A GEOMETRIC APPROACH TO FLETCHER'S IDEAL PENALTY FUNCTION

Bruce Christianson

School of Information Sciences, University of Hertfordshire
Hatfield, Herts AL10 9AB, England, Europe

Numerical Optimisation Centre Technical Report 280, November 1993
to appear Journal of Optimization Theory and Applications (1995)

Abstract

In this note we derive a geometric formulation, for equality constrained problems, of an ideal penalty function. This differentiable penalty function requires no parameter estimation or adjustment, has numerical conditioning similar to that of the target function from which it is constructed, and also has the desirable property that the strict second order constrained minima of the target function are precisely those strict second order unconstrained minima of the penalty function which satisfy the constraints. Such penalty functions can be used to establish termination properties for algorithms which avoid ill-conditioned steps. Numerical values for the penalty function and its derivatives can be efficiently calculated using automatic differentiation techniques.

Key Words. Automatic differentiation, constrained optimization, differentiable penalty function, reverse accumulation, termination proofs, validiation.

1. Introduction and Notation. Consider the following problem:

$$\text{Minimize } y = f(u) \text{ subject to } k(u) = 0 \text{ where } dim(k) < dim(u)$$

It is well known that, under fairly mild conditions which are usually assumed in practice, this equality constrained problem is locally equivalent to an unconstrained problem applied to a penalty function. In this note, we give a geometric derivation of a parameter free penalty function of a similar type to the parameterized functions considered by Fletcher (Refs. 1,2,3).

We assume that f and k have continuous second derivatives in a neighbourhood of the minimum point.

We write $\mathcal{M}$ to denote the manifold $\{u : k(u) = 0\}$, and for a point $u \in \mathcal{M}$ we write

$$P(u), Q(u)$$

to denote the orthogonal projections onto the normal and tangent spaces respectively of $\mathcal{M}$ at u. Note that $P + Q = I$.

In what follows, we use tensor notation and sum over repeated indices. We write ∂_j to denote differentiation with respect to u_j, and define

$$g_j = \partial_j f \quad H_{ij} = \partial_i \partial_j f \text{ and } N_{ij} = \partial_j k_i$$

We assume that the constraint normals are linearly independent at the minimum point in $\mathcal{M}$. If N has full rank at u, then the generalised inverse

$$\hat{N} = N'(NN')^{-1}$$

exists at u and satisfies $N\hat{N} = I, \hat{N}N = P$.

2. Strict Second Order Minima. Let λ be a vector of the same dimension as k and define

$$L(u, \lambda) = f(u) - \lambda_i k_i(u)$$

We call $u^* \in \mathcal{M}$ a *strict second order minimum* of f constrained to $\mathcal{M}$ *iff* at u^*

(i) $g_i Q_{ij} = 0$, and

(ii) if λ satisfies $g_j = \lambda_i N_{ij}$ then $z_i z_j (\partial_i \partial_j L) > 0$ for all $z \neq 0$ such that $z_i P_{ij} = 0$.

The first condition says that g lies in the space spanned by the constraint normals, ie that $g_j = \lambda_i N_{ij}$ for some choice of λ. This must be the case at any constrained minimum for f. Under our assumption of constraint independence at u^*, the only such λ is $\lambda^*(u^*)$ where we define

$$\lambda^*_j = g_i \hat{N}_{ij}$$

The second condition says that we have a second order minimum for f along all curvilinear directions away from u^* in $\mathcal{M}$. Now $\partial\partial L = H - A$ where $A_{ij} = \lambda^*_r \partial_i \partial_j k_r$, and certainly $Q(H - A)Q$ must be positive semi-definite at any constrained minimum for f. So since $xPx = xPPx \geq 0$ for all choices of x we have

$$xQ(H - A)Qx + xPx \leq 0 \text{ for some } x \neq 0 \quad \textit{iff}$$

$$zQ(H-A)Qz \leq 0 \text{ for some } z \neq 0 \text{ with } zP = 0$$

(to see this just set $z = xQ$). Consequently the second condition amounts to demanding precisely that $Q(H-A)Q + P$ is positive definite at u^*.

In particular, if f is unconstrained so that $\mathcal{M}$ is the whole of u-space then u^* is a strict second order minimum of f *iff* at u^* (i) $g = 0$ and (ii) H is positive definite.

We can regard $\lambda^*(u)$ as an approximation to the Lagrange multipliers $\lambda^*(u^*)$ for the constrained problem. Constrained minima for f correspond to stationary points of L, and we need only consider points of the form $(u, \lambda^*(u))$.

Now we consider $L(u, \lambda^*(u)) = f(u) - \lambda_i^*(u)k_i(u)$. We note that the term $\lambda_i^* k_i$ can be re-arranged as $g_i n_i$ where

$$n_i = \hat{N}_{ij} k_j$$

For any u define the point $m(u)$ by

$$m_i(u) = u_i - n_i(u)$$

If the constraints k were linear, then $m(u)$ would be the nearest point to u satisfying the constraints. The quantity $n^2(u)$ is thus a first order estimate of the square of the Euclidean distance from u to $\mathcal{M}$. This motivates the following:

Theorem 2.1. Let $\mathcal{M} = \{u : k(u) = 0\}$ where $dim(k) < dim(u)$, and suppose for some open neighbourhood U that that f, k have continuous second derivatives, and that the Jacobian of k has full rank on U. Define

$$\phi(u) = L(m(u), \lambda^*(u)) + \frac{1}{2} n^2(u) \text{ for } u \in U$$

Then a point $u^* \in \mathcal{M} \cap U$ is a strict second order constrained minimum for f *iff* u^* is a strict second order unconstrained minimum for ϕ.

Proof. We have

$$\phi = f^m - \lambda_r^* k_r^m + \frac{1}{2} n_i n_i$$

where we define $f^m(u) = f(m(u)), k_r^m(u) = k_r(m(u))$. Now

$$\partial_p n_i = (\partial_p \hat{N}_{ir}) k_r + \hat{N}_{ir} N_{rp}$$

so

$$\partial_p n_i = P_{ip} \text{ for } u^* \in \mathcal{M}$$

so $\partial_p m_i = I_{ip} - P_{ip} = Q_{ip}$ on $\mathcal{M}$. Now

$$\partial_p \phi = g_i^m (\partial_p m_i) - (\partial_p \lambda_r^*) k_r^m - \lambda_r^* N_{ri}^m (\partial_p m_i) + n_i (\partial_p n_i)$$

and since on $\mathcal{M}$ we have $k = 0, n = 0, \partial m = Q, NQ = 0$ this gives

$$\partial_p \phi = g_i Q_{ip} \text{ for } u^* \in \mathcal{M}$$

Thus ϕ has a stationary point at u^* *iff* $gQ = 0$ at u^*. Similarly

$$\partial_p\partial_q\phi = H^m_{ij}(\partial_p m_i)(\partial_q m_j) + g^m_i(\partial_p\partial_q m_i) - (\partial_p\partial_q\lambda^*_r)k^m_r$$

$$-(\partial_p\lambda^*_r)N^m_{ri}(\partial_q m_i) - (\partial_q\lambda^*_r)N^m_{ri}(\partial_p m_i) - \lambda^*_r(\partial_i\partial_j k^m_r)(\partial_p m_i)(\partial_q m_j)$$

$$-\lambda^*_r N^m_{ri}(\partial_p\partial_q m_i) + (\partial_p n_i)(\partial_q n_i) + n_i(\partial_p\partial_q n_i)$$

so since $\lambda^*_r N_{rj} = g_i\hat{N}_{ir}N_{rj} = g_i P_{ij}$ we have

$$\partial_p\partial_q\phi = Q_{pi}(H_{ij} - A_{ij})Q_{jq} + g_i Q_{ij}(\partial_p\partial_q m_j) + P_{pq} \text{ for } u^* \in \mathcal{M}$$

where

$$A_{pq} = g_i\hat{N}_{ir}\partial_p\partial_q k_r$$

Thus if $gQ = 0$ at $u^* \in \mathcal{M}$ then the Hessian of ϕ is positive definite *iff* $Q(H - A)Q + P$ is positive definite. **QED.**

Under the assumption that f and k are twice differentiable on U, we have shown that ϕ is once differentiable throughout U, and twice differentiable at u^*. In proving this we have implicitly used the fact that if a is continuous and b is differentiable with $b(u^*) = 0$ then the product ab is differentiable at u^* with $\partial(ab) = a(\partial b)$. In fact, ϕ is twice differentiable throughout U provided f and k are three times differentiable on U.

3. Other Penalty Formalisms. Fletcher's original penalty function is essentially

$$\psi_\mu = f - g_i n_i + \frac{\mu}{2} n_i n_i$$

where $\mu > \|H - A + I\|$. That this has similar properties to our ϕ can be seen by noting that

$$\phi = f - g_i n_i + \frac{1}{2} n_i n_j (H_{ij} - A_{ij} + I_{ij})$$

to second order in $\|n(u)\|$ and then making (once and for all) the indicated choice for the parameter μ. Note however that the penalty function ϕ defined here is parameter free. Essentially ϕ is the value of the Lagrangian, not at u, but at the nearest point satisfying the (locally linearised) constraints, and we add a penalty term proportional to the square of this displacement. Instead of forcing $\lambda = \lambda^*(u)$ we could leave the λ as free variables and add a further penalty term of the type $(g - \lambda N)^2$ or some scaling thereof as in Ref. 4. In our definition of ϕ we could alternatively have put $\lambda^*(m(u))$ in place of $\lambda^*(u)$.

Penalty functions are frequently used to transform a constrained optimization problem into an unconstrained problem (see for example Refs. 4,5,6). Such penalty formalisms often require a penalty parameter to be adjusted at each iteration. Frequently a naive proof of termination for a penalty algorithm assumes infinite precision arithmetic (and exact solution of subproblems), and allows arbitrary settings for the penalty parameter. In practice, such a form of the optimization algorithm runs the risk of ill-conditioned steps which could prevent convergence. The strategies used in practice to avoid ill-conditioned choices of penalty parameters and to satisfy inexact convergence criteria for subproblems frequently ruin the original proof of termination.

The analytical consideration of ideal penalty functions such as ϕ can allow convergence properties to be proved by showing that each iteration must lead to an appropriate decrease in ϕ, without having to take explicit account of penalty parameter adjustments (see for example Refs. 7,8.)

However it should also be noted that the techniques of automatic differentiation (Refs. 9,10) have now reached the point where they allow the direct numerical calculation of functions such as ϕ and its derivatives. We conclude this note by indicating briefly one strategy for doing this, suitable for problems with large numbers of variables and constraints.

4. Automatic Differentiation. It is well known (see for example Ref. 11) that the reverse accumulation method of automatic differentiation can be used to extract all components of the gradient vector g of any scalar valued function f for about 3 times the floating point computational cost of a single evaluation of f, where the constant 3 is independent both of the form of f and of the number of parameters (independent variables). Similarly (*op cit*) if b is any constant vector then reverse accumulation can evaluate the entire vector Hb (ie an arbitrary linear combination of rows of the Hessian H of f) for about 6 times the computational cost of a single evaluation of f. This makes reverse accumulation ideal for the case of unconstrained optimization where a truncated Newton method is applied to f. (The less efficient method of forward accumulation is applied to good effect in this context in Ref. 12.)

We can thus evaluate n as follows. Solve the equation $k = NN'x^*$ for x^* using automatic differentiation to evaluate NN'. Then $n = N'x^*$. Similarly λ^* is the solution of $gN' = \lambda^* NN'$ where g is calculated by reverse accumulation. Note that both λ^* and x^* are of the same dimension as k, rather than of u.

Once the vectors n and λ^* have been obtained, it is a simple matter to compute the value of the ideal penalty function ϕ. Consequently direct numerical use of an ideal penalty function such as ϕ as a validiation step to ensure convergence in solving optimization problems is now computationally feasible. But we can do more.

We show in Ref. 13 how reverse accumulation can be used to differentiate automatically functions such as ϕ which are formed by combining values such as N and g which were themselves obtained by reverse accumulation (see also Refs. 14, 15). The computation of ϕ can be made available in a form which is itself susceptible to automatic differentiation, and the extraction of gradients, directional or full Hessians and so forth. These can in turn be used by optimization software to find a local minimum point u^* of ϕ, which will correspond to the solution of the original constrained problem. Finally we can apply automatic differentiation to the components of u^* so as to perform an automatic error analysis or determine the sensitivities of the solution.

It is worth noting that we may use an iterative method of solving the linear equations for n and λ^*. This is particularly attractive if there is a large number of constraints. For example, Pearlmutter (Ref. 16) in the context of neural networks suggests applying a conjugate gradient algorithm (such as Ref. 17) to the quadratic problem: find λ^* to minimize $(\lambda NN' - gN')^2$. Similarly finding x^* to minimize $(NN'x - k)^2$ gives $n = N'x^*$. We show in Ref. 13 that reverse accumulation can be used to obtain vectors of the form $NN'x$, and $(\lambda N - g)N'$ for fixed vectors λ and x at a constant multiple of the computational cost of evaluating k, even when the components of k share intermediate variable values

in the computation. Techniques are also known which allow gradients to be extracted for functions which include in their construction the iterative solution of fixed point equations such as those for λ^* and x^* (Refs. 18, 19, 20).

Since most optimization codes can also be regarded as iterative contractive differentiable mappings, at least in a neighbourhood of the fixed point u^*, these iterative fixed point techniques can also be applied to the final optimization step to extract sensitivities of u^*. A further advantage of using reverse accumulation with this iterative fixed point formulation is that the sensitivities which reverse accumulation provides allow automatic error estimates to be made for the effect of truncating subproblem solution on the calculated function value (see Ref. 19).

Thus an iterative formulation allows us to solve the equations for λ and x with just sufficient accuracy to ensure that the calculated value of $\phi(u)$ is correct to the required accuracy (specified in advance) at each iteration step of the optimization algorithm.

5. Conclusion. In this note we have given a geometric formulation of a differentiable penalty function similar to those considered by Fletcher, but requiring no parameter estimation or adjustment. The strict second order constrained minima of the target function are precisely those strict second order unconstrained minima of the penalty function which satisfy the constraints.

Our penalty function also has the desirable property that near such an minimum point the penalty function has the same curvature as the Lagrangian of the target function in directions tangent to the constraint manifold, and unit positive curvature in directions normal to the constraint manifold. Near a minimum point, our penalty function thus has numerical conditioning similar to that of the target function and constraints from which it is constructed.

The penalty function can be used to establish theoretical termination properties for algorithms. Alternatively numerical values for the penalty function can be efficiently calculated using automatic differentiation techniques and used to validate a particular run of an algorithm. However it should also be possible to apply truncated Newton methods directly to the penalty function in order to find the optimal point for the target function.

In this note we have formulated an approach only for equality constrained problems. Inequality constrained problems are considered in Refs. 21 and 5.

References.

[1] FLETCHER, R., *A Class of Methods for Nonlinear Programming with Termination and Convergence Properties,* Integer and Nonlinear Programming, Edited by J. Abadie, North Holland, pp. 157–175, 1970.

[2] FLETCHER, R., and LILL, S., *A Class of Methods for Nonlinear Programming II: Computational Experience*, Nonlinear Programming, Edited by J.B. Rosen *et al*, Academic Press, pp. 67–92, 1970.

[3] FLETCHER, R., *A Class of Methods for Nonlinear Programming III: Rates of Convergence,*, Numerical Methods for Nonlinear Optimization, Edited by F.A. Lootsma, Academic Press, pp. 371–393, 1973.

[4] DI PILLO, G., and GRIPPO, L., *A New Class of Augmented Lagrangians in Nonlinear Programming,* Society for Industrial and Applied Mathematics Journal of Control and Optimization, Vol 17, No 5, pp. 618–628, 1979.

[5] DI PILLO, G., *et al*, *An RQP Algorithm using a Differentiable Exact Penalty Function for Inequality Constrained Problems*, Mathematical Programming, Vol 55, pp. 49–68, 1992.

[6] BERTSEKAS, D.P., *Enlarging the Region of Convergence of Newton's Method for Constrained Optimization,* Journal of Optimization Theory and Applications, Vol 36, No 2, pp. 221–252, 1982.

[7] BIGGS, M.C., *On the Convergence of some Constrained Minimization Algorithms Based on Recursive Quadratic Programming*, Journal of the Institute of Mathematics and its Applications, Vol 21, pp. 67–81, 1978.

[8] POWELL, M.J.D., and YUAN, Y., *A Recursive Quadratic Programming Algorithm that uses Differentiable Exact Penalty Functions,* Mathematical Programming, Vol. 35, pp. 265–278, 1984.

[9] GRIEWANK, A., *On Automatic Differentiation,* Mathematical Programming: Recent Developments and Applications, Edited by M. Iri and K. Tanabe, Kluwer Academic Publishers, Japan, pp. 83–108, 1989.

[10] GRIEWANK, A., and CORLISS, G., Editors, *Automatic Differentiation of Algorithms,* Society for Industrial and Applied Mathematics, Philadelphia, 1991.

[11] CHRISTIANSON, B., *Automatic Hessians by Reverse Accumulation,* Institute of Mathematics and its Applications Journal of Numerical Analysis, Vol. 12, pp. 135–150, 1992.

[12] DIXON, L.C.W., and PRICE, R.C., *Truncated Newton Algorithms for Large Scale Unconstrained Optimization using Automatic Differentiation,* Journal of Optimization Theory and Applications, Vol. 60, No. 2, pp. 261–275, 1989.

[13] CHRISTIANSON, B., *Reverse Accumulation of Functions containing Gradients,* University of Hertfordshire, Numerical Optimisation Centre Technical Report No. 278, Hatfield, England, 1993. An extended abstract appears in: *Procedings of the Theory Institute on Combinatorial Challenges in Automatic Differentiation*, Argonne National Laboratories, Illinois, 1993.

[14] KUBOTA, K., *An Implementation of Fast Automatic Differentiation with C++,* Abstracts of the Spring 1989 Meeting of the Operations Research Society of Japan, pp. 175–176, 1989. *(in Japanese)*

[15] BISCHOF, C., *et al*, *Structured Second- and Higher-Order Derivatives through Univariate Taylor Series,* Optimization Methods and Software, *to appear.*

[16] Pearlmutter, B., *Fast Exact Multiplication by the Hessian,* Neural Computing, *to appear.*

[17] Yoshida, T., *Rapid Learning Method for Multilayered Neural Networks using Two-Dimensional Conjugate Gradient Search,* Journal of Information Processing, Vol. 15, No. 1 pp. 79–86, 1992.

[18] Gilbert, J.C., *Automatic Differentiation and Iterative Processes,* Optimization Methods and Software, Vol. 1, No. 1, pp. 13 –21, 1992.

[19] Christianson, B., *Reverse Accumulation and Attractive Fixed Points,* Optimization Methods and Software, *to appear.*

[20] Griewank, A., *et al, Derivative Convergence for Iterative Equation Solvers,* Optimization Methods and Software, *to appear.*

[21] Fletcher, R., *An Exact Penalty Function for Nonlinear Programming with Inequalities,* Mathematical Programming, Vol. 5, pp. 129–150, 1973.