IMaCH is proposing to measure Health Expectancy with one index

REVES 2015. Singapore

Nicolas Brouard   Christopher Heathcote

National Institute for Demographic Studies (INED) and Austalian National University (ANU)

REVES Singapore, 3 June 2015

IMaCh, for Interpolated Markov Chain, is a software which provides period Health Expectancy (HE) estimated from a cross-longitudinal survey. Information required is only the age and status (Healthy, Unhealthy or Death) at each interview. Such data are now available in a few countries including Asian countries. HEs can be estimated by socio-economic status including covariates or different health statuses. The main current difficulty inherent to HE computed from transitions between health statuses resides in the computation itself. Because of different time exposures between successive interviews, the program maximises the likelihood of the sample based on a multinomial logistic model of a monthly transition probability (Lièvre et al., 2003).

Convergence itself and time to convergence are crucial. In recent versions of IMaCh various algorithms as well as various implementations of these algorithms are explored. The quickest method used so far was an algorithm of M.J.D. Powell published in 1964 (Powell, 1964) and implemented via the Numerical Recipes in C library (Press et al., 1992).

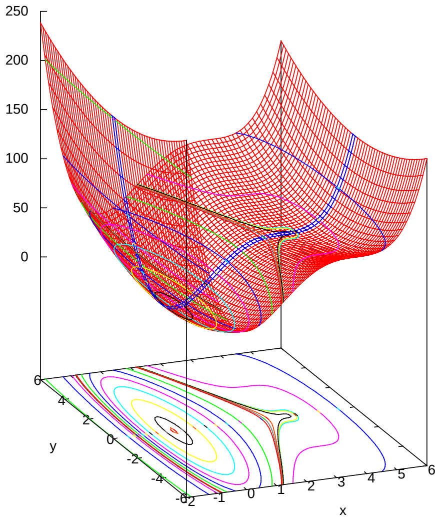
But compiled versions using the Intel C compiler in order to run efficiently on recent Windows 64-bit, gave more accurate results than the traditional GNU C compiler used so far. Investigations showed that NRC test for stopping iterations was inadequate as well as Powell's criteria (which seems to have never be clearly understood) is over defined. These two errors might explain why, sometimes, convergence could not be reached with a monthly transition but did converge with an annual transition model. About 40 years later, M.J.D. Powell published a series of new algorithms named NEWUOA (Powell, 2006) and BOBYQUA (Powell, 2002) with their corresponding "public domain" Fortran code.

These new algorithms are claimed (Rios and Sahinidis, 2013) to be more adapted to maximization of function with hundred of variables which is the case when covariates and health statuses are numerous. The comparison doesn't include Powell's determinant/conjugate gradient methods because no software, except NRC, is available. The GNU software library (GSL) doesn't implement this Powell algorithm but a few others which are unfortunately 5 times slower.

If there is only one scientific publication on the statistical method and on the period Health Expectancy index produced by the program, there was no presentation of the algorithm which are used by this program.
Probably more than 30 scientific publications produced results from this program. Fiona Matthews, Vikki O'Neill, Carol Jagger and Pia Wohland presented at the last REVES meeting in Edinburgh a *Comparison of methods and programs for calculating health life expectancies*, and IMaCh competed well.
In the early 90's when we designed with Agnès Lièvre and Christopher Heathcote (ANU) the program we wanted to avoid to reinvent the wheel and to trust specialists in Optimization computing, using a very efficient algorithm and a well known, ready to use, package; today we have to reconsider both decisions.

- IMaCH software: one single publication focusing on the statistical and demographical method to compute on single index from a cross-longitudinal survey (Lièvre et al., 2003);
- but no publication on the algorithms used; optimization was not our job…
- but the job of famous mathematicians specialized in algorithms like MJD Powell (Powell, 1964);
- also, do not reinvent the wheel: use a famous scientific library (Press et al., 1992);
- But could we trust them? The hidden face and burden of computer softwares.

Maximum Likelihood Estimators consist in finding the maximum of multivariable function. Here is an example of minimizing a function of two parameters, like mean and standard deviation. Importance of contour maps in the theory of optimization.

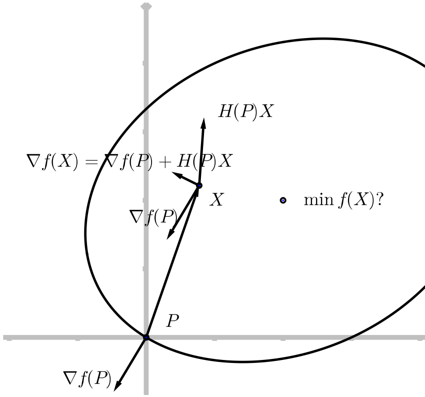


We can deduce the gradient at point X as the sum of the two vectors $\nabla_f(P)$, the *gradient* of f at P and the *Hessian matrix* HX.

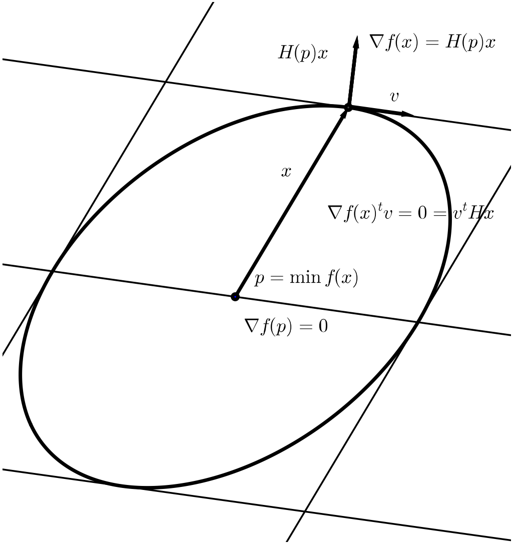
$$f(X) = f(P) + \nabla f(x)^T X + \frac{1}{2} X^t H X + \cdots$$

$$\nabla f(X) = \nabla f(P) + H(P)X$$

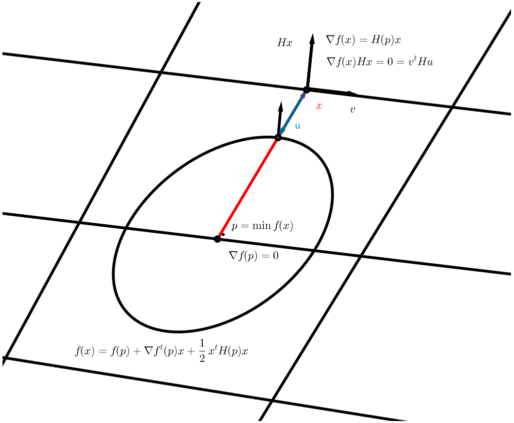
Minimum of the function is searched.



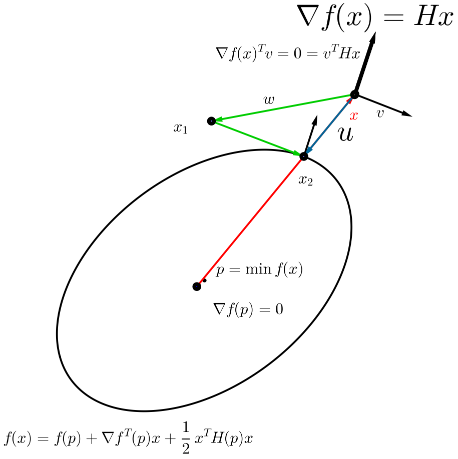
If p is the extremum then $\nabla f(p) = 0$ and at any point x, $\nabla f(x) = H(p)x$



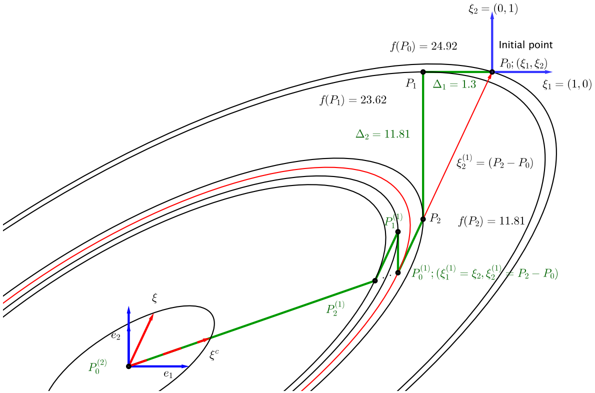
From minimum in direction v, how to find a direction u such that any point of u is still a minimum in direction v: v and u are conjugate.


Hx
$\nabla f(x) = H(p)x$
$\nabla f(x)Hx = 0 = v^t Hu$
x
v
u
$p = \min f(x)$
$\nabla f(p) = 0$
$f(x) = f(p) + \nabla f^t(p)x + \frac{1}{2}x^t H(p)x$

Finding the conjugate *u* of *v*: from minimum *v* take random directions *w* up to (random) x_1, minimize in direction *v* and find x_2; *u* is direction (x_2 - x).


$\nabla f(x) = Hx$
$\nabla f(x)^T v = 0 = v^T Hx$
w
x_1
x
v
u
x_2
$p = \min f(x)$
$\nabla f(p) = 0$
$f(x) = f(p) + \nabla f^T(p)x + \frac{1}{2}\,x^T H(p)x$

From an initial random point P_0 and two random or canonical directions ξ_1 and ξ_2, find P_1, P_2 and $P_0^{(1)}$ at iteration 1 and the extremum $P_0^{(2)}$ at second and last iteration.



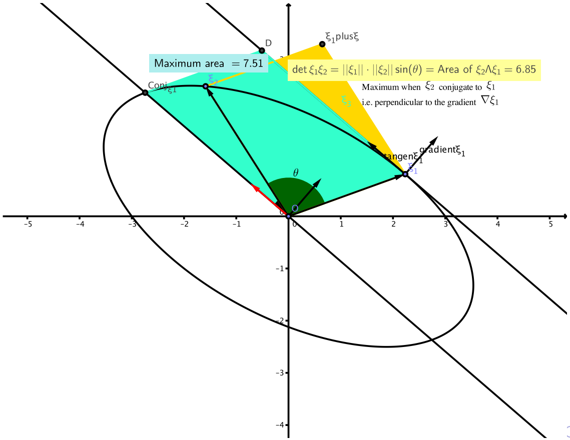
```
Data: Conjugate gradients
Result: Minimum of a function $f$
take $n$ directions $\xi_r$ and a point $P_0$;
while For each $r = 1, n$ do
    calculate $\lambda_r$ which minimizes $\min_{\lambda_r} f(P_{r-1} + \lambda_r \xi_r)$;
    and set point $P_r = P_{r-1} + \lambda_r \xi_r$ ;
end
while For each $r = 1, n-1$ do
    set $\xi_{r-1} = \xi_r$
end
Replace $\xi_n = P_n - P_0$ ;
Minimize $P_n + \lambda(P_n - P_0)$;
Set new $P_0 = P_n + \lambda_{\min} \xi_n$ (error in Powell: was $P_0 = P_0 + \lambda_{\min} \xi_n$) Go
back to the beginning of current section with a new $P_O$ and new set of
rotated $\xi_i$: $\xi_1$ is dropped and new $\xi_n$ is old $P_n - P_0$;
```

Algorithm 1: Powell conjugated gradients algorithm

But this simple algorithm doesn't behave satisfactorily when dimension are bigger than 5 Powell (1964) because the new direction might tend to be colinear to another direction and the maximum be in a space of lower dimensions and not an extremum.
And thus sometimes it can be better NOT to use direction $P_n - P_0$ but to keep the set of directions of previous iteration.

Powell's theorem on maximal determinant of A-normed directions

The determinant of two A-normed $(A = \frac{1}{2}H)$ directions reaches a maximum when both directions are conjugated


D
ξ1plusξ
Maximum area = 7.51
$\det \xi_1\xi_2 = ||\xi_1|| \cdot ||\xi_2|| \sin(\theta) =$ Area of $\xi_2 \Lambda \xi_1 = 6.85$
Maximum when ξ_2 conjugate to ξ_1
i.e. perpendicular to the gradient $\nabla\xi_1$
Conjξ1
ξ2
ξ3
tangenξ1
gradientξ1
ξ1
θ
O

Powells method with evaluation of the determinant

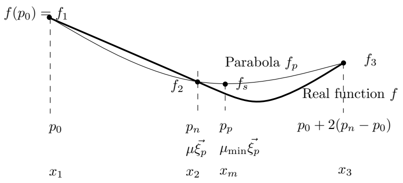
The brilliant idea of Powell consists in exploiting this property and proving that the inclusion of the direction $P_0 - P_n$ should be accepted only if the new direction improves the 'conjugation' of the n directions, that is correspond to an increase of the determinant of the new set compared to the determinant of the former set. The new direction $p_0 - p_n$ is a linear combination of each direction $(p_0 - p_1)$, $(p_1 - p_2)$ which also corresponds to a decrease of the function $\Delta_1 = f(p_1) - f((p_0)$, $\Delta_2 = f(p_2) - f((p_1)$. It can be seen that if we use A-normed vectors for each direction, each decrease is exactly the square of the coordinate α_i:

$$f(p_2) - f(p_1) = \Delta_i = (\alpha_1 \xi_1)^T A(\alpha_1 \xi_1) = \alpha_1^2 ||\xi_1||_A$$

$$p_n - p_0 = \alpha_1 \xi_1 + \alpha_2 \xi_2 + \cdots + \alpha_i \xi_i + \cdots + \alpha_n \xi_n = \mu \xi$$

$$\det(\xi_1, \xi_2, \ldots, \xi_{i-1}, \xi_p, \xi_i, \ldots, \xi_n) = \frac{\alpha_i}{\mu} \det(\xi_1, \xi_2, \ldots, \xi_i, \ldots, \xi_n)$$

$$\max_{i=1,n} \alpha_i \geq \mu \,.$$



Also the estimation of the minimum of the function in the direction $p_0 - p_n$ is costly and might be not necessary but only the curvature of the function in order to get the A-normed value of its direction (named ξ_p). In fact only its length μ is needed: $||p_0 - p_n|| = \mu ||\xi_p||_A$.
In order to estimate the curvature, the function is estimated at point $f(p_0 + 2(p_n - p_0))$ because the parabola is unique and the second derivative can be obtained. In summary the criteria FOR inclusion is simply:

$$\max_{i=1,n} \{f(p_{i-1}) - f(p_i)\} = \Delta > \mu = \frac{f(p_0) - 2f(p_n) + f(p_0 + 2(p_n - p_0))}{2} \quad (1)$$

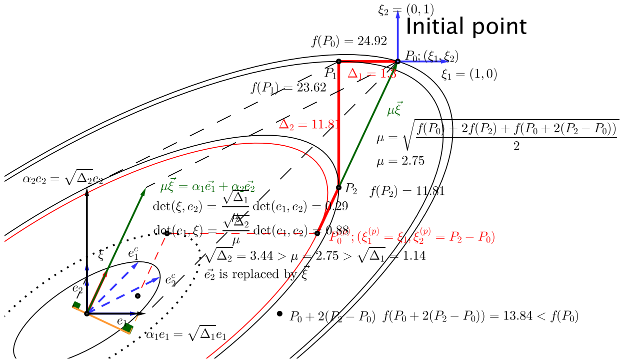
In the example shown, if ξ is replacing e_1 at next iteration, ($||e_1||_A = 1$), the numerical value of the determinant of the new set of directions:
$\det(\xi, e_2) = \frac{\sqrt{\Delta_1}}{\mu} \det(e_1, e_2) = 0.29$
is lower than the determinant obtained if ξ is replacing e_2:
$\det(e_1, \xi) = \frac{\sqrt{\Delta_2}}{\mu} \det(e_1, e_2) = 0.88$.
Also, $\sqrt{\Delta_2} = 3.44$ is bigger than $\mu = 2.75$ meaning that there is a real gain in the replacement of e_2 for next iteration.

As the real function will probably not behave as a parabola between P_0 and $P_0 + 2(P_2 - P_0)$, the minimum of the real function must be found along that direction until new point $P_0^{(p)}$ ((p) for Powell) is found.
A new iteration is started with $P_0^{(p)}$ and the directions $(\xi_1^{(p)} = \xi_1, \xi_2^{(p)} = P_2 - P_0)$. The red dotted line explained these new points $P_1^{(p)}$ and $P_2^{(p)}$.
Direction $P_0^{(p)}, P_2^{(p)}$ conducts to extremum.

This example shows how at P_2, the minimum in direction $P_2 - P_0$, is not estimated (costly) but calculated if the function was behaving like a quadratic function (parabola). From $P_0^{(1)}$, direction ξ_2 is dropped because (1) it corresponds to the maximum gap Δ_2 (2) the $A-$*normed* distance between P_2 and P_1 is bigger than the $A-$*normed* distance between P_0 and P_2:

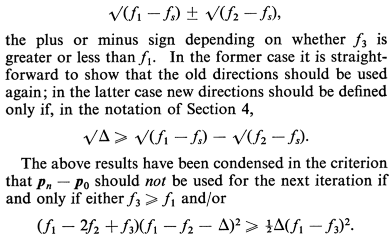
$$||P_2 - P_1||_A > ||P_0 - P_2||_A$$

$$\text{or } \sqrt{\Delta_2} = \alpha_2 > \mu\,.$$


$\xi_2 = (0,1)$
Initial point
$f(P_0) = 24.92$
$P_0; (\xi_1, \xi_2)$
$\xi_1 = (1,0)$
P_1
$\Delta_1 = 1.3$
$f(P_1) = 23.62$
$\mu\vec{\xi}$
$\Delta_2 = 11.81$
$\mu = \sqrt{\frac{f(P_0) - 2f(P_2) + f(P_0 + 2(P_2 - P_0))}{2}}$
$\mu = 2.75$
$\alpha_2 e_2 = \sqrt{\Delta_2} e_2$
$\mu\vec{\xi} = \alpha_1\vec{e_1} + \alpha_2\vec{e_2}$
P_2
$f(P_2) = 11.81$
$\det(\xi, e_2) = \frac{\sqrt{\Delta_1}}{\mu}\det(e_1, e_2) = 0.29$
$\det(e_1, \xi) = \frac{\sqrt{\Delta_2}}{\mu}\det(e_1, e_2) = 0.88$
$P_0^{(1)}; (\xi_1^{(p)} = \xi_1, \xi_2^{(p)} = P_2 - P_0)$
ξ
e_1^c
$\sqrt{\Delta_2} = 3.44 > \mu = 2.75 > \sqrt{\Delta_1} = 1.14$
e_2^c
$\vec{e_2}$ is replaced by $\vec{\xi}$
e_2
$P_0 + 2(P_2 - P_0)$ $f(P_0 + 2(P_2 - P_0)) = 13.84 < f(P_0)$
e_1
$\alpha_1 e_1 = \sqrt{\Delta_1} e_1$

Powell's strange conditions

Strangely, M. D. Powell doesn't use his own criteria but a more complex inequality, that doesn't seem to be understood in the literature probably because it is not fully understandable. Using Powell's simplified notations, here is an excerpt from his article for NOT inclusion.
[...] μ is predicted as



$$\sqrt{(f_1 - f_s)} \pm \sqrt{(f_2 - f_s)},$$

the plus or minus sign depending on whether f_3 is greater or less than f_1. In the former case it is straightforward to show that the old directions should be used again; in the latter case new directions should be defined only if, in the notation of Section 4,

$$\sqrt{\Delta} \geqslant \sqrt{(f_1 - f_s)} - \sqrt{(f_2 - f_s)}.$$

The above results have been condensed in the criterion that $\boldsymbol{p}_n - \boldsymbol{p}_0$ should *not* be used for the next iteration if and only if either $f_3 \geqslant f_1$ and/or

$$(f_1 - 2f_2 + f_3)(f_1 - f_2 - \Delta)^2 \geqslant \tfrac{1}{2}\Delta(f_1 - f_3)^2.$$

We can remark that the condition for a new direction to be included is one of the 3 equivalent inequalities:

$$\mu^2 < \Delta \iff 0 < f_1 - 2f_2 + f_3 < 2\Delta \iff f_1 - f_2 - \Delta < \frac{f_1 - f_3}{2}$$

If $f_3 \leq f_1$ and $f_1 - f_2 \geq \Delta$ the right most inequality can be squared (adding a parasite solution)

$$0 \leq (f_1 - f_2 - \Delta)^2 \leq \frac{(f_1 - f_3)^2}{4}.$$

Multiplying it by former inequality:

$$0 < f_1 - 2f_2 + f_3 < 2\Delta$$

we get, Powell's inequality FOR inclusion of the new direction:

$$(f_1 - 2f_2 + f_3)(f_1 - f_2 - \Delta)^2 < \frac{(f_1 - f_3)^2}{2}\Delta$$

the equality being obtained for $\Delta = \mu^2$ but a parasite solution has been added:

$$\Delta = \lambda^2 = \frac{2(f_1 - f_2)^2}{f_1 - 2f_2 + f_3} \qquad (2)$$

and Powell's condition FOR inclusion is unfortunately equivalent to:

$$\mu^2 < \Delta < \lambda^2$$

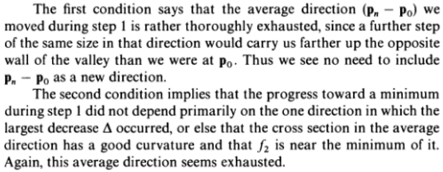
$$\frac{f_1 - 2f_2 + f_3}{2} = \mu^2 < \Delta < \lambda^2 = \frac{2(f_1 - f_2)^2}{f_1 - 2f_2 + f_3}$$

Powell's strange conditions

On the right is a case where the new direction will not be accepted according to Powell's inequality (red) even if the new set of direction was more conjugate.

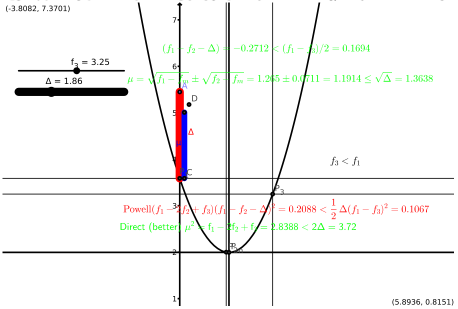
In reality, this parasite solution is not altering substantially the speed of convergence but it seemed important to discuss that point.

Remark It can be seen from Acton (1970) famous book entitled "Numerical methods that work" quoted below (p. 465) that his comment doesn't refer neither to the necessary increase of the determinant in order to get conjugate directions, nor to the useless second limit λ.



The first condition says that the average direction $(\mathbf{p}_n - \mathbf{p}_0)$ we moved during step 1 is rather thoroughly exhausted, since a further step of the same size in that direction would carry us farther up the opposite wall of the valley than we were at $\mathbf{p}_0$. Thus we see no need to include $\mathbf{p}_n - \mathbf{p}_0$ as a new direction.

The second condition implies that the progress toward a minimum during step 1 did not depend primarily on the one direction in which the largest decrease Δ occurred, or else that the cross section in the average direction has a good curvature and that f_2 is near the minimum of it. Again, this average direction seems exhausted.



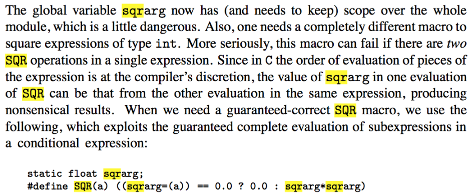
A more important error concerns the bracketing function published in NRC (Press et al., 1992) and the error was found while running IMaCh on Windows 64 bit PC because current free Gnu C Compiler are not optimized for 64 bit processor and 3.5 slower (see table below)!

Time	OS (64bit)	Compiler	Processor
18' 24"	Windows 8.1	gcc 4.8.0	E3-1225V3
5' 27"	Windows 8.1	Intel C ilc	E3-1225V3
5' 18"	OS/X	Apple 64 bit gcc 4.2.1	Core I7

Thus we tried to compile with Intel C/C++ compiler but it did not compile on the following line of Numerical Recipes in C because of an "SQR macro":

```
t=2.0*(fp-2.0*(*fret)+fptt)*SQR(fp-(*fret)-del)-del*SQR(fp-fptt);
```

Interestingly here is how NRC (Press *et al.*) explained why they used this complex macro SQR (instead of using a simple `x*x`) which was supposed to run on any C compiler:


The global variable `sqrarg` now has (and needs to keep) scope over the whole module, which is a little dangerous. Also, one needs a completely different macro to square expressions of type `int`. More seriously, this macro can fail if there are *two* `SQR` operations in a single expression. Since in C the order of evaluation of pieces of the expression is at the compiler's discretion, the value of `sqrarg` in one evaluation of `SQR` can be that from the other evaluation in the same expression, producing nonsensical results. When we need a guaranteed-correct `SQR` macro, we use the following, which exploits the guaranteed complete evaluation of subexpressions in a conditional expression:

```
static float sqrarg;
#define SQR(a) ((sqrarg=(a)) == 0.0 ? 0.0 : sqrarg*sqrarg)
```

In fact, as pointed out by Intel in a discussion after my own report on this the potential bug, the macro does not respect the *sequence point* between the evaluation of SQR and a standard C compiler is allowed to evaluate the value of the last SQR and to use its value in the first. Despite the fact that this syntax error was already reported by someone else in 2011 on the NR forum as a warn that another compiler than gcc could give a wrong result, nothing have been and the current version (3rd edition) is giving wrong results if compiled by Intel C/C++ compiler.

Thus the corrupted line should be split into two steps:

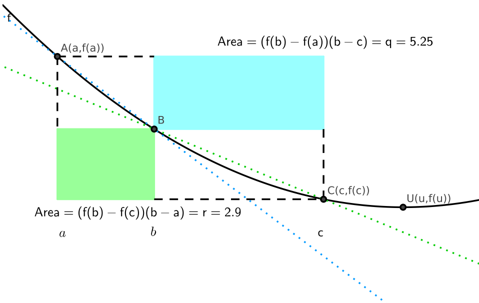
```
t=2.0*(fp-2.0*(*fret)+fptt)*SQR(fp-(*fret)-del)
t=t-del*SQR(fp-fptt);
```

Fortunately, after reporting the error of NRC, Intel sponsored the IMaCh project by providing free licenses for the 3 operating systems Windows (64 and 32 bit), OS/X and Linux which run IMaCh.

Starting from two points, A and B such that $(a < b, f(a) > f(b)$, the idea of the algorithm consists in finding, in the downhill direction (right direction here) a point c, which could bracket the *minimum* of the *real* function, that is such that $f(c) > f(u) > f(b)$. A and B can be replaced by other points during the process, the output should be a bracket.
We choose c at 1 plus golden distance from b, $(1+0.61803)(b-a)$ and compute the costly $f(c)$. Depending on its position we have different cases:

$f(c) > f(b)$	We found a bracket and the minimum could be between $[a,b]$ or $[b,c]$ but not outside $[a,c]$.
$f(c) < f(b)$	The minimum can be between (b,c) or beyond c but not between $[a,b]$ because otherwise the function will not have a unique minimum. Thus we can start again the process with points B and C and estimate f at D. $f(d) = f(c+(1+g)^2)$. If the minimum was between $[b,c]$, $f(d)$ will be greater than $f(c)$ otherwise the function will not have a unique minimum. **Thus if the minimum was between $[b,c]$ the bracket will be obtained at the next step** but if the minimum was beyond c, we will start again but enlarging of a factor $1+g$ our interval of search to the right.

Parabolic function in 3 points A,B,C and determination of its minimum at U: $u = b - \frac{(b-c)q-(b-a)r}{2(q-r)}$, involving two slopes and areas q and r as shown on the right. When area q is bigger than area r, the minimum is on the right.
The text in **bold** was omitted in `mnbrak` algorithm of NRC, causing wrong maximization of the Likelihood function and wrong Health expectancy index.


f
A(a,f(a))
Area = (f(b) − f(a))(b − c) = q = 5.25
B
C(c,f(c))
U(u,f(u))
Area = (f(b) − f(c))(b − a) = r = 2.9
a
b
c

Here is the modified code of NRC.

```
while (*fb > *fc) { /* Declining a,b,c with fa> fb > fc */
  r=(*bx-*ax)*(*fb-*fc);
  q=(*bx-*cx)*(*fb-*fa);
  u=(*bx)-((*bx-*cx)*q-(*bx-*ax)*r)/
    (2.0*SIGN(FMAX(fabs(q-r),TINY),q-r));
  /* Minimum abscissa of a parabolic estimated from (a,fa), (b,fb) and (c,fc). */
  ulim=(*bx)+GLIMIT*(*cx-*bx);
  /* Maximum abscissa where function should be evaluated */
  if ((*bx-u)*(u-*cx) > 0.0) { /* if u_p is between b and c */
    fu=(*func)(u);
#ifdef MNBRAKORIGINAL
#else
      dum=u; /* Shifting c and u */
      u = *cx;
      *cx = dum;
      dum = fu;
      fu = *fc;
      *fc =dum;
#endif
  } else if ((*cx-u)*(u-ulim) > 0.0) {
    /* u is after c but before ulim */
```

Using the sample data that is distributed with the IMaCh program since 15 years, the error in the Maximum Likelihood (more precisely 2*Log Likelihood) is equal to 5.11; and this value is comparable to the addition of a significant covariate (like sex), because the 2LL ratio behaves as a χ^2 with one degree of liberty and its 95% value is 3.84!
The NRC library error seems to have been fixed (in a slightly different manner) in a later edition but the information did not reach our research institute and no errata book is published unfortunately.

#iterations	-2*LL	Value of estimated parameters							
		1	2	3	4	5	6	7	8
34	46542.387	-12.2458	0.0923	-10.6725	0.0609	-2.6445	-0.0223	-4.7740	0.0078
62	46537.279	-12.3115	0.0930	-10.1982	0.0554	-1.6774	-0.0338	-5.6564	0.0179
	5.11	=Difference in log likelihood ratio							

In conclusion concerning the IMaCh program,

- the most important error is the NRC error in the algorithm of the mnbrak routine because it clearly gives wrong results.
- Powell's criteria for choosing the set of conjugate directions
  - was not really described even in the specialized literature:
  - authors who commented the inequality did not discuss it;
  - authors who tried to interpret its meaning are, like Acton Acton (1970) very wrong.
  - The inequality (1) presented here to change the set of directions is simpler and improves the quality of this method which is the most efficient method from the family of "linear searches".
  - More time is needed to better understand Powell's new algorithms which belong to the family of "trust regions".
  - Current test using NEWUOA with 8 paramaters does not change the speed of convergence. But newer IMaCh will be able to incorporate many more covariates and the move to "trust regions" algorithms will probably be justified.

References I

Acton, F. S. (1970). *Numerical Methods that Work*. Harper & Row.

Lièvre, A., Brouard, N., and Heathcote, C. (2003). The Estimation of Health Expectancies from Cross-longitudinal studies. *Mathematical Population Studies*, 10(4):221–248.

Powell, M. (2002). UOBYQA: unconstrained optimization by quadratic approximation. *Mathematical Programming*, 92(3):555–582.

Powell, M. (2006). The NEWUOA software for unconstrained optimization without derivatives. In Di Pillo, G. and Roma, M., editors, *Large-Scale Nonlinear Optimization*, volume 83 of *Nonconvex Optimization and Its Applications*, pages 255–297. Springer US.

Powell, M. J. D. (1964). An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162.

References II

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical Recipes in C (2nd Ed.): The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA.

Rios, L. and Sahinidis, N. (2013). Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293.

Thank you for your attention!