The VMCON optimisation solver explained

The PROCESS team: P. J. Knight, M. D. Kovari, H. Lux, J. Morris

Culham Centre for Fusion Energy/ United Kingdom Atomic Energy Authority
Culham Science Centre, Abingdon, Oxon, OX14 3DB, UK

To give the user a better understanding of the optimisation solver implemented in `PROCESS` and the interpretation of its results, we give a short introduction into the mathematical background for solving these type of problems as well as the specific algorithm used.

In section 1, the general nonlinear programming problem is defined, which is the mathematical description of our problem. This problem is typically formulated using Lagrange multipliers (c.f. 2) and is solved numerically most efficiently using sequential quadratic programming (c.f. 3). The Fortran subroutine that is used to implement such an optimisation solver in `PROCESS` is called `VMCON` (c.f. 4), which iterates between solving a local quadratic subproblem (c.f. 5) and a one dimensional line search (c.f. 6). As the method uses a quasi-Newtonian approach the Hessian matrix is approximated by a variant of the Broyden-Fletcher-Goldfarb-Shanno update (7). Section 8 summarises the symbol convention used in this chapter.

1 The General Nonlinear Programming Problem

Mathematically the *general nonlinear programming problem* or *nonlinear constrained optimisation problem* is defined as

$$\text{minimise } f(\boldsymbol{x}), \tag{1a}$$

$$\text{subject to } c_i(\boldsymbol{x}) = 0, \qquad i = 1, \ldots, k, \tag{1b}$$

$$\text{and } c_i(\boldsymbol{x}) \geq 0, \qquad i = k+1, \ldots, m, \tag{1c}$$

where both the *objective function*[1] $f(\boldsymbol{x})$ and the *constraints* $c_i(\boldsymbol{x})$ are nonlinear functions of the n-dimensional vector of variables $\boldsymbol{x}$ with bounds $\boldsymbol{x} \in \Omega$. In this context, all $\boldsymbol{x} \in \Omega$ that fulfill the constraints $c_i(\boldsymbol{x})$ are called *feasible*. They describe the allowed space in which we are trying to optimise the objective function $f(\boldsymbol{x})$. Note that any maximisation problem can be written as a minimisation by using $f_{\text{new}}(\boldsymbol{x}) = -f(\boldsymbol{x})$ and that any equality constraint $c(\boldsymbol{x}) = a$ can be rewritten as $c_{\text{new}}(\boldsymbol{x}) = c(\boldsymbol{x}) - a = 0$. Any inequality constraint can therefore be rearranged analogously to fit the form described in eq. 1c.

2 The Lagrange Method

The general nonlinear programming problem can be solved mathematically using Lagrange's method. It assumes that the constraints cannot be used to explicitly reduce the parameter space of the iteration variables - as it is typically the case for non-linear constraints and objective functions - and is therefore a powerful method applicable to a general class of problems.

If we assume for simplicity that we have a 2D problem with only one equality constraint $c(x, y) = 0$, we know that we only need to search for the optimum along that constraint. At the optimum, the value of the objective function will then be stationary, i.e. it does not locally increase or decrease along the constraint. As the gradient of a function is perpedicular to its contour lines of $f(x, y) = d$, this is equivalent to saying that the gradient of the objective function at that point is parallel to the gradient of the constraints:

$$\nabla f(x, y) = -\lambda \nabla c(x, y), \tag{2}$$

where the factor λ is necessary as only the direction, but not the magnitude nor the sign of the gradients need to be equal. This is also illustrated in Figure 1.

When expanding the method to several equality and inequality constraints we can make use of the *Lagrange function.* For the nonlinear programming problem described by 1 it is given by

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \sum_{i=1}^{m} \lambda_i c_i(\boldsymbol{x}), \tag{3}$$

with the corresponding *Lagrangian multipliers* λ_i. It allows us to formulate the *first order necessary* conditions for a constrained optimum $\boldsymbol{x}^*$ with corresponding Lagrange multipliers $\boldsymbol{\lambda}^*$, the Karush-

[1]Please note that what is called *figure of merit* in `PROCESS` is called *objective function* in the mathematical optimisation context. Hence, both names are used equivalently in this document.

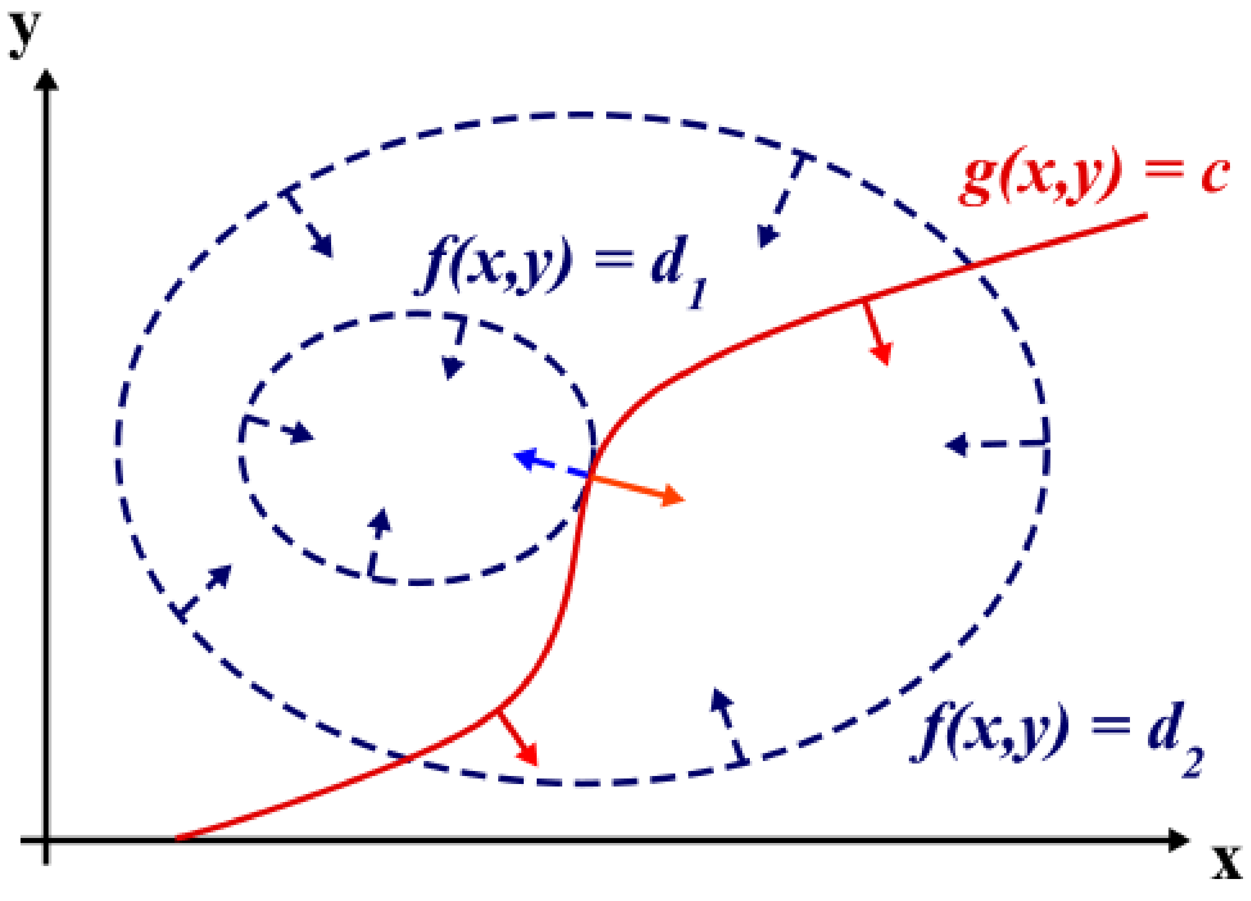


Figure 1: *Illustration of Lagrange Multiplier Method (credit Wikipedia) showing two contour lines of the objective function* $f(x,y) = d_i$ *(dark blue dashed lines) and the nonlinear constraint* $g(x,y) = c$ *(red solid line) as well as their gradients (blue and red arrows) at various positions including the constrained optimum (light blue and orange arrows).*

Kuhn-Tucker (KKT) conditions,

$$
\begin{aligned}
\nabla_x L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) = \nabla_x f(\boldsymbol{x}^*) - \sum_{i=1}^{m} \lambda_i \nabla_x c_i(\boldsymbol{x}^*) = 0, & & \text{(4a)}\\
\lambda_i^* c_i(\boldsymbol{x}^*) = 0, & \quad i = 1, \ldots, m, & \text{(4b)}\\
c_i(\boldsymbol{x}^*) = 0, & \quad i = 1, \ldots, k, & \text{(4c)}\\
\lambda_i^* \geq 0, & \quad i = k+1, \ldots, m, & \text{(4d)}\\
c_i(\boldsymbol{x}^*) \geq 0, & \quad i = k+1, \ldots, m. & \text{(4e)}
\end{aligned}
$$

Please note that by construction the Lagrange multipliers λ_i^* fulfilling the KKT conditions are describing the derivative of the objective function with respect to the constraint equation $\frac{df}{dc_i}$ and are therefore a measure of how much the objective function changes with respect to each constraint.

In the special case of a *continuously differentiable convex* objective function $f(\boldsymbol{x})$ and equality constraints as well as *affine* inequality constraints, these KKT conditions are also sufficient for a global optimum. The `PROCESS` optimisation solver has been designed to converge on the KKT conditions, but does not test whether these are *sufficient* for a global optimum. It is therefore crucial that the user verifies that a global optimum has been found.

Furthermore, these conditions and therefore the solver, assume that both the objective function and constraints are at least *first order continuously partially differential*, i.e. that their first order partial derivatives are all continuous functions. This might not always be the case in `PROCESS` and is a potential source of errors.

3 Sequential Quadratic Programming (SQP)

Based on the Lagrange method, sequential (also successive or recursive) quadratic programming is the most efficient method to numerically solve constrained nonlinear optimisation problems [1]. It combines a simple solver for a quadratic optimisation problem with linear constraints that determines a search direction $\boldsymbol{\delta}$ with a line search to find an optimum along that search direction. It is a type of the more general *feasible direction* methods which is itself a type of *primal method* that solve nonlinear programming problems in the $n - m$ dimensional feasible subspace [2].

4 `VMCON`

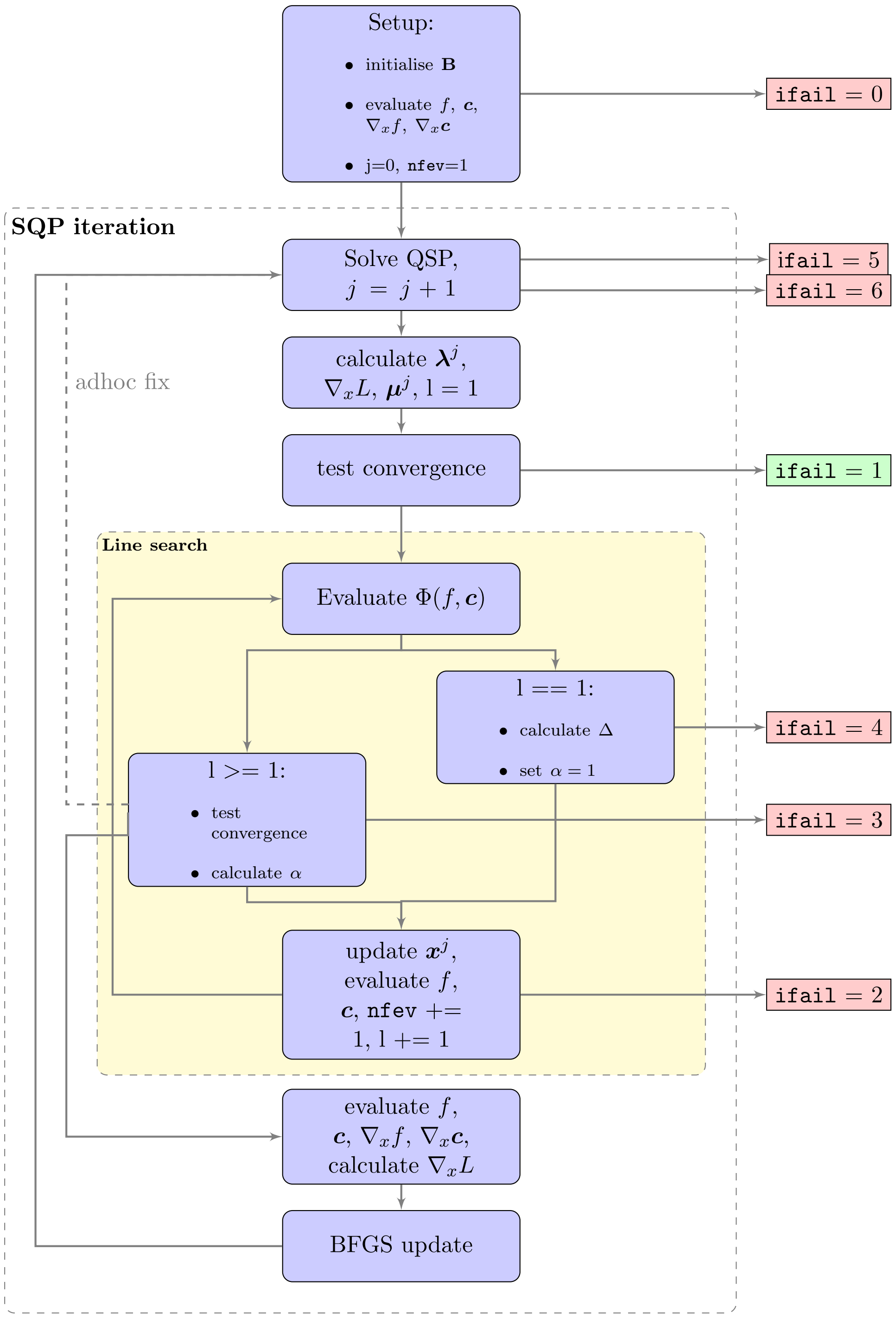
The optimisation solver implemented in `PROCESS` is the Fortran routine `VMCON` [3]. It is a modified version of the `vf02ad` routine from the Harwell Software Library[2] and implements a SQP method originally suggested by Powell[3] [5] based on work by Han [6]. As stated before `VMCON` is designed to converge on a solution of the *necessary* KKT conditions 4, but does not check the *sufficient* conditions. Its convergence criterion is therefore given by

$$
\left|\nabla_x f(\boldsymbol{x}^{j-1})^T \cdot \boldsymbol{\delta}^j\right| + \sum_{i=1}^{m} \left|\lambda_i^j c_i(\boldsymbol{x}^{j-1})\right| < \texttt{epsvmc} \tag{5}
$$

where `epsvmc` is a user specified error tolerance, $\boldsymbol{\delta}^j$ is the vector in direction of the next line search (c.f. section 6) and j is the counter of the sequential quadratic programming iteration. Hence, the first

[2]www.hsl.rl.ac.uk

[3]This should not be confused with Powell's algorithm [4] which solves a multidimensional unconstrained minimisation problem without derivatives.



Setup:
initialise $\mathbf{B}$
evaluate f, $\boldsymbol{c}$, $\nabla_x f$, $\nabla_x \boldsymbol{c}$
j=0, nfev=1
ifail = 0
SQP iteration
Solve QSP, $j = j + 1$
ifail = 5
ifail = 6
adhoc fix
calculate $\boldsymbol{\lambda}^j$, $\nabla_x L$, $\boldsymbol{\mu}^j$, l = 1
test convergence
ifail = 1
Line search
Evaluate $\Phi(f, \boldsymbol{c})$
l == 1:
calculate Δ
set $\alpha = 1$
ifail = 4
l >= 1:
test convergence
calculate α
ifail = 3
update $\boldsymbol{x}^j$, evaluate f, $\boldsymbol{c}$, nfev += 1, l += 1
ifail = 2
evaluate f, $\boldsymbol{c}$, $\nabla_x f$, $\nabla_x \boldsymbol{c}$, calculate $\nabla_x L$
BFGS update

Figure 2: *This is the flow chart of the* `VMCON` *optimisation solver. The criteria for and the interpretation of the successful (*`ifail` *= 1) and unsuccessful (*`ifail` *$\neq$ 1) return parameters are described in Table 1.*

`ifail`	Description	Meaning	Recommendation
0	`VMCON`: Improper input parameters	The input parameters to the solver are wrong, e.g. negative number of iteration variables.	This needs to be fixed on the developer side and should only occur in the test phase of new modules.
1	`VMCON`: Normal return	`VMCON` has found a solution that fulfills the necessary conditions within the specified error tolerances (c.f. eq. 5).	Test whether the solution is a global solution by using different starting parameters.
2	Too many function calls	During the line search `VMCON` has reached the maximum number of total function calls (`maxfev`=100)	`VMCON` struggles to find a solution. Retry with different start parameters.
3	Line search required 10 functions calls	The results produced by input objective function/constraints and their gradients are inconsistent. This can be due to numerical noise in the input functions.	The developer needs to check the consistency and numerical robustness of the objective function, constraints and their derivatives. Perhaps the accurracy in numerical integrations/differentiations needs to be higher. As a user, try changing the iteration bounds or adding other iteration variables.
4	Uphill search direction was calculated	The quadratic subproblem has suggested a search direction in which the objective function only increases.	This happens if an inconsistency between the objective function, the constraints and their respective derivatives occurs (c.f. `ifail` =3).
5	`qpsub`: No feasible solution or bad approximation of Hessian	Either no optimum lies within the space allowed by the constraints and variable bounds or the identity matrix is not a good first approximation of the Hessian.	As a user, add a new iteration variable, as developer, try using a multiple of the identity matrix as initial Hessian instead.
6	`qpsub`: Singular matrix in quadratic subproblem or restriction by artificial bounds	This is fairly self-explanatory.	If this is meaningful, widen the boundaries of the iteration variables.

Table 1: Summary of the description and meaning of the `VMCON` return parameters `ifail`.

part estimates the predicted improvement due to the next line search and the second part measures the error in the complimentary condition 4b of the KKT conditions.

Figure 2 describes the flow chart of the `VMCON` routine, while the various values of the return parameter `ifail`[4] are described and interpreted in Table 1.

5 The Quadratic Subproblem (QSP)

Within sequential quadratic programming the complex nonlinear problem is broken down into solving a sequence of local quadratic subproblems with linear constraints. This is based on the assumption that locally the problem is well approximated by a second order Taylor expansion of the Lagrange function. Hence, the local quadratic subproblem is described by

$$\text{minimise } Q(\boldsymbol{\delta}) = f(\boldsymbol{x}^{j-1}) + \boldsymbol{\delta}^T \nabla_x f(\boldsymbol{x}^{j-1}) + \frac{1}{2}\boldsymbol{\delta}^T \nabla_{xx} L(\boldsymbol{x}^{j-1}, \boldsymbol{\lambda}^{j-1})\boldsymbol{\delta} \tag{6a}$$

$$\text{subject to } \boldsymbol{\delta}^T \nabla_x c_i(\boldsymbol{x}^{j-1}) + c_i(\boldsymbol{x}^{j-1}) = 0,\ i = 1, \ldots, k, \tag{6b}$$

$$\text{and } \boldsymbol{\delta}^T \nabla_x c_i(\boldsymbol{x}^{j-1}) + c_i(\boldsymbol{x}^{j-1}) \geq 0,\ i = k+1, \ldots, m, \tag{6c}$$

where $\boldsymbol{\delta} = \boldsymbol{x} - \boldsymbol{x}^{j-1}$, the index $j-1$ indicates the parameter values of the previous iteration[5] and $\nabla_{xx} L(\boldsymbol{x}^{j-1}, \boldsymbol{\lambda}^{j-1})$ is the Hessian of the Lagrange function. The solution of the QSP $\boldsymbol{\delta}$ describes the change of the current iteration variable vector that minimises the local approximation of the problem. To assure convergence even from bad starting points, it is not directly used to update the iteration variable vector, but describes the direction of the line search in which a 1d function is minimised using a Newtonian method (c.f. section 6). Being a second order approximation to the original problem, it typically has a faster convergence that sequential linear programming (SLP) methods [7, chap. 10.4].

To allow the applicability of the solver to more general problems, Powell [5] substituted the Hessian $\nabla_{xx} L(\boldsymbol{x}^{j-1}, \boldsymbol{\lambda}^{j-1})$ with a positive definite approximation $\mathbf{B}$. This means that both the objective function $f(\boldsymbol{x})$ and the constraint equations $c_i(\boldsymbol{x})$ only have to be continuously differentiable instead of twice continuously differentiable with respect to the iteration variable vector $\boldsymbol{x}$. This makes the method a Quasi-Newtonian method as opposed to true Newtonian methods. How this approximation is initialised and updated is described in more detail in the section about the Broyden-Fletcher-Goldfarb-Shanno update 7.

To solve the QSP `VMCON` uses `harwqp` a modification of the the Harwell library routine `VE02AD`, which in itself uses the subroutine `harwfp/LA02AD` to find a feasible point within the variable bounds and linearised constraints. Both routines go back to a method by Fletcher [8, 9, 10]. The Lagrange multipliers are also determined from results of the `harwqp` routine.

If the routine cannot find a feasible point it fails with `ifail` = 5 (c.f. Table 1). As the routine is only checking the local linear approximation of the constraints rather the full non-linear constraints, there is a chance that a feasible point exists even though the routine fails with `ifail` = 5. In these cases, it is possible that the first approximation of the Hessian has not been good and the algorithm has, therefore, taken an inappropriately large step. Then using a multiple of the identity matrix will improve convergence of the algorithm.

If a singular matrix is encountered with the QSP solver or the solution is restricted by the artificial bounds, `VMCON` fails with `ifail` = 6. In this case it can be helpful to widen the boundaries of the iteration variables.

[4]Note, within the `VMCON` routine `ifail` is called `info`.

[5]Note j is called `nqp` in the `VMCON` routine.

6 The Line Search

The line search is an essential part of the SQP algorithm. As Powell [5] pointed out, it is necessary to allow convergence from poor starting conditions. It uses the vector $\boldsymbol{\delta}$ from the QSP to update $\boldsymbol{x}^j = \boldsymbol{x}^{j-1} + \alpha^j \boldsymbol{\delta}^j$ where the step-length parameter $\alpha^j > 0$ is determined by the line search as the parameter that minimises

$$\Phi(\alpha) = f(\boldsymbol{x}) + \sum_{i=1}^{k} \mu_i \, |c_i(\boldsymbol{x})| + \sum_{i=k+1}^{m} \mu_i \, |min(0, c_i(\boldsymbol{x}))| \tag{7}$$

where the weights are defined as

$$\mu_i = \begin{cases} |\lambda_i^1| & \text{if } j = 1, \\ max\left(|\lambda_i^j|, 1/2(\mu_i^{j-1} + |\lambda_i^j|)\right) & \text{if } j > 1 \end{cases} \tag{8}$$

to assure maximally efficient convergence [6]. Note, that in this method locally *inactive* inequality constraints (see Userguide) are not having any effect. It should always be possible, to find a solution that fulfills

$$\Phi(\alpha = 0) > \Phi(\alpha^j), \tag{9}$$

if

$$\left.\frac{d\Phi}{d\alpha}\right|_{\alpha=0} < 0. \tag{10}$$

In case the derivative is positive (`dflsa` ≥ 0), an uphill search direction has been determined and the code stops with `ifail` $= 4$. This typically only happens, if the objective function or constraints are inconsistent with their own derivatives.

As the line search tries to determine the optimum of a one dimensional, but fully non-linear function $\Phi(\alpha)$, it creates a series of α_l values[6]. At each iteration l, a quadratic local function $\Phi_l(\alpha)$ fullfilling the boundary conditions $\Phi_l(0) = \Phi(0)$, $\Phi_l'(0) = \Delta$ and $\Phi_l(\alpha_{l-1}) = \Phi(\alpha_{l-1})$ is minimised, where typically $\Delta = \Phi'(0)$. This leads to

$$\Phi_l(\alpha) = \Phi(0) + \Delta\alpha + \frac{\Phi(\alpha_{l-1}) - \Phi(0) - \Delta\alpha_{l-1}}{\alpha_{l-1}^2}\alpha^2 \tag{11}$$

and minimising this gives

$$\alpha_{min} = -\frac{\Delta\alpha_{l-1}^2}{2(\Phi(\alpha_{l-1}) - \Phi(0) - \Delta\alpha_{l-1})}. \tag{12}$$

Powell then sets $\alpha_l = \min(\alpha_{min}, 0.1\alpha_{l-1})$. On each iteration the convergence of the line search is tested. It is reached, if

$$\Phi(\alpha_l) - \Phi(0) < 0.1\Delta. \tag{13}$$

which assures that the change in the function is small in comparison to its derivative and 0.1 is a factor determined by experience. If this criterion is successful, the code exits the line search and updates all function evaluations.

As a modification to the original code, we have added an adhoc fix that exits the line search in the case of

$$\Phi(\alpha_l) - \Phi(0) > 0. \tag{14}$$

This has been added as experience have shown that `VMCON` typically does not converge in these situations, but if it is forced to caculate a new search direction in this way, it sometimes successfully

[6]In the actual code $\alpha =$`calpha` and $\alpha_l =$`alpha`$*\alpha_{l-1}$.

finishes. Note, that this cannot force `VMCON` to converge on any false solutions, as it only exits positively when the convergence criterion 5 is fulfilled.

Typically, the line search already converges after one iteration and, therefore, $\alpha = 1$. Hence, the `VMCON` line search has an upper limit of maximally 10 iterations before it terminating with `ifail` = 3 (c.f. Table 1). This is higher than Powell's original limit of 5 to avoid a few cases of early termination without a major effect on efficiency.

Within the line search `VMCON` also checks that the total number of function calls has not exceeded `maxfev` = 100. If this is the case, it exits with error code `ifail` = 2. Both checks assure that the routine stops, if it does not seem to be able to find a solution.

7 The Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton update

`VMCON` uses a quasi-Newtonian update, the Broyden-Fletcher-Goldfarb-Shanno update, in the approximation of the Hessian of the Lagrange function. This means it is applicable to all continuously differentiable objective and constraint functions. Note, that due to the constraints it is not actually necessary for the Hessian of the solution to be positive definite, even though this is essential for the convergence of the algorithm.

In quasi-Newtonian methods the identity matrix $\mathbf{I}$ is often chosen as an initial estimate of the Hessian because of its positive definiteness. In some cases it can be helpful to chose a constant multiple of the identity matrix instead. The BGFS update [11] is one way of revising the initial Hessian approximation using the update of the iteration variable vector

$$\boldsymbol{\xi} = \boldsymbol{x}^j - \boldsymbol{x}^{j-1} \tag{15}$$

and the update of the Jacobian of the Lagrange function

$$\boldsymbol{\gamma} = \nabla_x L(\boldsymbol{x}^j, \boldsymbol{\lambda}^j) - \nabla_x L(\boldsymbol{x}^{j-1}, \boldsymbol{\lambda}^j). \tag{16}$$

Note that unless $\alpha = 1$ in the line search, $\boldsymbol{\xi} \neq \boldsymbol{\delta}$. To assure the positive definiteness of $\mathbf{B}$ Powell [5] suggested a variation of the standard formalism that uses $\boldsymbol{\eta}$ instead of $\boldsymbol{\gamma}$

$$\boldsymbol{\eta} = \theta\boldsymbol{\gamma} + (1-\theta)\mathbf{B}\boldsymbol{\xi} \tag{17}$$

with

$$\theta = \begin{cases} 1 & \boldsymbol{\xi}^T\boldsymbol{\gamma} \geq 0.2\boldsymbol{\xi}^T\mathbf{B}\boldsymbol{\xi} \\ \frac{0.8\boldsymbol{\xi}^T B\boldsymbol{\xi}}{\boldsymbol{\xi}^T} & \boldsymbol{\xi}^T\boldsymbol{\gamma} < 0.2\boldsymbol{\xi}^T\mathbf{B}\boldsymbol{\xi} \end{cases} \tag{18}$$

to calculate a new approximation of the Hessian

$$\mathbf{B}_{new} = \mathbf{B} - \frac{\mathbf{B}\boldsymbol{\xi}\boldsymbol{\xi}^T\mathbf{B}}{\boldsymbol{\xi}^T\mathbf{B}\boldsymbol{\xi}} + \frac{\boldsymbol{\eta}^T\boldsymbol{\eta}}{\boldsymbol{\xi}^T\boldsymbol{\eta}}. \tag{19}$$

Using the modified version of the BFGS update as suggested by Powell, furthermore, assures superlinear convergence, even if the true Hessian is indefinite [12].

8 Symbols

In the previous sections the following conventions have been assumed: $\boldsymbol{x}$, $\boldsymbol{\delta}$, $\boldsymbol{\xi}$, $\boldsymbol{\gamma}$ and $\boldsymbol{\eta}$ are n-dimensional vectors, where n is the number of iteration variables. $\boldsymbol{c}$, $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$ are m-dimensional vectors, where m is the total number of constraints and c_i, λ_i and μ_i $(i = 1\ldots m)$ are their respective components. $\mathbf{B}$ and $\mathbf{I}$ are $n \times n$-dimensional matrices, while $\nabla_x \boldsymbol{c}$ is an $n \times m$-dimensional matrix.

References

[1] P.J. Knight. *Nonlinear Programming Codes.* Springer, Berlin, 1980.

[2] D. G. Luenberger and Yinyu Ye. *"Linear and Nonlinear Programming"*. International Series in Operations Research and Management Science. Springer Science and Business Media LCC,, 3rd edition edition, 2008.

[3] R.L. Crane, K.E. Hillstrom, and M. Minkoff. Solution of the general nonlinear programming problem with subroutine vmcon. *Argonne National Laboratory Report ANL-80-64*, 1980.

[4] M.J.D. Powell. *An efficient method for finding the minimum of a function of several variables without calculating derivatives.* Computer Journal, 1964.

[5] M.J.D. Powell. A fast algorithm for nonlinearly constrained optimization calculations. *Lecture Notes in Mathematics, Springer-Verlag, Berlin*, 630:144–157, 1978.

[6] S.-P. Han. *A globally convergent method for nonlinear programming.* Technical Report 75-257, Department for Computer Science, Cornell University, 1975.

[7] H. D. Sherali M. S. Bazaraa and C. M. Shetty. *"Nonlinear Programming: Theory and Algorithms"*. John Wiley & Sons, Inc., New York, 1993.

[8] R. Fletcher. *A general quadratic programming algorithm.* Technical Report T.P. 401, Atomic Energy Research Establishment, 1970.

[9] R. Fletcher. *The calculation of feasible points for linearly constrained optimisation problems.* Technical Report R.6354, Atomic Energy Research Establishment, 1970.

[10] R. Fletcher. *A fortran subroutine for general quadratic programming.* Technical Report R.6370, Atomic Energy Research Establishment, 1970.

[11] M. Avriel. Nonlinear programming: Analysis and methods. *Dover Publications, Inc., Mineola, NY*, 2003.

[12] M.J.D. Powell. *The convergence of variable metric methods for non-linearly constrained optimisation calculations.* presented at Nonlinear Programming Symposium 3, Madison, Wisconsin,, 1977.