cheapest harness evaluations yielding values of f only. The consequent reductions in numbers of high- and medium-cost harness calls are, respectively, 18 and 43.

In order to quantify savings in arithmetic effort when $m >> n$ we can make the following estimates. We can sssume that a function and gradient call costs roughly n times a function-only call and that a function, gradient and search direction call costs roughly n^2 times a function-only call. Therefore the two-dimensional search is beneficial if

$$\rho_{2D} = \frac{\text{extra } C_F}{n^2(\text{decrease in } C_{Fgp}) + n(\text{decrease in } C_{Fg})} < 1$$

For problem 3, with $n = 4$, $\rho_{2D} = 85/(16 \times 18 + 4 \times 43) \approx 0.185$ indicating that there is an overall saving of effort. The following tables show ρ_{2D} for each test problem and they the 2D search is beneficial in about 90% of the tests. The best values of ρ_{2D} are not necessarily in the places where we intuitively put them based on counts of iterations and function calls.

η	0.499	0.45	0.25	0.1
s_{min} = 0.04	0.7	0.61	1	1
s_{min} = 0.05	0.65	0.74	0.5	0.5

Table 6: Performance gain ρ_{2D} for Problem 1

η	0.499	0.45	0.25	0.1
s_{min} = 0.03	0.06	0.06	0.012	1.7
s_{min} = 0.04	0.14	0.14	0.07	0.32

Table 7: Performance gain ρ_{2D} for Problem 2

η	0.499	0.45	0.25	0.1
$s_{min} = 0.02$	1	0.11	0.04	0.006
$s_{min} = 0.03$	0.18	0.11	0.08	0.013

Table 8: Performance gain ρ_{2D} for Problem 3

η	0.499	0.45	0.25	0.1
$s_{min} = 0.01$	0.09	0.05	0.08	0.05
$s_{min} = 0.015$	0.19	0.14	0.33	0.22

Table 9: Performance gain ρ_{2D} for Problem 4

η	0.499	0.45	0.25	0.1
$s_{min} = 0.01$	0.03	0.04	0.09	0.1
$s_{min} = 0.02$	0.05	0.06	0.09	0.1

Table 10: Performance gain ρ_{2D} for Problem 5

3.2 Invariance to scaling

An important issue with regard to general purpose software like `GN_solver` is that behaviour should be relatively insensitive to scaling of the function or variables (which might depend on a user's choice of units, for example). The tests (2) – (4) are intended to be fairly robust and to ensure that, so long as a user makes reasonable choices of ε, τ_a and τ_f, the algorithm will neither terminate a long way from a solution nor waste many iterations making trivial improvements in the vicinity of a solution. The next two tables show that the 2D search is fairly insensitive to scaling.

Table 11 shows that the behaviour of `GN_solver` does not change significantly when the function F is scaled by a constant σ. Comparison of these results with those in the previous section shows that the changes in numbers of iterations and function evaluations is relatively insensitive to such scaling on the objective function. Table 12 gives a similar set of results showing performance of `GN_solver` when the variables in the problem are scaled by a factor σ. Once again the variations in numbers of function calls and iterations are very small.

The results in this section suggest that the convergence tests (2) – (4) are quite robust. It is also noteworthy that the fall-back two-dimensional search strategy does not appear to be affected by scaling on the function or the variables, since the number of special iterations is the same for the scaled and unscaled versions of the problems in our experiments.

Problem 1			
σ	$\eta = 0.499,\ s_{min} = 0.0$	$\eta = 0.25,\ s_{min} = 0.04$	$\eta = 0.1,\ s_{min} = 0.05$
1000	69(0) 367	80(0) 308	77(1) 309
0.001	69(0) 367	79(0) 306	76(1) 307
Problem 2			
σ	$\eta = 0.499,\ s_{min} = 0.0$	$\eta = 0.25,\ s_{min} = 0.03$	$\eta = 0.1,\ s_{min} = 0.04$
1000	159(0) 907	198(1) 797	199(2) 815
0.001	158(0) 905	198(1) 797	199(2) 815
Problem 3			
σ	$\eta = 0.499,\ s_{min} = 0.0$	$\eta = 0.25,\ s_{min} = 0.02$	$\eta = 0.1,\ s_{min} = 0.03$
1000	120(0) 664	105(2) 446	74(4) 355
0.001	118(0) 660	103(2) 442	73(4) 353
Problem 4			
σ	$\eta = 0.499,\ s_{min} = 0.0$	$\eta = 0.25,\ s_{min} = 0.01$	$\eta = 0.1,\ s_{min} = 0.015$
1000	173(0) 879	137(5) 710	112(23) 928
0.001	172(0) 877	136(5) 708	111(23) 926
Problem 5			
σ	$\eta = 0.499,\ s_{min} = 0.0$	$\eta = 0.25,\ s_{min} = 0.01$	$\eta = 0.1,\ s_{min} = 0.02$
1000	116(0) 795	75(10) 511	66(11) 480
0.001	115(0) 793	75(10) 508	66(11) 477

Table 11: Performance of `GN_solver` when function is scaled

4 Discussion and further work

We have described a robust version of the Gaiss-Newton method which includes a two-dimensional search to prevent slow convergence or premature termination. This has been implemented as the algorithm `GN_solver` and results quoted in the previous section indicate that it fulfills its purpose of being a reliable general-purpose code for solving non-linear least-squares problems. We have paid particular attention to testing the robustness of its convergence tests and the effectiveness of the fall-back two-dimensional search to be used if the Gauss-Newton direction proves unsatisfactory.

The current version of the program allows a user to specify parameters η and s_{min} which control the accuracy of the line search and the threshold steplength which triggers the two-dimensional search. The test results do not suggest hard and fast guidelines for choosing parameter values to give the ”best” performance on any particular problem. However it seems safe to use $0.5 > \eta > 0.1$ and $0.02 > s_{min} \geq 0$. Fine-tuning of the parameters is probably worthwhile only for users who routinely solve data-fitting problems involving one type of highly nonlinear model.

We conclude by mentioning some further possible refinements to `GN_solver`.

Problem 1			
σ	$\eta = 0.499,\ s_{min} = 0.0$	$\eta = 0.25,\ s_{min} = 0.04$	$\eta = 0.1,\ s_{min} = 0.05$
1000	69(0) 367	79(0) 306	76(1) 307
0.001	69(0) 367	80(0) 308	76(1) 307
Problem 2			
σ	$\eta = 0.499,\ s_{min} = 0.0$	$\eta = 0.25,\ s_{min} = 0.03$	$\eta = 0.1,\ s_{min} = 0.04$
1000	158(0) 905	198(1) 797	199(2) 815
0.001	158(0) 905	198(1) 797	199(2) 815
Problem 3			
σ	$\eta = 0.499,\ s_{min} = 0.0$	$\eta = 0.25,\ s_{min} = 0.02$	$\eta = 0.1,\ s_{min} = 0.03$
1000	119(0) 662	104(2) 444	74(4) 355
0.001	120(0) 664	105(2) 446	74(4) 355
Problem 4			
σ	$\eta = 0.499,\ s_{min} = 0.0$	$\eta = 0.25,\ s_{min} = 0.01$	$\eta = 0.1,\ s_{min} = 0.015$
1000	172(0) 877	137(5) 710	112(23) 928
0.001	173(0) 879	137(5) 710	111(23) 926
Problem 5			
σ	$\eta = 0.499,\ s_{min} = 0.0$	$\eta = 0.25,\ s_{min} = 0.01$	$\eta = 0.1,\ s_{min} = 0.02$
1000	115(0) 793	75(10) 508	66(11) 477
0.001	116(0) 795	75(10) 510	66(11) 480

Table 12: Performance of `GN_solver` when variables are scaled

4.1 Replacing the bisection method in the 2D search

Although the bisection method has worked quite well as a method of finding a new point in the p, g-plane, the algorithm in section 2.3 may not be the most efficient approach that could be taken. It could prove more economical in terms of function evaluations to combine bisection with quadratic searching. As soon as bisection locates a range in which the least function value is at the midpoint we know that a quadratic fitted polynomial will have a minimum within the range and we can expect to estimate the true minimum more quickly and accurately by repeated quadratic interpolation than by continuing with bisection.

4.2 Combining extrapolation with the 2D search

Suppose two-dimensional search in case *(a)* of section 2.4 yields a point $x^{=}$ such that $F(x^{+}) < F(x^{(k+1)})$, where $x^{(k+1)}$ is the point reached by the standard line search along p. Then, instead of simply setting $x^{(k+1)} = x^{+}$ it might be beneficial to extrapolate by means of a line search along the new direction $\tilde{p} = x^{+} - x^{(k)}$ (as is done in case *(b)* of section 2.4). A suitable trigger for doing such an extrapolation could be based on evaluating

$$D = \frac{F(x^+) - F(x^{(k)})}{g^{(k)T}(x^+ - x^{(k)})}.$$

If $D > 0.75$ (say) then a larger step to (at least) $x^{(k)} + 2\tilde{p}$ would probably produce a significant further decrease on F.

4.3 Using the latest gradient in the 2D search

In case *(a)* of section 2.4 the two-dimensional search is in the plane defined by p and $-g^{(k)}$, the steepest descent direction at the start of the current iteration. However, a reason for a line search along p to terminate with a small step $s < s_{min}$ may be that the gradient vector is changing rapidly. If this is so, then it may be more effective to perform a two-dimensional search in the plane defined by p and some combination of $g^{(k)}$ and $g^{(k+1)}$ where $-g^{(k+1)}$, the steepest descent direction at the stopping point of the line search.

References

[1] Cox, M.G., A.B. Forbes, P.M. Fossati, P.M. Harris and I.M. Smith, Techniques for the efficient solution of large-scale calibration problems, Technical Report CMSC 25/03, National Physical Laboratory, Teddington, U.K., 2003.

[2] Gill P.E., Murray, W., and Wright, M.H., Practical Optimization, Academic Press, London, New York, 1981

[3] Bartholomew-Biggs M.C., Nonlinear Optimization with Engineering Applications, Springer 2008

[4] Bartholomew-Biggs M.C. , The estimation of the Hessian matrix in nonlinear least squares problems with nonzero residuals, Math. Prog., 12:67–80, 1977.

[5] Al-Baali M. and Fletcher R., Variational methods for non-linear least squares. J. Oper. Res. Soc., 36:405–421, 1985.