LINE SEARCH PROCEDURES BASED ON QUASI-NEWTON AND CONJUGATE GRADIENT DIRECTIONS

Mr. Phaichayon Sirisathienwatthana

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science in Applied Mathematics

Suranaree University of Technology

Academic Year 2002

ISBN 974-533-245-3

กระบวนการค้นหาตามเส้นในทิศทางกึ่งนิวตันและเกรเดียนท์สังยุค

นายไพชยนต์ สิริเสถียรวัฒนา

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาคณิตศาสตร์ประยุกต์
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2545

ISBN 974-533-245-3

LINE SEARCH PROCEDURES BASED ON QUASI-NEWTON AND CONJUGATE GRADIENT DIRECTIONS

Suranaree University of Technology has approved this thesis submitted in partial fulfillment of the requirements for a Master's Degree

Thesis Examining Committee

....

(Assoc. Prof. Dr. Nikolay P. Moshkin)

Chairperson

....

(Assoc. Prof. Dr. Prapasri Asawakun)

Member (Thesis Advisor)

....

(Prof. Dr. Sergey V. Meleshko)

Member

....

(Assoc. Prof. Dr. Tawit Chitsomboon)

Vice Rector for Academic Affairs

....

(Assoc. Prof. Dr. Prasart Suebka)

Dean of the Institute of Science

ไพชยนต์ สิริเสถียรวัฒนา : กระบวนการค้นหาตามเส้นในทิศทางกึ่งนิวตันและเกรเดียนท์สังยุค (LINE SEARCH PROCEDURES BASED ON QUASI-NEWTON AND CONJUGATE GRADIENT DIRECTIONS)
อ. ที่ปรึกษา: รศ. ดร. ประภาศรี อัศวกุล, 89 หน้า. ISBN 974-533-245-3

วิทยานิพนธ์นี้ศึกษากระบวนการค้นหาตามเส้นในทิศทางกึ่งนิวตันและเกรเดียนท์สังยุค เพื่อการแก้ปัญหาค่าต่ำสุดแบบไม่มีเงื่อนไข โดยใช้เทคนิคการถอยกลับ เงื่อนไขของวูล์ฟ เงื่อนไขที่แกร่งกว่าของวูล์ฟ และกฎของอาร์มีโฮ สำหรับการเลือกระยะความยาวขั้นในทิศทางที่ใช้หาจุดต่ำสุด ได้นำเสนอทิศทางค้นหาที่เกิดจากการรวมทิศทางกึ่งนิวตัน ทิศทางเกรเดียนท์สังยุค และทิศทางเชิงลดชันสุด ทำให้เกิดทิศทางผสมแบบต่าง ๆ และได้ทำการทดสอบประสิทธิภาพของทิศทางผสมโดยเปรียบเทียบกับการค้นหาในทิศทางเดี่ยว ปัญหาที่ใช้ในการทดสอบเป็นปัญหามาตรฐานที่ใช้ในการทดสอบการหาค่าต่ำสุดแบบไม่มีเงื่อนไขของมอร์เร่ การ์โบว์ และฮิลล์สตรอม (1981) ผลทดสอบเชิงตัวเลขแสดงให้เห็นว่า ทิศทางผสมสามารถช่วยลดจำนวนรอบของการทำซ้ำและจำนวนครั้งของการคำนวณค่าฟังก์ชันในกระบวนการค้นหาตามเส้น

สาขาวิชาคณิตศาสตร์ ลายมือชื่อนักศึกษา__________________

ปีการศึกษา 2545 ลายมือชื่ออาจารย์ที่ปรึกษา__________________

PHAICHAYON SIRISATHIENWATTHANA: LINE SEARCH PROCEDURES BASED ON QUASI-NEWTON AND CONJUGATE GRADIENT DIRECTIONS

THESIS ADVISOR: ASSOC. PROF. PRAPASRI ASAWAKUN, Ph. D. 89 PP. ISBN 974-533-245-3

UNCONSTRAINED MINIMIZATION/QUASI-NEWTON/

CONJUGATE GRADIENT/STEEPEST DESCENT/HYBRID DIRECTIONS

The line search procedures based on quasi-Newton and conjugate gradient directions for solving the unconstrained minimization problems are investigated in this thesis. Backtracking techniques, Wolfe conditions, strong Wolfe conditions and Armijo's rule are used as the criteria for choosing the step length along the search directions. Combinations of these directions and steepest descent direction to produce the hybrid directions are also proposed in this thesis. Significant reduction on the number of iterations and function evaluations are demonstrated on the standard test problems of Moré, J.J., Garbow, B.S., and Hillstrom, K.E. (1981) as results of the search along the proposed hybrid directions within the line search framework.

School of Mathematics Signature of Student ____________________

Academic Year 2002 Signature of Advisor ____________________

Acknowledgements

I would like to express my sincere gradtitude to my thesis advisor, Assoc. Prof. Dr. Prapasri Asawakun. Without her long term guidance, encouragement, support and patient help, this thesis could not have been carried out.

I also would like to express my sincere thanks to all the teachers who taught and helped me during my studies at Suranaree University of Technology. They are Prof. Dr. Sergey V. Meleshko, Assoc.Prof. Dr. Boris I. Kovasov, Assoc. Prof. Dr. Nikolay P. Moshkin, Asst. Prof. Dr. Eckart Schulz, Assoc. Prof. Dr. Pairote Sattayatham, and Asst. Prof. Dr. Arjuna Chaiyasena.

I have also benefited from my senior and junior friends at Suranaree University of Technology, to whom I would like to extend my sincere thanks for their support.

Finally, I would like to express my deep gratitude to my parents, my brothers and sister for their understanding, patience and moral support during all these years, and also my thanks to my friends, who always give their support and help.

Phaichayon Sirisathienwatthana

Contents

	Page
Abstract in Thai	I
Abstract in English	II
Acknowledgements	III
Contents	IV
List of Tables	VI
Chapter	
I Introduction	1
II Line Search Procedures and Search Directions	7
2.1 Search Directions	8
2.1.1 Steepest Descent Directions	8
2.1.2 Newton Directions	12
2.1.3 Quasi-Newton Directions	14
2.1.4 Conjugate Gradient Directions	20
2.2 Conditions on the Step Lengths	25
2.2.1 Exact Line Search	25
2.2.2 Inexact Line Search	25
III Hybrid Directions	28
3.1 Descent Property	28
3.2 Expanding Subspace Property	29
3.3 Hybrid Directions	35
3.4 Standard Test Problems	37

Contents (Continued)

Chapter **Page**

IV Numerical Results 40

4.1 Implementation of the Hybrid Direction Algorithm 40

4.2 Numerical Results 42

4.3 Discussion 61

V Conclusion 63

References 65

Appendix

Appendix A. Terminology 70

A.1 Types of Solution 70

A.2 Necessary Conditions 70

A.3 Convex Functions 71

A.4 Types of Convergence 72

A.5 Sherman-Morrison-Woodbury formula 73

Appendix B. Fortran Program 74

Curriculum Vitae 89

List of Tables

Table **Page**

4.1 Results for the Variably Dimensioned Function 43

4.2 Results for the Penalty Function I 49

4.3 Results for the Penalty Function II 55

4.4 Results for the Biggs EXP6 Function 57

4.5 Results for the Brown Badly Scaled Function 59

4.6 Results for the Brown and Dennis Function 61

Chapter I

Introduction

The unconstrained minimization problem is considered as one of the important problems in continuous optimization, both in theoretical and application aspects. For the theoretical aspect, the problem involves a wide range of mathematical subjects, from fundamental subject such as advanced calculus, mathematical analysis and linear algebra, to the advanced subjects such as functional analysis, differential geometry and operator theory etc. For the application aspect, the problem always has its place in many practical or real-world problems in various disciplines such as science, engineering, economics, computer graphic design etc. The latter aspect also leads to the continuous development of computational methods for solving the unconstrained minimization problems, as no single all-purpose algorithm can handle a variety of unconstrained minimization problems, in particular those arising from real-world problems.

The class of objective functions in the unconstrained minimization problems considered here will be restricted to the class of continuous differentiable functions on $\mathbb{R}^n$. This restriction makes the unconstrained minimization problem equivalent to solving a system of n equations with n unknowns. Determining a minimizer of an objective function is, in general, not easy, as for the problem of solving a system of nonlinear equations. Since, many factors involve in the iterative process, such as the choices of starting points, the choices of directions for searching for the minimizer, the criteria for determining a step length along the search direction, the complexity in computing the Hessian, in particular, when dealing with problems with high dimensions etc. For these reasons, many methods have been developed and are still being continuously developed to solve the unconstrained minimization problems efficiently. Some well-known and classical methods are the steepest descent method, Newton method, conjugate gradient method and the quasi-Newton methods. Some other methods, such as optimization bisection (OPTBIS) method for imprecise function and gradient values (Vrahatis, M.N., Androulakis, G.S., and Manoussakis, G.E. (1996)) and a dimension-reducing (DROPT) optimization method (Grapsa, T.N. and Vrahatis, M.N. (1996)) have also been recently developed. However, most of the methods share one common task, i.e., how to construct the suitable search directions for locating the minimizer. The efficiency of the method therefore relies very much on the choices of search directions.

Newton's method for unconstrained minimization problems is analogous to the Newton's method for solving nonlinear equations. As this method requires the computation of the Newton direction from the inverse of the Hessian of the objective function. The attractive feature of this method is that it produces a sequence of iterates which converges quadratically to the minimizer if the starting point lies sufficiently close to the minimizer. In this sense, it is a local method. However, the computation of the Hessian in each iteration makes this method less attractive when the dimension of the problem is high. Consequently, some modifications on the Hessian computation or approximations of the Hessian were developed. One approach is to construct the least change secant update for approximating the Hessian. Some well-known updates, such as, symmetric rank-one (SR1) updates, Davidon-Fletcher-Powell (DFP) update, Broyden-Fletcher-Goldfarb-Shanno (BFGS) update were then developed. For details of the development of these updates for Hessian approximations can be found, for example, in

Nocedal, J. and Wright, S. J. (1999), Kelley, C.T. (1999), Luenberger, D.G. (1984) and Dennis, J.E., JR. and Schnabel, R.B. (1983). The method which employs this modified Newton direction is called the variable metric method or quasi-Newton method. The framework of the Newton method is still used for this method. The only difference is that the true Hessian is replaced by the Hessian approximation. Two efficient updates which are usually employed for achieving the Hessian approximation are Davidon-Fletcher-Powell (DFP) and Broyden-Fletcher-Goldfarb-Shanno (BFGS) updates. Both updates preserve the positive definiteness of the inverse Hessian approximation. The quasi-Newton method, using the BFGS update (or DFP update) was proved to produce a q-superlinearly convergent method (Broyden-Dennis-Moré, 1973) under the suitable choices of the starting point and initial Hessian approximation. In 1997, Liao, A. modified the BFGS update and gave the convergence results. Li, D-H. and Fukushima, M. (2001) modified the update for nonconvex minimization and also established the convergence results. For the trust region method, the symmetric rank-one update is found to be more suitable as the Hessian approximation produced is close to the true Hessian. The global convergence of the trust region method was shown by Conn, A.R., Gould, N.I.M. and Toint, Ph.L. (1991) under the suitable conditions. In 1989, Lukšan, L. improved the variable metric methods based on the controlled scaling and on the pertinent combination of the rank one method with other variable metric methods. Recently, a new variable metric method for large scale cases has been introduced by Vlček, J. and Lukšan, L. (2002). Moreover, they suggested some modifications and improvements of reduced-Hessian methods.

The conjugate gradient direction was first proposed by Hestenes and Stiefel in 1950. It was initially investigated for a convex quadratic function. It turned out that for this specific case, a number of interesting theoretical and geometrical results were found, such as, the conjugacy condition, the finite termination property, the expanding subspace minimization, and the Krylov subspace relations. Details and proofs of these properties, can be found in Nocedal, J. and Wright, S. J. (1999), Nazareth, L. (1979) and Buckley, A. (1978). For dealing with a more general class of problems, the conjugate gradient direction was then modified by Fletcher and Reeves in the 1960s. Many methods were then developed based on their ideas and some are widely used in practice. Two well-known methods are Fletcher-Reeves (FR) and Polak-Riebière (PR) methods. The convergence properties of these methods are discussed, for example, in Nocedal, J. and Wright, S. J. (1999), Dai, Y.H., Han, J., Liu, G., Sun, D., Yin, H. and Yuan, Y-X. (1999), Grippo, L. and Lucidi, S. (1997) and Dai, Y.H. and Yuan, Y-X. (2000). In 1977, Powell proposed a restart strategy for the conjugate gradient method to improve the convergence. Recently, the new efficient restart strategy has been introduced by Lukšan, L. (1991). Nonetheless, it tends to be very sensitive to round off error.

The investigations in this thesis will utilize the framework of the line search procedure with different criteria for choosing the scalar or step length along the search direction. The Armijo's rule (Luenberger, D.G. (1984)), backtracking technique (Dennis, J.E., JR. and Schnabel, R.B. (1983)), Wolfe conditions, Strong Wolfe conditions (Nocedal, J. and Wright, S.J. (1999) and Fletcher, R. (1987)), for choosing the step length will be used here. Various search directions such as the steepest descent, quasi-Newton and conjugate gradient directions are used as the search directions. These directions will be also used in such a way that they are combined to produce the hybrid directions. The behaviours and performances of the constructed hybrid directions will be tested on some standard test problems for unconstrained minimization problems from Moré, J.J. *et al.* (1981). The idea of producing such a hybrid direction will serve as the basis for further development especially for parallel computation, as these direction can be produced independently.

The proposed research will therefore focus on two main aspects. The first one is to investigate the theoretical aspects and properties related to the combined directions for solving unconstrained minimization problems within the line search framework. The ideas are based on minimization on a linear variety, i.e., instead of searching along a single direction, the line search is performed along a combined direction so that the minimizer along this combined direction will be as close as possible to the minimizer of the objective function on the linear variety. The combined directions will be constructed based on the quasi-Newton and conjugate gradient directions. The second one is the computational aspect, i.e., to develop a numerical method and implement it for observing the performances and efficiency of the combined directions on the standard test problems from the collection of Moré, J.J., *et al.* (1981). The effects of the choices of the step length based on various criteria will also be tested numerically.

The search directions will be restricted to the quasi-Newton directions based on the BFGS update and the CG directions based on the Polak-Riebière (PR) choice of the scalar and also the steepest descent directions. The numerical investigations will be based on the four combinations of these three directions with various values of the scalar multiples in the linear combinations. These choices only serve as the preliminary directions for investigation.

The investigation proposed here should help establish another approach for solving the unconstrained minimization problems. It is intended here that the proposed method based on the line search along the combined direction will serve as the basis for developing a parallel algorithm which will help speed up the convergence and reduce the number of function evaluations.

The thesis contents consist of five chapters. Chapter I presents the literature survey on the methods for solving the unconstrained minimization problems, emphasizing on the line search framework and well-known search directions. Chapter II presents the theoretical background of the search directions and the line search procedures. Some advantages and disadvantages of the presented search directions are also discussed in Chapter II. Chapter III presents the theoretical properties related to minimization on a linear variety and a numerical algorithm based on the hybrid directions within the line search framework will be proposed. The numerical results and discussion of the performances of the proposed hybrid directions on the standard test problems are given in Chapter IV. Finally, the conclusion is presented in Chapter V, and the FORTRAN program is given in the Appendix.

Chapter II

Line Search Procedures and Search Directions

The problem considered here is an unconstrained minimization of a nonlinear function in n real variables, $f : \mathbb{R}^n \to \mathbb{R}$

$$\min_{x \in \mathbb{R}^n} f(x). \tag{2.1}$$

There are various methods for solving this problem. The line search framework is often used as one of the approaches for solving this problem. The line search framework can be formulated as follows:

Given a starting point x_0, the sequence $\{x_k\}$ generated in the line search framework takes the form

$$x_{k+1} = x_k + \lambda_k d_k, \quad k = 0, 1, 2, \ldots \tag{2.2}$$

where d_k is the search direction which has to be a descent direction at x_k and λ_k is a positive scalar reflecting the step length taken in the d_k direction. There are various strategies in constructing the search directions which involve a lot of theoretical considerations as discussed in Dennis, J.E., JR. and Schnabel, R.B. (1983), Kelley, C.T. (1999) and Nocedal, J. and Wright, S.J.(1999) and also the computational methods. The properly chosen step lengths also play a very important role for successfully locating the minimizer of f as well as the convergence speed of the sequence $\{x_k\}$ to the minimizer of f. Some properties of the search directions and the criteria for choosing the step length are briefly discussed in the following sections.

2.1 Search Directions

2.1.1 Steepest Descent Directions

The basic method for unconstrained optimization is the classical steepest descent (SD) method which makes use of the gradient of f, $\nabla f(x)$, at each iteration. As it is known that the maximum decrease of f from the point x is along the negative of the gradient of f at x. The search direction for this method is then called the steepest descent direction and in each iteration of the line search, the search direction is taken to be

$$d_k^{SD} = -g_k, \tag{2.3}$$

where $g_k = \nabla f(x_k)$.

The convergence analysis of this method is based on the investigation of applying this method to find the minimizer of the convex quadratic function

$$\phi(x) = \frac{1}{2}x^T Q x - b^T x, \tag{2.4}$$

where Q is an $n \times n$ symmetric positive definite matrix. The minimizer for this quadratic case is, in fact, the unique solution, x^*, of the linear system,

$$Qx = b. \tag{2.5}$$

As introduced in Luenberger, D.G. (1984), the error function,

$$E(x) = \frac{1}{2}(x - x^*)^T Q (x - x^*) = \phi(x) + \frac{1}{2}x^{*T} Q x^*, \tag{2.6}$$

is used instead of the initial objective function for analyzing the convergence of the steepest descent method. Using the exact line search, that is, solving for the value of λ_k which minimizes the function of a single variable λ

$$h(\lambda) = \phi(x_k - \lambda g_k), \tag{2.7}$$

where $g_k = \nabla \phi(x_k) = Qx_k - b$. It was shown in Luenberger, D.G. (1984) that at iteration k,

$$E(x_{k+1}) \leq \left(\frac{ev_{\max} - ev_{\min}}{ev_{\max} + ev_{\min}} \right)^2 E(x_k), \tag{2.8}$$

or

$$E(x_{k+1}) \leq \left(\frac{r-1}{r+1} \right)^2 E(x_k), \tag{2.9}$$

where $ev_{\max}$ and $ev_{\min}$ are the largest and smallest eigenvalues of the Hessian of f at x^*, $\nabla^2 f(x^*)$, respectively, and $r = ev_{\max}/ev_{\min}$, is the ratio of the largest to the smallest eigenvalue. The inequality (2.8) shows that from any starting point x_0, the steepest descent method converges to the unique minimizer x^*. However, the rate of convergence depends on the ratio r which will cause slow convergence as this ratio increases when the largest and the smallest eigenvalues are very much different.

For the nonquadratic case, the steepest descent method should be implemented with the inexact line search. The exact line search is not appropriate for computation in this case as it involves an exact one-dimensional minimization problem in each iteration. Therefore, only a suitable scalar λ in (2.7) which guarantees the sufficient decrease of the value of function f in the direction d_k is required. The steepest descent method when applied to a nonquadratic function, with either exact or inexact line search, and under some mild assumptions, can be shown to converge to a local minimizer or saddle point of $f(x)$. That is, if the steepest descent method produces a sequence $\{x_k\}$ converging to a local minimizer x^* where the Hessian $\nabla^2 f(x^*)$ is positive definite, and $ev_{\max}$ and $ev_{\min}$ are the largest and smallest eigenvalues of $\nabla^2 f(x^*)$, then it can be shown that $\{x_k\}$ satisfies

$$\lim_{k \to \infty} \sup \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} \leq c, \qquad c = \frac{ev_{\max} - ev_{\min}}{ev_{\max} + ev_{\min}}. \tag{2.10}$$

As in the quadratic case, the convergence is linear and the convergence rate depends on the largest and smallest eigenvalues of the Hessian of f at the minimizer x^*.

In 1966, Armijo modified the steepest descent method by proposing the scheme for adapting the step length λ_k in (2.2) and also gave the convergence result as stated in the following theorem.

Theorem 2.1.

Suppose that the objective function $f : \mathbb{R}^n \to \mathbb{R}$ satisfies the following conditions:

1. *f is continuous and bounded below on $\mathbb{R}^n$,*
2. *For a given $x_0 \in \mathbb{R}^n$, f is continuously differentiable on the bounded level set $\mathcal{L}(x_0) = \{x : f(x) \leq f(x_0)\}$,*
3. *f has a unique minimizer $x^* \in \mathbb{R}^n$,*
4. *$\nabla f(x) = 0$ is satisfied for $x \in \mathcal{L}(x_0)$ if and only if $x = x^*$,*
5. *∇f is Lipschitz continuous on $\mathcal{L}(x_0)$.*

Let $\lambda_m = \lambda_0/2^{m-1}$, $m = 1, 2, \ldots$, where λ_0 is any assigned positive number. Then the sequence $\{x_k\}$ generated by

$$x_{k+1} = x_k + \lambda_{m_k} d_k^{SD}, \quad k = 1, 2, \ldots, \tag{2.11}$$

where $\lambda_{m_k} = \lambda_0/2^{m_k-1}$ and m_k is the smallest positive integer for which

$$f(x_k + \lambda_{m_k} d_k^{SD}) \leq f(x_k) + \frac{1}{2}\lambda_{m_k} \nabla f(x_k)^T d_k^{SD}, \tag{2.12}$$

converges to the minimizer x^* of f.

The advantages of the steepest descent method under the line search framework can be described as follows:

1. The computation of the search direction in each iteration is simple, since only the gradient of the function at the current iterate is required, i.e., $d_k = -\nabla f(x_k)$.

2. The search direction being descent can always be assured since $d_k = -\nabla f(x_k)$ satisfies the following condition:

$$\nabla f(x_k)^T d_k < 0. \tag{2.13}$$

3. The method is a global method if the scalars λ chosen along the steepest descent directions satisfy

$$f(x_k + \lambda d_k) \leq f(x_k) + \alpha\lambda\nabla f(x_k)^T d_k \tag{2.14a}$$

or

$$f(x_{k+1}) \leq f(x_k) + \alpha\nabla f(x_k)^T(x_{k+1} - x_k), \tag{2.14b}$$

for some fixed constant $\alpha \in (0, 1)$, and

$$\nabla f(x_{k+1})^T d_k = \nabla f(x_k + \lambda d_k)^T d_k \geq \beta\nabla f(x_k)^T d_k \tag{2.15a}$$

or

$$\nabla f(x_{k+1})^T(x_{k+1} - x_k) \geq \beta\nabla f(x_k)^T(x_{k+1} - x_k), \tag{2.15b}$$

for some fixed constant $\beta \in (\alpha, 1)$.

The above two conditions (2.14) and (2.15) are known as Armijo and Goldstein's conditions.

Some disvantages of the steepest descent method are discussed for example in Vrahatis, M.N., *et al.* (2000). They are as follows:

1. Each iteration is calculated independently of the others, that is, no information is stored and used to help accelerate convergence.

2. It is not generally a finite procedure for minimizing a convex function.
3. The rate of convergence depends strongly on the morphology of the objective function.

2.1.2 Newton Directions

The idea for constructing the Newton direction is based on the following local quadratic model of f about the current iterate x_k,

$$m_k(x_k + d) = f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T \nabla^2 f(x_k) d. \tag{2.16}$$

The minimizer of this model (2.16) is the point $x_k + d_k$, where $\nabla m_k(x_k + d_k) = 0$, or d_k satisfies,

$$\nabla^2 f(x_k) d = -\nabla f(x_k). \tag{2.17}$$

The search direction is then called the Newton (N) direction. Denote this direction by d_k^N, and in each iteration d_k^N is given by

$$d_k^N = -\left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k). \tag{2.18}$$

As for solving the system of nonlinear equations, it is required that the Hessian in (2.18) has to nonsingular and moreover positive definite for this problem for d_k to be a descent direction. The corresponding step length λ_k in (2.2) for the Newton direction is, in general, taken to be 1. Since this will help capture the fast local quadratic convergence when the iterates get closer to the minimizer of f as in the case of the Newton direction when applied to find the roots of the nonlinear functions.

However, the Newton direction has some restrictions. Specifically, if the starting point is too far from a minimizer, the Hessian, $\nabla^2 f(x_k)$, may not be positive definite and the local quadratic model will not have a local minimizer, and the local linear model of $\nabla f(x_k)$, will have a root which corresponds to the local maxima or saddle point of m_k. For these reasons, some modifications were introduced. As discussed in Dennis, J.E., JR. and Schnabel, R.B. (1983) and Luenberger, D.G. (1984), the Hessian $\nabla^2 f(x_k)$ is modified by taking

$$H_k = \nabla^2 f(x_k) + \epsilon_k I, \tag{2.19}$$

for some positive constant ϵ_k that makes H_k positive definite. Discussions and references on how to obtain ϵ_k are provided in Dennis, J.E., JR. and Schnabel, R.B. (1983) and Luenberger, D.G. (1984). So, the modified Newton directions provides the estimates of the minimizer as

$$x_{k+1} = x_k - \lambda_k H_k^{-1} \nabla f(x_k), \tag{2.20}$$

where the scalar λ_k has to satisfy conditions such as Armijo's conditions, Goldstien's conditions or other conditions to be discussed later. However, λ_k should be close to 1 when the iterates are close to the minimizer x^*, where $\nabla^2 f(x^*)$ is positive definite and ϵ_k in (2.19) is close to zero.

Next, the statements of theorem on the convergence of the Newton method are given. Details of the proof can be found in Dennis, J.E., JR. and Schnabel, R.B. (1983), Nocedal, J. and Wright, S.J. (1999) and Kelley, C.T. (1999).

Theorem 2.2.

Let f be twice Lipschitz continuously differentiable on $D \subset \mathbb{R}^n$, that is, there exists a constant $\gamma > 0$ such that

$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq \gamma \|x - y\|, \quad \textit{for all } x, y \in D.$$

Suppose further that $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite. Then there is a $\delta > 0$ such that if $x_0 \in N_\delta(x^*)$, the point generated by Newton direction

$$x_{k+1} = x_k - \left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k),$$

converges q-quadratically to x^*.

Some advantages and disadvantages of the Newton method can be summarized as follows:

Advantages

1. The method generates the sequence which converges q-quadratically to the minimizer if the objective function and the starting point satisfy the conditions in Theorem 2.2.
2. The minimizer is found in one iteration if the objective function is strictly convex.

Disdvantages

1. The Newton method is a local method.
2. The full Hessian has to be calculated in each iteration.
3. Solving a system of linear equations is required in each iteration and the Hessian matrix may be ill-conditioned.

2.1.3 Quasi-Newton Directions

An alternative for the Newton direction is developed based on approximating the Hessian in equation (2.17) by an $n \times n$ nonsingular matrix B_k. The search direction then takes the following form,

$$d_k^{QN} = -B_k^{-1} \nabla f(x_k), \tag{2.21}$$

and is called the quasi-Newton (QN) direction. Theoretical considerations and development have lead to various forms of B_k in equation (2.21). The main condition is to require that B_{k+1} satisfies the multidimensional secant equation,

$$B_{k+1} s_k = y_k, \tag{2.22}$$

where $s_k = x_{k+1} - x_k$, and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$. Equation (2.22) describes an underdetermined system. Therefore, other conditions need to be imposed in order to determine B_{k+1} uniquely. These conditions are discussed for example in Nocedal, J. and Wright, S.J. (1999) and they lead to the so-called quasi-Newton updates. Some important and widely used quasi-Newton updates are discussed in the following.

Rank One Update

A well known rank one update is the Broyden's update or secant update which was proposed by Broyden, C. in 1965. It is mainly used for replacing the Newton's direction for solving a nonlinear system.

$$F(x) = 0, \tag{2.23}$$

where $F : \mathbb{R}^n \to \mathbb{R}^n$ and $x \in \mathbb{R}^n$. The major difference is that the Jacobian needs not be computed in this approach. That is, the major ideas of this update is an attempt based on the approximation of the Jacobian, $J(x)$, by using the old data and old Jacobian approximation in the previous iteration. The method for solving nonlinear system (2.23) based on the Broyden's update, or the Broyden's method, generates the sequence $\{x_k\}$ of the estimates of the root in (2.23) of the form

$$x_{k+1} = x_k - A_k^{-1} F(x_k). \tag{2.24}$$

The Broyden's update of A_k for the next iteration is given by

$$A_{k+1} = A_k + \frac{\left(y_k - A_k s_k\right) s_k^T}{s_k^T s_k}, \tag{2.25}$$

where $s_k = x_{k+1} - x_k$ and $y_k = F(x_{k+1}) - F(x_k)$. The update A_{k+1} satisfies the secant equation (2.22).

The objective of Broyden's update is to save the amount of the computation of Jacobian matrix required for the Newton direction. However, it is a local method which produces the iterates converging superlinearly to the solution x^* when the starting point x_0 is sufficiently close to x^*, where $J(x^*)$ is nonsingular and A_0 is also sufficiently closed to $J(x_0)$. In practice, the finite difference is used for obtaining are initial Jacobian approximation, A_0. The following theorem gives the convergence results of the Broyden's method.

Theorem 2.3. (Dennis-Moré, 1974)

Let $D \subset \mathbb{R}^n$ be an open convex set $F : \mathbb{R}^n \to \mathbb{R}^n$, $J(x_k) \in \mathrm{Lip}_\gamma(D)$, $x^* \in D$ and $J(x^*)$ nonsingular. Let $\{A_k\}$ be a sequence of nonsingular matrices in $\mathbb{R}^{n\times n}$, and suppose for some $x_0 \in D$ that the sequence of points generated by (2.24) *remain in D, and satisfies $x_k \neq x^*$ for any k, and $\lim_{k\to\infty} x_k = x^*$. Then the sequence $\{x_k\}$ converges q–superlinearly to x^* in some norm $\|\cdot\|$, and $F(x^*) = 0$, if and only if*

$$\lim_{k\to\infty} \frac{\|\big(A_k - J(x^*)\big)s_k\|}{\|s_k\|} = 0, \tag{2.26}$$

where $s_k = x_{k+1} - x_k$.

For the application to the minimization problems, this update is not suitable because the update formula in (2.25) does not preserve the positive definiteness, that is, A_{k+1} may not be positive definite even if A_k is positive definite.

Rank Two Updates

Some important and popular rank two updates for quasi-Newton methods for unconstrained minimization problems are presented in the following.

(1) DFP Update

In 1959, Davidon, Wm.C. proposed a rank two update for solving unconstrained minimization problems and due to Fletcher, R. and Powell, M.J.D. that the update become popular. The Davidon-Fletcher-Powell (DFP) update for the unconstrained minimization problems has the following form,

$$B_{k+1} = \left(I - \gamma_k y_k s_k^T\right) B_k \left(I - \gamma_k s_k y_k^T\right) + \gamma_k y_k y_k^T, \tag{2.27}$$

with

$$\gamma_k = \frac{1}{y_k^T s_k}.$$

Equation (2.27) shows that B_k is updated in each iteration to get the new approximation to the Hessian, B_{k+1}. However, to save the amount of computations and to avoid the factorization B_k in each iteration, the inverse form of B_{k+1}, denoted by H_{k+1}, can be obtained by using the Sherman-Morrison-Woodbury formula (A.1) as

$$H_{k+1} = H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \frac{s_k s_k^T}{y_k^T s_k}, \tag{2.28}$$

where H_k and H_{k+1} denote the inverse of B_k and B_{k+1} respectively. Equation (2.28) shows that the inverse H_k is updated to get H_{k+1}. The search direction is then directly given by

$$d_k^{DFP} = -H_k \nabla f(x_k). \tag{2.29}$$

The DFP update is considered to be quite effective but it was soon replaced by the BFGS update, which is considered to be the most effective quasi-Newton updates.

(2) BFGS Update

The quasi-Newton update which is widely used in unconstrained optimization problems is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update. As it exhibits the desirable property, that is, preserving positive definiteness.

The inverse update formula of the approximation of the inverse Hessian is

$$H_{k+1} = \left(I - \rho_k s_k y_k^T\right) H_k \left(I - \rho_k y_k s_k^T\right) + \rho_k s_k s_k^T, \tag{2.30}$$

with

$$\rho_k = \frac{1}{y_k^T s_k},$$

and the search direction is therefore given by

$$d_k^{BFGS} = -H_k \nabla f(x_k). \tag{2.31}$$

For this update, the curvature condition

$$y_k^T s_k > 0, \tag{2.32}$$

has to be satisfied in each iteration in order to preserve the positive definiteness, given that the initial approximation, H_0, is symmetric positive definite.

Using the Sherman-Morrison-Woodbury formula (A.2), Equation (2.30) can be transformed into

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}, \tag{2.33}$$

which directly updates the Hessian approximation. For the convergence results related to this update, Dennis, JE.JR. and Moré, J.J. (1974) gave the following theorem.

Theorem 2.4.

If the function f is twice continuously differentiable in an open convex set D, and assume that $\nabla^2 f \in Lip_\gamma(D)$. Consider a sequence $\{x_k\}$ generated by (2.2), where $\nabla f(x_k)^T d_k < 0$ for all k and λ_k is chosen to satisfy (2.52a) *with an $\alpha < \frac{1}{2}$, and* (2.52b)*. If $\{x_k\}$ converges to a point $x^* \in D$ at which $\nabla^2 f(x^*)$ is positive definite, and if*

$$\lim_{k\to\infty} \frac{\|\nabla f(x_k) + \nabla^2 f(x_k) d_k\|}{\|d_k\|} = 0, \tag{2.34}$$

then there is an index $k_0 \geq 0$ such that for all $k \geq k_0$, $\lambda_k = 1$ is admissible. Furthermore, $\nabla f(x^*) = 0$, and if $\lambda_k = 1$ for all $k \geq k_0$, then $\{x_k\}$ converges q-superlinearly to x^*.

In 1987, Byrd, R.H., Nocedal, J. and Yuan, Y-X. also gave the global convergence results of a class of quasi-Newton methods on convex problems. The interesting and efficient computational implementation of the quasi-Newton methods based on a parallel algorithm design using the BFGS update was also presented in Caprioli, P. and Holmes, M.H. (1998), and Chen, Z., Fei, P. and Zheng, H. (1995).

(3) SR1 Update

The other important update is the symmetric rank one (SR1) update which has the following form

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}. \tag{2.35}$$

By applying the Sherman-Morrison-Woodbury formula (A.1), the corresponding update formula for the inverse Hessian approximation, H_k can be obtained as follows

$$H_{k+1} = H_k + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^T}{(s_k - H_k y_k)^T y_k}. \tag{2.36}$$

It can be seen that even if B_k is positive definite, B_{k+1} may not have this property; the same is true for H_{k+1}. The matrices generated by the SR1 update formula tend to be very good approximations of the true Hessian. In Conn, A.R., Gould, N.I.M. and Toint, Ph.L. (1991), the convergence of the sequence of the matrices generated by the SR1 update was shown under suitable conditions. They also pointed that by maintaining the positive definiteness of the update as in the case of the BFGS update can cause some drawbacks. The first one is that the true Hessian at points far away from the minimizer may not be positive definite and therefore maintaining positive definiteness of the Hessian approximations is not appropriate and the concept of the Hessian approximation has to be revised. The second one is that the true Hessian may be indefinite at the solution which lies in a feasible region defined by bounds of variables or is subject to more general constraints. However, the SR1 update has to implemented within a setting different from the line search framework. The trust region which is another practical approach for solving an unconstrained minimization problem provides the right setting for implementing the SR1 update.

2.1.4 Conjugate Gradient Directions

The development of the conjugate gradient direction is based on solving the convex quadratic problem. Later it is modified to cover a more general class of unconstrained minimization problems, in particular, it works well for large scale and smooth problems.

For the convex quadratic problem, that is, Q is positive definite in (2.4), the search directions d_i are required to satisfy the *conjugacy condition*

$$d_i^T Q d_j = 0, \quad \text{for all } i \neq j. \tag{2.37}$$

For the formulation of conjugate gradient direction, the gradient of ϕ in (2.4) is refered to as the residual of the linear system, that is

$$\nabla \phi(x) = Qx - b = r(x). \tag{2.38}$$

The first direction for solving (2.4) is the steepest descent direction,

$$d_0 = -r_0 = -\nabla f(x_0), \tag{2.39}$$

where x_0 is any starting point in $\mathbb{R}^n$. The iterates then taken the form

$$x_{k+1} = x_k + \alpha_k d_k, \tag{2.40}$$

where α_k is the scalar which solves the exact minimization of $\phi(x)$ along $x_k + \alpha d_k$. The directions d_k, for $k \geq 1$, have the form

$$d_k = -r_k + \beta_k d_{k-1}, \tag{2.41}$$

where $r_k = \nabla\phi(x_k)$ and

$$\beta_k = \frac{r_k^T r_k}{r_{k-1}^T r_{k-1}}. \tag{2.42}$$

The direction d_k in (2.41) is refered to as the linear conjugate gradient direction. The end result is that the minimizer x^* of ϕ is obtained in n iterations, that is

$$x^* = x_0 + \alpha_0 d_0 + \alpha_1 d_1 + \cdots + \alpha_{n-1} d_{n-1}. \tag{2.43}$$

This is the so-called, *finite termination property.*

Next, some interesting theoretical results are reviewed without the proof, the finite termination property, the conjugacy condition, the subspace relations and the Krylov subspace relations etc. The detailed discussions can be found in Luenberger, D.G. (1984) and Nocedal, J. and Wright, S.J. (1999).

Definition 2.1. (*Conjugacy Condition*)

A set of nonzero vectors $\{d_0, d_1, \ldots, d_k\}$ *is said to be conjugate with respect to the symmetric positive definite matrix* Q *if*

$$d_i^T Q d_j = 0, \quad for\ all\ \ i \neq j. \tag{2.44}$$

From this result, it follows that for any set of nonzero vectors which satisfies (2.44) then these vectors are linearly independent.

Theorem 2.5. (*Finite Termination Property*)

For any $x_0 \in \mathbb{R}^n$ *the sequence* $\{x_k\}$ *generated by* (2.40) *converges to the solution* x^* *in at most* n *steps.*

Theorem 2.6. (*Expanding Subspace Minimization*)

Let $x_0 \in \mathbb{R}^n$ be any starting point and suppose that the sequence $\{x_k\}$ is generated by (2.40). *Then*

$$r_k^T d_i = 0, \quad for \;\; i = 0, 1, 2, \dots, k-1,$$

and x_k is the minimizer of (2.4) *over the linear variety*

$$\{x \big| x = x_0 + span\{d_0, d_1, \dots, d_{k-1}\}\}.$$

Theorem 2.7. (*Krylov Subspace Relations*)

Suppose that the k–th iterate generated by the linear conjugate gradient direction is not the solution point x^*. The following four properties hold:

(1) $r_k^T r_i = 0, \quad for \;\; i = 0, 1, 2, \dots, k-1,$

(2) $span\{r_0, r_1, \dots, r_k\} = span\{r_0, Qr_0, Q^2 r_0, \dots, Q^k r_0\},$

(3) $span\{d_0, d_1, \dots, d_k\} = span\{r_0, Qr_0, Q^2 r_0, \dots, Q^k r_0\},$

(4) $d_k^T Q d_i = 0, \quad for \;\; i = 0, 1, , 2, \dots, k-1.$

Therefore, the sequence x_k converges to x^* in at most n steps.

Simple geometrical interpretation of the linear conjugate gradient direction is also given in Nocedal, J. and Wright, S.J.(1999). If the matrix Q in (2.4) is diagonal, the contours of the functions $\phi(x)$ are elliptical with axes parallel to the coordinate axes. The linear conjugate gradient directions are simply the coordinate directions and therefore the one-dimensional minimization in each iteration is carried out along the coordinate direction. If Q is not diagonal, Q can be transformed into a diagonal matrix and the one-dimensional minimization occurs in the transformed coordinate directions. The finite termination can then be achieved.

The amount of computation in each iteration of the conjugate gradient direction is not greater than n^2, because there is one computation of the matrix and vector product, Qd_k, two calculations of the dot product, $d_k^T Q d_k$ and $r_k^T r_k$, and three vector sums. In fact, the linear conjugate gradient method is equivalent to the Gaussian elimination for solving the linear system.

In general cases, the form of linear conjugate gradient directions can still serve as the form of the search direction with some modifications on the scalar β_k in (2.41). The general forms of the conjugate gradient directions are as follows,

$$d_k^{CG} = -\nabla f(x_k) + \beta_k d_{k-1}, \tag{2.45}$$

where β_k is a scalar subject to various choices due to Fletcher-Reeves (FR), Polak-Ribière (PR) and Hestenes-Stiefel (HS). These choices are

$$\begin{aligned} \beta_k^{FR} &= \frac{\nabla f(x_k)^T \nabla f(x_k)}{\nabla f(x_{k-1})^T \nabla f(x_{k-1})}, \\ \beta_k^{PR} &= \frac{\nabla f(x_k)^T \big(\nabla f(x_k) - \nabla f(x_{k-1})\big)}{\nabla f(x_{k-1})^T \nabla f(x_{k-1})}, \\ \beta_k^{HS} &= \frac{\nabla f(x_k)^T \big(\nabla f(x_k) - \nabla f(x_{k-1})\big)}{\big(\nabla f(x_k) - \nabla f(x_{k-1})\big)^T d_{k-1}}. \end{aligned} \tag{2.46}$$

However, all of β_k's in the above formulas coincide in the case where the objective function is convex quadratic and the line search is exact. The performance of conjugate gradient directions in (2.45) depend on the these choices. The PR choice, as discussed in Nocedal, J. and Wright, S.J. (1999) gives better performance than the FR choice and not significantly different when compared with the HS choice.

In a large scale problem, it may be necessary to refresh the information as the bad effects may accumulate. The restart is therefore recommended and the simple choice is to restart by the steepest descent direction. That is, the search direction (2.45) is replaced by the negative of the gradient at the current iterate. The condition used to test when the restart is necessary is

$$\frac{\left|\nabla f(x_k)^T \nabla f(x_{k-1})\right|}{\nabla f(x_k)^T \nabla f(x_k)} \geq \mu, \tag{2.47}$$

where μ is usually taken to be 0.1. The inequality (2.47) simply tries to detect when the two consecutive gradients tend not to be orthogonal.

Another choice of restart was also proposed by Buckley, A.G. (1978). That is, instead of the steepest descent direction, the BFGS update was used to generate the search direction. The relationship between the BFGS and CG directions was also investigated in Buckley, A.G. (1978) and Nazareth, L. (1979).

The convergence results for general nonlinear objective functions of the conjugate gradient method with the FR choice are given in the following theorem (Detailed proof can be found in Nocedal, J. and Wright, S.J.(1999)).

Assumption 2.1.

1. *The level set* $\mathcal{L} = \{x \big| f(x) \leq f(x_0)\}$ *is bounded.*
2. *In some neighbourhood* $\mathcal{N}$ of $\mathcal{L}$, *the objective* f *is Lipschitz continuously differentiable, that is, there exists a constant* $L > 0$ *such that*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|, \quad \textit{for all } x, y \in \mathcal{N}. \tag{2.48}$$

Theorem 2.8.

Suppose that Assumption 2.1 *holds, and that the sequence of iterates* $\{x_k\}$ *is generated by conjugate gradient directions and the Fletcher-Reeves formula is implemented with a line search that satisfies strong Wolfe conditions* (2.53). *Then*

$$\lim_{k\to\infty} \inf \|\nabla f(x_k)\| = 0.$$

The above theorem can be applied to the conjugate gradient method with the PR choice under the assumption that the function f is strongly convex and the line search is exact.

2.2 Conditions on the Step Lengths

There are two main approaches in selecting a step length along the search directions in the line search framework.

2.2.1 Exact Line Search

The exact line search is to choose $\lambda_k > 0$ which solves the exact minimization of $f(x)$ along $x_k + \lambda d_k$. That is, find $\lambda > 0$ which solves the following problem,

$$\min_{\lambda>0} f(x_k + \lambda d_k). \tag{2.49}$$

2.2.2 Inexact Line Search

There are important conditions which are widely used in practical implementation for selecting the step length along the search direction.

(1) The Armijo's Conditions

The Armijo's conditions require λ to satisfy

$$f(x_k + \lambda d_k) \leq f(x_k) + \xi \lambda \nabla f(x_k)^T d_k, \tag{2.50}$$

where $0 < \xi < 1$. This inequality guarantees that λ is not too large and the next condition is to ensure that λ is not be too small. That is, choose λ to satisfy

$$f(x_k + \eta \lambda d_k) > f(x_k) + \xi \lambda \nabla f(x_k)^T d_k, \tag{2.51}$$

where η is a positive integer. In practice, $\eta = 2$ or $\eta = 10$ and $\xi = 0.2$ are usually used (Luenberger, D.G. (1984)).

(2) The Wolfe Conditions

The Wolfe conditions are known as the sufficient decrease condition and ensure that the step length is not small along the search direction. The Wolfe conditions require the step length λ along the search direction d_k to satisfy

$$f(x_k + \lambda d_k) \leq f(x_k) + \alpha\lambda\nabla f(x_k)^T d_k, \tag{2.52a}$$

$$\nabla f(x_k + \lambda d_k)^T d_k \geq \beta\nabla f(x_k)^T d_k, \tag{2.52b}$$

where $0 < \alpha < \beta < 1$. The Wolfe conditions are used in most line search procedures, and are particularly important when implemented with the quasi-Newton search directions.

(3) The Strong Wolfe Conditions

The strong Wolfe conditions require λ to lie in a broad neighbourhood of a local minimizer or stationary point of $f(x_k + \lambda d_k)$. The step length λ under the strong Wolfe conditions has to satisfy

$$f(x_k + \lambda d_k) \leq f(x_k) + \alpha\lambda\nabla f(x_k)^T d_k, \tag{2.53a}$$

$$\left|\nabla f(x_k + \lambda d_k)^T d_k\right| \leq \beta\left|\nabla f(x_k)^T d_k\right|, \tag{2.53b}$$

where $0 < \alpha < \beta < 1$. The only difference with Wolfe conditions is that the derivative $f'(\lambda_k) = \nabla f(x_k + \lambda_k d_k)^T d_k$ is not allowed to be too positive. The strong Wolfe conditions is used in the implementation with the conjugate gradient directions.

In practice, α is chosen to be 10^{-4}, β is chosen to be 0.9 when the search directions d_k are Newton or quasi-Newton directions, and 0.1 when d_k is a nonlinear conjugate gradient direction.

(4) The Goldstein Conditions

The Goldstein conditions are similar to the Wolfe conditions. They guarantee that the step length λ provides sufficient decrease and λ is not too small.

The Goldstein conditions can be stated as the following pair of inequalities,

$$f(x_k) + (1-c)\lambda_k \nabla f(x_k)^T d_k \leq f(x_k + \lambda_k d_k) \leq f(x_k) + c\lambda_k \nabla f(x_k)^T d_k, \quad (2.54)$$

where $0 < c < \frac{1}{2}$.

However, the first inequality in (2.54) may exclude all minimizers of $f(x_k + \lambda d_k)$. The Goldstein conditions are often used with the Newton directions, but not quite suitable for the quasi-Newton directions which are obtained from the positive definite Hessian approximation.

(5) Backtracking Techniques

The backtracking technique is an approach to choose the suitable step length so that the sufficient decrease (2.52a) condition is satisfied but the reasonable progression of the step length (2.52b) is not directly implemented. The general form of the backtracking technique is as follows:

$$\begin{aligned}
&\text{Choose} \quad \lambda_k > 0,\ \rho, \alpha \in (0,1) \\
&\text{While } f(x_k + \lambda_k d_k) > f(x_k) + \alpha \lambda_k \nabla f(x)^T d_k, \text{do} \\
&\qquad \lambda_k = \rho \lambda_k; \qquad\qquad (2.55) \\
&\text{Set } x_{k+1} = x_k + \lambda_k d_k;
\end{aligned}$$

A strategy was given in Dennis, J.E., JR. and Schnabel, R.B. (1983) for setting the new step length or the backtracking in (2.55). If the sufficient decrease condition is not satisfied, then the quadratic fit is used and if necessary the cubic fit, by using the interpolation of the function values and gradients available in the iteration step. Usually, the value λ_k is first assigned to be 1 and if after the first backtrack, λ_k is too small, i.e., λ_k is less than 0.1 (see Dennis, J.E., JR. and Schnabel, R.B. (1983)), then λ_k is taken to be 0.1.

Chapter III

Hybrid Directions

In this chapter, some combinations of the search directions mentioned in Chapter II are taken as the new search directions for the line search procedures with the usual conditions for selection the scalars along these new combined directions. However, the condition for checking the search direction whether it is descent or not will be maintained throughout the implementation. If the descent condition is not satisfied, then a restart with the steepest descent direction will be used.

First, some theoretical considerations of the combined directions are given.

3.1 Descent Property

Consider the unconstrained minimization problem

$$\min_{x \in \mathbb{R}^n} f(x), \tag{3.1}$$

where f is twice continuously differentiable on $\mathbb{R}^n$. Let $d_0, d_1, \ldots, d_k \in \mathbb{R}^n$, $k \leq n$, be the search directions at some location x_c in $\mathbb{R}^n$. Suppose that each d_i $(1 \leq i \leq k)$ is a descent direction of f at x_c. That is,

$$\nabla f(x_c)^T d_i < 0, \tag{3.2}$$

for $i = 1, 2, \ldots, k$. By taking a linear combination of these directions with positive scalars, the resulting search direction is also a descent direction of f at

x_c, or the combined direction satisfies

$$\nabla f(x_c)^T(\alpha_1 d_1 + \alpha_2 d_2 + \cdots + \alpha_k d_k) < 0, \tag{3.3}$$

for any positive scalar $\alpha_1, \alpha_2, \ldots, \alpha_k$.

3.2 Expanding Subspace Property

It is worth to stress the important properties of the expanding subspace property which is related to the conjugate directions for the following convex quadratic problem,

$$f(x) = \frac{1}{2}x^T Qx - b^T x + c, \tag{3.4}$$

where $Q \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix, b is a fixed vector in $\mathbb{R}^n$ and c is a real number. It is well-known that for any given set of nonzero directions $\{d_0, d_1, \ldots, d_{k-1}\}$ which satisfy the conjugacy condition

$$d_i^T Q d_j = 0, \quad \text{for } i \neq j \tag{3.5}$$

and for any $x_0 \in \mathbb{R}^n$, the sequence $\{x_k\}$ defined by

$$x_k = x_{k-1} + \alpha_{k-1} d_{k-1}, \tag{3.6}$$

where α_{k-1} is the minimizer of $f(x_{k-1} + \alpha d_{k-1})$. Then the minimizer of f is found in at most n iterations. Moreover, x_k is the minimizer of f over the linear variety $x_0 + V_k$, where V_k is the subspace spanned by $d_0, d_1, \ldots, d_{k-1}$. That is, the line minimizer x_k of $f(x)$ along $x_{k-1} + \alpha d_{k-1}$ is the global minimizer of $f(x)$ over $x_0 + V_k$. This can be expressed as

$$\min_{x \in x_0 + v_k} f(x) = \min_{\alpha} f(x_{k-1} + \alpha d_{k-1}). \tag{3.7}$$

The expanding subspace properly based on the conjugate directions minimizing (3.4) motivates the idea of the possibility of combining a collection of search directions and solve for a minimizer of a more general class of problems.

Let $\{d_0, d_1, \ldots, d_{k-1}\}$ be a collection of search directions in $\mathbb{R}^n$ for solving (3.1). The approach for the investigation is based on taking a linear combination

$$v_k = \beta_0 d_0 + \beta_1 d_1 + \cdots + \beta_k d_{k-1}, \tag{3.8}$$

where $\beta_0, \beta_1, \ldots, \beta_k$ are some scalars. Then find an estimate of the minimizer of $f(x)$ along v_k. That is, the estimate of the minimizer takes the similar form as given in (3.6), with d_k replaced by v_k,

$$x_{k+1} = x_k + \alpha_k v_k. \tag{3.9}$$

The approach can now be outlined in the general form as follows:

Algorithm 3.1.

Given $f : \mathbb{R}^n \to \mathbb{R},\ f \in C^1$ and a starting point $x_0 \in \mathbb{R}^n$.

At iteration j, $j = 0, 1, 2, \ldots$.

***Step** A. Generate a set of linearly independent search directions*

$$d_0^j,\ d_1^j,\ \ldots,\ d_{k-1}^j,$$

where $k \leq n$.

(The superscript denotes the iteration number and the subscript denotes the search direction number.)

***Step** B. Take a linear combination of the directions from Step A. Set*

$$v^j = \beta_0 d_0^j + \beta_1 d_1^j + \cdots + \beta_{k-1} d_{k-1}^j.$$

Step *C. Perform the line search from* x_j *along* v^j *to obtain the admissible scalar* λ_j *and set the new iterate as*

$$x_{j+1} = x_j + \lambda_j v^j.$$

Step *D. Test the admissibility of* x_{j+1}*. If* x_{j+1} *is admissible then stop, else go back to Step A.*

Some behaviours or properties of the combined directions in Step A. can be investigated based on the problem (3.4). First, some definitions and theorems necessary for the development in this section are reviewed (Luenberger, D.G. (1984)).

Definition 3.1. *Let* $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$*. For any given* $x \in \Omega$*, a vector* d *is a feasible direction at* x *if there is an* $\bar{\alpha} > 0$ *such that* $x + \alpha d \in \Omega$ *for all* $\alpha, 0 \leq \alpha \leq \bar{\alpha}$*.*

Theorem 3.1. *(First-order necessary condition)*

Let $f : \Omega \subset \mathbb{R}^n \to \mathbb{R}$ *and let* $f \in C^1$ *on* Ω*. If* x^* *is a relative minimum point of* f *over* Ω*, then for any* $d \in \mathbb{R}^n$ *which is a feasible direction at* x^**, it follows that*

$$\nabla f(x^*)^T d \geq 0.$$

If follows from Theorem 3.1 that if x^* is the interior point of Ω then

$$\nabla f(x^*) = 0.$$

Theorem 3.2.

Let f *be given as in* (3.4) *and let* $d_0, d_1, \ldots, d_{k-1}$ *be a sequence of nonzero vectors in* $\mathbb{R}^n$ *which satisfy the conjugacy condition in* (3.5)*. Then for any* $x_0 \in \mathbb{R}^n$ *the sequence* $\{x_k\}$ *generated by*

$$x_{k+1} = x_k + \alpha_k d_k$$

where α_k is the minimizer of f along the line $x_k + \alpha d_k$, has the property that

$$f(x_k) = \min_{x \in x_0 + V_k} f(x) = \min_{\alpha \in \mathbb{R}^n} f(x_{k-1} + \alpha d_{k-1}), \tag{3.10}$$

where $x_0 + V_k$ is the linear variety with $V_k = span\{d_0, d_1, \ldots, d_{k-1}\}$.

Proof. See Luenberger, D.G. (1984). □

The idea of the expanding subspace theorem can be extended to investigate the combined directions.

Theorem 3.3.

Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable and convex on $\mathbb{R}^n$. Let $d_0, d_1, \ldots, d_{k-1}$ be a set of nonzero vectors in $\mathbb{R}^n$ which are linearly independent, and let V_k be the subspace spanned by these vectors. Therefore,

$$f(x_k) = \min_{x \in x_0 + V_k} f(x) \tag{3.11}$$

if and ony if $\nabla f(x_k)$ is orthogonal to V_k.

Proof. First suppose that x_k minimizes f over the linear variety $x_0 + V_k$. Since for any x in $x_0 + V_k$, both $x_k - x$ and $x - x_k$ are in V_k and they are feasible directions at x_k. Therefore by the necessary condition in Theorem 3.1,

$$\nabla f(x_k)^T (x_k - x) \geq 0$$

and

$$\nabla f(x_k)^T (x - x_k) \geq 0.$$

If follows that

$$\nabla f(x_k)^T (x_k - x) = 0$$

which implies that

$$\nabla f(x_k) \perp V_k.$$

For proving the "only if "part, by the convexity of f, and for any $x \in x_0 + V_k$,

$$f(x) - f(x_k) \geq \nabla f(x_k)^T (x - x_k).$$

Since $x - x_k \in V_k$, and $\nabla f(x_k) \perp V_k$,

$$\nabla f(x_k)^T (x - x_k) = 0.$$

Therefore, $f(x) \geq f(x_k)$ which proves the "only if "part. □

The ideas on searching for the minimizer on a larger region are motivated by the expanding subspace property based on the conjugate directions and Theorem 3.3. Instead of performing a line search along one single direction in each iteration, a linear combination of some linearly independent directions can be taken as a search direction and perform the line search along this combined direction. A global minimizer can then be attained on a larger region, in particular, the subspace spanned by these linearly independent directions. The extension of searching along one single direction is shown in the following theorem.

Theorem 3.4. *(Minimization on the linear variety)*

Let f be given as in (3.4) *and let $\{d_0, d_1, \dots, d_{k-1}\}$ be a collection of linearly independent vectors in $\mathbb{R}^n$ with $k \leq n$. Let V_k be the subspace spanned by $\{d_0, d_1, \dots, d_{k-1}\}$. For any $x_0 \in \mathbb{R}^n$, let*

$$x_V = x_0 + \lambda v_k,$$

where v_k be any nonzero vector in V_k. Therefore,

$$f(x_V) = \min_{x \in x_0 + V_k} f(x) \tag{3.12}$$

if and only if

$$\lambda = -\frac{\nabla f(x_0)^T d_i}{d_i^T Q v_k}, \tag{3.13}$$

for $i = 0, 1, \dots, k-1$.

Proof. Since

$$x_V = x_0 + \lambda v_k,$$

therefore,

$$\nabla f(x_V) = \nabla f(x_0) + \lambda Q v_k.$$

Using Theorem 3.3, it follows that x_V is the minimizer of f on $x_0 + V_k$ if and only if $\nabla f(x_V) \perp V_k$. Then

$$\nabla f(x_V)^T d_i = \nabla f(x_0)^T d_i + \lambda d_i^T Q v_k = 0,$$

for $i = 0, 1, \ldots, k-1$. Hence,

$$\lambda = -\frac{\nabla f(x_0)^T d_i}{d_i^T Q v_k},$$

for $i = 0, 1, \ldots, k-1$. □

It is clear that if $V = span\{d\}$, a one-dimensional subspace, then with $v_k = d$ and (3.13) gives

$$\lambda = -\frac{\nabla f(x_0)^T d}{d^T Q d},$$

which is the same as obtained from the exact minimization of $f(x)$ along $x_0 + \lambda d$. Also, it follows from (3.13) that if the collection $\{d_0, d_1, \ldots, d_{k-1}\}$ is a mutually conjugate set, with respect to Q, then with $v_k = d_0 + \cdots + d_{k-1}$, (3.13) gives

$$\lambda = -\frac{\nabla f(x_0)^T d_i}{d_i^T Q d_i}, \tag{3.14}$$

for $i = 0, 1, \ldots, k-1$.

The question now is how to further extend Theorem 3.4 to cover a more general class of functions. Consider the following two cases.

Case 1. The class of strictly convex and continuously differentiable functions $f : \mathbb{R}^n \to \mathbb{R}$. The basic approach is to approximate f locally by a quadratic model, say at the current estimate of the minimizer x_c,

$$f(x) \approx f(x_c) + \nabla f(x_c)^T d + \frac{1}{2}(x - x_c)^T \nabla^2 f(x_c)(x - x_c). \tag{3.15}$$

Since $\nabla^2 f(x_c)$ is positive definite, Q in (3.13) can be replaced by $\nabla^2 f(x_c)$.

Case 2. The class of twice continuously differentiable functions $f : \mathbb{R}^n \to \mathbb{R}$. As in Case 1., some approximations on the Hessian can replace Q in (3.13), in particular, some quasi-Newton updates with positive definiteness preservation property.

However, the approach to be taken for the implementation in this thesis is to perform the inexact line search with the properly chosen step length satisfying the criteria for convergence.

3.3 Hybrid Directions

Some combinations of the existing and widely used directions will be taken to test numerically on the standard test problems from Moré, J.J. *et al.* (1981). Based on the approach in Section 3.2 and at the same time to fit the line search framework, the descent properly is checked for the combined directions. Since the combined directions are taken from the existing directions, They are called the hybrid directions.

The hybrid directions taken here are the following choices,

$$(1)\ \ v = (1-\gamma)d^{PR} + \gamma d^{BFGS}, \quad \gamma = 0, 0.1, \ldots, 1, \tag{3.16}$$

$$(2)\ \ v = \gamma d^{PR} + d^{BFGS}, \quad \gamma = 0, 0.1, \ldots, 1, \tag{3.17}$$

$$(3)\ \ v = d^{SD} + d^{PR} + d^{BFGS}, \tag{3.18}$$

$$(4)\ \ v = d^{SD} + (1-\gamma)d^{PR} + \gamma d^{BFGS}, \quad \gamma = 0, 0.1, \ldots, 1. \tag{3.19}$$

Algorithm 3.2. (Hybrid Direction Algorithm)

Given $f : \mathbb{R}^n \to \mathbb{R},\ f \in C^1$, *a starting point* $x_0 \in \mathbb{R}^n$, *and* $tol,\ \epsilon > 0$.

At iteration j, $j = 0,\ 1,\ \ldots.$

Step *A. Generate the search directions* $d_0^j,\ d_1^j,\ \ldots,\ d_{k-1}^j$.

(*For the implementation in this thesis,* $k = 2$ *or* 3, *and the choices of the directions are*

$$d_0^j = d_j^{SD} = -\nabla f(x_j),$$

$$d_1^j = d_j^{PR} = -\nabla f(x_j) + \beta_j^{PR} d_{j-1},$$

where $\beta_j^{\mathrm{PR}} = \dfrac{\nabla f(x_j)^T\big(\nabla f(x_j) - \nabla f(x_{j-1})\big)}{\nabla f(x_{j-1})^T \nabla f(x_{j-1})}$.

$$d_2^j = d_j^{BFGS} = -H_j \nabla f(x_j),$$

where $H_j = \left(I - \rho_j s_j y_j^T\right) H_j \left(I - \rho_j y_j s_j^T\right) + \rho_j s_j s_j^T$ *and*
$\rho_j = \dfrac{1}{y_j^T s_j}$, *with* $y_j^T s_j > 0$.)

Step *B. Take a linear combination of the directions from Step A. Set*
$v^j = \beta_0 d_0^j + \beta_1 d_1^j + \cdots + \beta_{k-1} d_{k-1}^j$.

(*The four choices used in the implementation in this thesis are*

$$\begin{aligned} v_{(1)}^j &= (1-\gamma)d_1^j + \gamma d_2^j, \\ v_{(2)}^j &= \gamma d_1^j + d_2^j, \\ v_{(3)}^j &= d_0^j + d_1^j + d_2^j, \\ v_{(4)}^j &= d_0^j + (1-\gamma)d_1^j + \gamma d_2^j. \end{aligned}$$)

Step *C. Check the descent property of the combined direction in Step B,* v^j. *If* $\nabla f(x_j)^T v^j < 0$ *goto next step, if not, restart with the steepest descent direction. Set*

$$v^j = -\nabla f(x_j).$$

Step *D. Perform the line search from* x_{j-1} *along* $v^j_{(i)}$ *to obtain the admissible scalar* λ_j *and set the new iterate as*

$$x_{j+1} = x_j + \lambda_j v^j_{(i)}.$$

Obtain the scalar λ *by using the Armijo's conditions, backtracking techniques, Wolfe or strong Wolfe conditions.*

Step *E. Test the admissibility of* x_{j+1} *by checking the conditions,*

$$\|\nabla f(x_{j+1})\| \leq \epsilon$$

and

$$\|x_{j+1} - x_j\| \leq tol.$$

If these conditions are satisfied then STOP, *else go back to Step A.*

3.4 Standard Test Problems

To test the performances of the proposed hybrid directions described in Section 3.3, Algorithm 3.2 is implemented with the test functions taken from the standard test problems for unconstrained minimization of Moré, J.J. *et al.* (1981). These selected test functions are listed as follows:

1. The Variably Dimensioned Function

$$f(x) = \sum_{i=1}^{m} f_i^2(x), \; m = n + 2,$$

where n is the number of variables and

$$f_i(x) = x_i - 1, \quad 1 \le i \le n,$$
$$f_{n+1}(x) = \sum_{j=1}^{n} j(x_j - 1),$$
$$f_{n+2}(x) = \left(\sum_{j=1}^{n} j(x_j - 1) \right)^2 .$$

2. The Penalty Function I

$$f(x) = \sum_{i=1}^{m} f_i^2(x), \; m = n + 1,$$

where n is the number of variables and

$$f_i(x) = a^{1/2}(x_i - 1), \quad 1 \le i \le n,$$
$$f_{n+1}(x) = \left(\sum_{j=1}^{n} x_j^2 \right)^2 - \frac{1}{4},$$

where $a = 10^{-5}$.

3. The Penalty Function II

$$f(x) = \sum_{i=1}^{m} f_i^2(x), \; m = 2n,$$

where n is the number of variables and

$$f_1(x) = x_1 - 0.2,$$
$$f_i(x) = a^{1/2} \left(e^{\frac{x_i}{10}} + e^{\frac{x_{i-1}}{10}} - y_i \right), \quad 2 \le i \le n,$$
$$f_i(x) = a^{1/2} \left(e^{\frac{x_i - n + 1}{10}} - e^{\frac{-1}{10}} \right), \quad n < i < 2n,$$
$$f_{2n}(x) = \left(\sum_{j=1}^{n} (n - j + 1) x_j^2 \right) - 1,$$

where $a = 10^{-5}$ and $y_i = e^{\frac{i}{10}} + e^{\frac{i-1}{10}}$.

4. The Biggs EXP6 Function

$$f(x) = \sum_{i=1}^{m} f_i^2(x), \; m \ge n, n = 6,$$

where n is the number of variables and

$$f_i(x) = e^{-t_i x_1} - x_4 e^{-t_i x_2} + x_6 e^{-t_i x_5} - y_i,$$

where $t_i = (0.1)i$ and $y_i = e^{-t_i} - 5e^{-10t_i} + 3e^{-4t_i}$.

5. The Brown Badly Scaled Function

$$f(x) = (x_1 - 10^6)^2 + (x_2 - 2 \cdot 10^{-6})^2 + (x_1 x_2 - 2)^2.$$

6. The Brown and Dennis Function

$$f(x) = \sum_{i=1}^{m} f_i^2(x), \; m \geq n, \; n = 4,$$

where n is the number of variables and

$$f_i(x) = (x_1 + t_i x_2 - e^{-t_i})^2 + (x_3 + x_4 \sin(t_i) - \cos(t_i))^2,$$

where $t_i = i/5$.

Chapter IV

Numerical Results

In Chapter III, some choices of hybrid directions are presented for investigation within the line search framework. Algorithm 3.2 described in Section 3.3 has been implemented by using some standard test problems stated in Section 3.4, as the test cases. Performances of these proposed hybrid directions are illustrated by the numerical results obtained from the implementation of Algorithm 3.2.

4.1 Implementation of the Hybrid Direction Algorithm

The implementation of Algorithm 3.2 aims at the following.

1. To compare the performances based on the hybrid directions with those based on a single direction, i.e., the steepest descent direction, the PR-CG direction and BFGS direction.
2. To compare the efficiency of the different conditions used to obtain the scalar along the search direction.
3. To compare the performances between the choices of the hybrid directions.

The details for implementing Algorithm 3.2 can be described as follows.

1. The line search routines satisfying the Wolfe and strong Wolfe conditions are coded as given in Algorithms 3.2 and 3.3, pp.59-60 in Nocedal, J. and Wright, S.J. (1999) with $0 < \alpha < \beta < 1$. The values for α and β are set to be 10^{-4} and 10^{-1}, respectively.

2. The backtracking techniques is taken from Numerical Recipes in Fortran 77: The Art of Scientific Computing (Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Fiannery, B.P. (1986-1992)).
3. The Armijo line search is coded according to Algorithm 1. for the modified steepest descent method in Vrahatis, M.N. *et al.* (2000).
4. The computer codes are written in Fortran 90 and implemented in double precision arithmetic. The codes are run on a FORTRAN PowerStation4.0 at the Computer Laboratory, School of Mathematics, Suranaree University of Technology.
5. The stopping conditions: $\|\nabla f(x_k)\| \leq 10^{-5}$ and $\|x_{k+1} - x_k\| \leq 10^{-10}$.

The descriptions of the parameters presenting in Tables 4.1-4.6 are as follows.

$n =$ dimension of the test problems,

$x_0 = (x_1, x_2, \ldots, x_n)$ is the starting point,

IT = the number of iterations,

FE = the total number of function evaluations including the gradient components,

MAXFE = the maximum number of function evaluations(FE),

$\gamma =$ constants used in the linear combination of the search directions,

$\theta = 1 - \gamma$,

Diverge : indication of divergence after 3000 iterations or FE > 90000,

PR : conjugate gradient direction based on Polak-Ribière formula,

SD : steepest descent direction,

BFGS : quasi-Newton directions based on the BFGS update,

θPR + γBFGS : Hybrid direction (1) between PR and BFGS directions with $\gamma = 0, 0.1, \ldots, 0.9, 1$,

γPR + BFGS : Hybrid direction (2) between PR and BFGS directions with $\gamma = 0, 0.1, \ldots, 0.9, 1$,

SD + PR + BFGS : Hybrid direction (3) between SD, PR, and BFGS directions with scalar multiple = 1,

SD + θPR + γBFGS : Hybrid direction (4) between SD and Hybrid direction (1),

Backtracking : the backtracking techniques,

Strong Wolfe : the line search with the strong Wolfe conditions,

Wolfe : the line search with the Wolfe conditions,

Armijo : the line search based on the adaptive step length of the modified steepest descent method given by Vrahatis, M.N. *et al.* (2000).

4.2 Numerical Results

The performances of the hybrid directions (1)–(4) described in Section 3.3 can be expressed in 3 cases based on the numerical results obtained from implementing Algorithm 3.2 with the standard test problems described in Section 3.4.

Case 1. The hybrid directions give better performances than the single direction. The objective functions are the Variably dimensioned function and the Penalty function I. The dimensions of these selected problems can be varied as shown in Examples 4.1 and 4.2. The numerical results for these 2 functions are given in Tables 4.1 and 4.2.

Example 4.1. *Variably Dimensioned Function, (Moré, J.J., et al., 1981). The function f is given by*

$$f(x) = \sum_{i=1}^{m} f_i^2(x), \; m = n + 2,$$

where n is the number of variables and

$$\begin{aligned} f_i(x) &= x_i - 1, \\ f_{n+1}(x) &= \sum_{j=1}^{n} j(x_j - 1), \\ f_{n+2}(x) &= \left(\sum_{j=1}^{n} j(x_j - 1) \right)^2 . \end{aligned}$$

The standard starting point is $x_0 = (\xi_j)$, where $\xi_j = 1 - (j/n)$. The numerical results are shown in Table 4.1.

Table 4.1. Results for the Variably Dimensioned Function

Directions	n	γ	Backtracking	Strong Wolfe	Wolfe	Armijo
			IT/FE	IT/FE	IT/FE	IT/FE
SD	4	1.00	5 /61	5 /89	123 /1257	13 /185
PR		.00	5 /61	5 /89	123 /1257	13 /185
BFGS		1.00	9 /59	4 /77	5 /78	13 /84
(1) θPR+γBFGS		.10	8 /83	8 /164	35 /373	14 /195
		.20	9 /98	6 /140	20 /220	15 /199
		.30	6 /64	6 /119	15 /168	18 /232
		.40	14 /128	6 /118	11 /132	22 /276
		.50	11 /104	5 /85	314 /2854	12 /163
		.60	12 /108	6 /135	22 /219	15 /185
		.70	147 /903	7 /113	12 /131	21 /245
		.80	15 /116	6 /112	26 /230	14 /162
		.90	7 /64	6 /117	43 /324	13 /141
(2) γPR+BFGS		.10	7 /64	6 /107	49 /366	13 /141
		.20	16 /122	7 /131	29 /254	14 /162
		.30	259 /1575	7 /113	13 /140	21 /245
		.40	6 /67	6 /123	23 /228	15 /185
		.50	12 /111	4 /67	10 /120	26 /320
		.60	7 /80	6 /118	12 /142	21 /265
		.70	6 /64	7 /139	16 /178	18 /232

continued on next page

continued from previous page

Directions	n	γ	Backtracking	Strong Wolfe	Wolfe	Armijo
			IT/FE	IT/FE	IT/FE	IT/FE
		.80	14 /139	6 /140	21 /230	15 /199
		.90	8 /83	5 /111	42 /443	14 /195
(3) SD+PR+BFGS		.00	14 /131	5 /93	193 /2149	12 /185
(4) SD+θPR+γBFGS		.10	15 /143	7 /125	66 /747	13 /196
		.20	9 /93	8 /171	35 /407	14 /208
		.30	14 /136	5 /116	31 /362	15 /220
		.40	10 /102	6 /145	20 /239	15 /213
		.50	9 /86	8 /172	18 /216	17 /237
		.60	9 /86	6 /124	15 /182	18 /249
		.70	13 /136	6 /117	16 /203	20 /273
		.80	9 /97	6 /123	11 /142	22 /297
		.90	12 /125	7 /95	12 /153	24 /321
SD	8	1.00	8 /141	4 /135	32 /564	20 /447
PR		.00	8 /141	4 /135	32 /564	20 /447
BFGS		1.00	16 /168	4 /173	4 /164	18 /192
(1) θPR+γBFGS		.10	9 /152	7 /248	20 /360	23 /501
		.20	11 /191	6 /189	13 /241	24 /508
		.30	10 /176	5 /156	9 /173	28 /580
		.40	10 /178	5 /148	157 /2534	17 /378
		.50	5 /96	4 /132	32 /533	20 /428
		.60	23 /291	6 /184	13 /229	24 /485
		.70	18 /254	5 /144	176 /2663	17 /362
		.80	5 /90	5 /154	13 /217	24 /462
		.90	12 /172	5 /150	14 /219	24 /439
(2) γPR+BFGS		.10	17 /227	5 /150	14 /219	24 /439
		.20	5 /90	5 /154	14 /232	24 /462
		.30	19 /267	5 /144	189 /2858	17 /362
		.40	24 /302	6 /184	13 /229	24 /485
		.50	5 /96	4 /132	33 /549	20 /428
		.60	11 /190	5 /148	168 /2710	17 /378
		.70	11 /188	5 /156	9 /173	28 /580
		.80	12 /203	6 /189	13 /241	24 /508
		.90	10 /164	7 /248	20 /360	23 /501
(3) SD+PR+BFGS		.00	8 /140	4 /138	32 /595	20 /466
(4) SD+θPR+γBFGS		.10	5 /98	6 /244	25 /469	22 /504
		.20	9 /173	7 /254	20 /379	23 /523
		.30	9 /169	5 /182	16 /307	24 /542
		.40	6 /119	6 /194	13 /253	24 /531
		.50	9 /168	7 /213	11 /217	26 /569
		.60	8 /146	5 /160	9 /181	28 /607
		.70	10 /179	6 /137	7 /145	30 /645
		.80	13 /225	5 /152	157 /2690	17 /394
		.90	8 /141	5 /190	44 /769	18 /411
SD	12	1.00	15 /331	6 /250	17 /448	27 /752

continued on next page

continued from previous page

Directions	n	γ	Backtracking	Strong Wolfe	Wolfe	Armijo
			IT/FE	IT/FE	IT/FE	IT/FE
PR		.00	15 /331	6 /250	17 /448	27 /752
BFGS		1.00	19 /278	5 /295	9 /322	23 /339
(1) θPR+γBFGS		.10	12 /272	4 /188	10 /275	30 /824
		.20	15 /345	7 /231	9 /268	32 /860
		.30	7 /166	7 /300	57 /1325	19 /542
		.40	16 /379	8 /430	30 /719	23 /634
		.50	9 /213	6 /245	17 /432	27 /726
		.60	7 /171	7 /225	9 /260	32 /829
		.70	14 /323	8 /423	30 /690	23 /612
		.80	10 /247	7 /219	9 /252	32 /798
		.90	21 /409	7 /213	9 /244	32 /767
(2) γPR+BFGS		.10	19 /378	7 /213	9 /244	32 /767
		.20	10 /247	7 /219	9 /252	32 /798
		.30	17 /372	8 /423	30 /690	23 /612
		.40	7 /171	7 /225	9 /260	32 /829
		.50	9 /213	6 /245	17 /432	27 /726
		.60	14 /344	8 /430	30 /719	23 /634
		.70	7 /166	7 /300	57 /1325	19 /542
		.80	15 /347	7 /231	9 /268	32 /860
		.90	12 /272	4 /188	10 /275	30 /824
(3) SD+PR+BFGS		.00	15 /362	6 /255	17 /464	27 /778
(4) SD+θPR+γBFGS		.10	14 /364	8 /356	14 /389	28 /803
		.20	10 /248	4 /191	10 /284	30 /853
		.30	10 /244	8 /262	10 /280	31 /878
		.40	10 /242	7 /237	9 /276	32 /891
		.50	151 /2505	8 /309	125 /2952	19 /561
		.60	48 /906	7 /306	57 /1381	19 /560
		.70	28 /542	8 /378	39 /951	21 /608
		.80	17 /368	8 /437	30 /748	23 /656
		.90	20 /410	7 /336	23 /583	25 /704
SD	16	1.00	6 /185	9 /519	26 /796	26 /891
PR		.00	6 /185	9 /519	26 /796	26 /891
BFGS		1.00	21 /395	10 /519	10 /450	28 /527
(1) θPR+γBFGS		.10	8 /232	7 /330	14 /447	29 /978
		.20	10 /302	6 /348	17 /558	32 /1065
		.30	14 /418	8 /337	10 /342	35 /1138
		.40	41 /910	9 /506	33 /946	22 /754
		.50	17 /419	9 /511	26 /771	26 /866
		.60	9 /233	6 /343	17 /542	32 /1034
		.70	7 /211	9 /498	33 /914	22 /733
		.80	11 /295	6 /338	17 /526	32 /1003
		.90	10 /265	6 /333	17 /510	32 /972
(2) γPR+BFGS		.10	10 /265	6 /333	18 /535	32 /972
		.20	11 /295	6 /338	17 /526	32 /1003
		.30	7 /211	9 /498	33 /914	22 /733

continued on next page

continued from previous page

Directions	n	γ	Backtracking	Strong Wolfe	Wolfe	Armijo
			IT/FE	IT/FE	IT/FE	IT/FE
		.40	9 /233	6 /343	17 /542	32 /1034
		.50	17 /419	9 /511	26 /771	26 /866
		.60	41 /910	9 /506	33 /946	22 /754
		.70	14 /420	8 /337	10 /342	35 /1138
		.80	10 /302	6 /348	17 /558	32 /1065
		.90	8 /232	7 /330	14 /447	29 /978
(3) SD+PR+BFGS		.00	7 /213	9 /527	26 /821	26 /916
(4) SD+θPR+γBFGS		.10	9 /300	8 /481	22 /716	27 /946
		.20	8 /258	7 /336	14 /460	29 /1006
		.30	9 /279	8 /380	17 /557	30 /1036
		.40	13 /395	6 /353	17 /574	32 /1096
		.50	11 /311	6 /269	11 /380	33 /1112
		.60	7 /233	8 /344	10 /351	35 /1172
		.70	12 /342	9 /436	123 /3508	21 /747
		.80	8 /242	9 /514	33 /978	22 /775
		.90	12 /337	7 /369	38 /1119	24 /833
SD	32	1.00	20 /955	9 /836	22 /1116	32 /1759
PR		.00	20 /955	9 /836	22 /1116	32 /1759
BFGS		1.00	26 /918	12 /1220	13 /1055	28 /992
(1) θPR+γBFGS		.10	12 /605	10 /823	21 /1090	35 /1903
		.20	52 /2055	9 /678	13 /694	38 /2047
		.30	28 /1241	7 /622	300 /13953	24 /1353
		.40	17 /877	9 /830	45 /2202	28 /1540
		.50	14 /676	9 /828	22 /1095	32 /1728
		.60	9 /459	9 /670	13 /682	38 /2010
		.70	16 /764	9 /822	50 /2385	28 /1513
		.80	12 /596	9 /662	13 /670	38 /1973
		.90	10 /492	9 /654	13 /658	38 /1936
(2) γPR+BFGS		.10	10 /492	9 /654	13 /658	38 /1936
		.20	12 /598	9 /662	13 /670	38 /1973
		.30	16 /767	9 /822	48 /2293	28 /1513
		.40	9 /459	9 /670	13 /682	38 /2010
		.50	13 /640	9 /828	22 /1095	32 /1728
		.60	22 /1059	9 /830	48 /2340	28 /1540
		.70	29 /1282	7 /622	322 /14965	24 /1353
		.80	52 /2055	9 /678	13 /694	38 /2047
		.90	13 /643	10 /823	21 /1090	35 /1903
(3) SD+PR+BFGS		.00	13 /663	9 /844	22 /1137	32 /1790
(4) SD+θPR+γBFGS		.10	9 /462	9 /693	22 /1158	33 /1839
		.20	13 /644	10 /832	21 /1110	35 /1937
		.30	10 /518	9 /847	18 /971	37 /2035
		.40	7 /368	9 /686	13 /706	38 /2084
		.50	15 /766	9 /508	9 /508	39 /2115
		.60	13 /674	7 /628	300 /14252	24 /1376
		.70	13 /675	10 /812	69 /3359	26 /1471

continued on next page

continued from previous page

Directions	n	γ	Backtracking	Strong Wolfe	Wolfe	Armijo
			IT/FE	IT/FE	IT/FE	IT/FE
		.80	9 /470	9 /838	45 /2246	28 /1567
		.90	15 /757	10 /1148	29 /1445	30 /1663
SD	64	1.00	13 /1154	11 /1884	27 /2371	38 /3495
PR		.00	13 /1154	11 /1884	27 /2371	38 /3495
BFGS		1.00	32 /2178	15 /2901	18 /2838	44 /2979
(1) θPR+γBFGS		.10	13 /1093	13 /1673	21 /1870	41 /3744
		.20	9 /787	12 /1250	13 /1201	44 /3993
		.30	20 /1701	10 /1256	159 /13000	29 /2721
		.40	14 /1208	11 /1808	38 /3220	33 /3048
		.50	11 /996	11 /1874	32 /2749	38 /3458
		.60	11 /972	12 /1239	13 /1189	44 /3950
		.70	14 /1259	11 /1798	38 /3183	33 /3016
		.80	12 /1045	12 /1228	13 /1177	44 /3907
		.90	25 /1933	11 /1138	13 /1165	44 /3864
(2) γPR+BFGS		.10	35 /2629	12 /1217	13 /1165	44 /3864
		.20	13 /1117	12 /1228	13 /1177	44 /3907
		.30	16 /1418	11 /1798	38 /3183	33 /3016
		.40	11 /973	12 /1239	13 /1189	44 /3950
		.50	11 /997	11 /1874	31 /2669	38 /3458
		.60	12 /1070	11 /1808	38 /3220	33 /3048
		.70	22 /1846	10 /1256	159 /13000	29 /2721
		.80	9 /787	12 /1250	13 /1201	44 /3993
		.90	15 /1233	13 /1673	21 /1870	41 /3744
(3) SD+PR+BFGS		.00	10 /912	11 /1894	27 /2397	38 /3532
(4) SD+θPR+γBFGS		.10	11 /991	9 /1393	25 /2260	39 /3616
		.20	10 /886	13 /1685	21 /1890	41 /3784
		.30	18 /1608	10 /1748	24 /2146	42 /3868
		.40	18 /1595	12 /1261	13 /1213	44 /4036
		.50	12 /1082	9 /929	11 /1031	45 /4098
		.60	19 /1657	10 /1265	159 /13158	29 /2749
		.70	11 /1002	7 /1344	57 /4843	31 /2914
		.80	13 /1195	11 /1818	38 /3257	33 /3080
		.90	201 /14330	13 /2565	32 /2791	35 /3246
SD	128	1.00	21 /3248	9 /2949	20 /3171	41 /6657
PR		.00	21 /3248	9 /2949	20 /3171	41 /6657
BFGS		1.00	34 /4583	17 /6272	14 /5110	56 /7565
(1) θPR+γBFGS		.10	15 /2432	13 /3436	24 /3821	44 /7107
		.20	13 /2049	11 /2617	16 /2632	47 /7557
		.30	22 /3412	10 /2588	143 /21396	34 /5575
		.40	14 /2283	10 /3238	41 /6312	37 /6021
		.50	14 /2229	9 /2941	20 /3152	41 /6617
		.60	14 /2242	11 /2607	16 /2617	47 /7511
		.70	15 /2351	10 /3229	40 /6125	37 /5985
		.80	16 /2539	11 /2597	16 /2602	47 /7465

continued on next page

continued from previous page

Directions	n	γ	Backtracking	Strong Wolfe	Wolfe	Armijo
			IT/FE	IT/FE	IT/FE	IT/FE
		.90	16 /2495	11 /2587	16 /2587	47 /7419
(2) γPR+BFGS		.10	24 /3590	11 /2587	16 /2587	47 /7419
		.20	18 /2823	11 /2597	16 /2602	47 /7465
		.30	14 /2217	10 /3229	41 /6272	37 /5985
		.40	15 /2384	11 /2607	16 /2617	47 /7511
		.50	13 /2095	9 /2941	20 /3152	41 /6617
		.60	17 /2719	10 /3238	41 /6312	37 /6021
		.70	18 /2865	10 /2588	144 /21545	34 /5575
		.80	12 /1920	11 /2617	16 /2632	47 /7557
		.90	18 /2871	13 /3436	24 /3823	44 /7107
(3) SD+PR+BFGS		.00	12 /1949	9 /2957	20 /3190	41 /6697
(4) SD+θPR+γBFGS		.10	15 /2418	11 /3142	22 /3585	42 /6848
		.20	13 /2092	13 /3448	24 /3844	44 /7150
		.30	17 /2755	8 /2562	19 /3106	45 /7301
		.40	15 /2425	11 /2627	16 /2647	47 /7603
		.50	13 /2114	10 /2105	17 /2821	48 /7728
		.60	17 /2714	10 /2597	143 /21538	34 /5608
		.70	18 /2895	13 /4350	44 /6772	35 /5757
		.80	18 /2825	10 /3247	41 /6352	37 /6057
		.90	18 /2897	10 /4136	33 /5193	39 /6357

In Table 4.1, the numerical results show that as the dimension gets higher, $0.5d^{PR} + 0.5d^{BFGS}$ and $0.2d^{PR} + 0.8d^{BFGS}$ when implemented with backtracking technique give significant reduction in the number of iterations and number of function evaluations in comparison with the performances based on the single direction d^{SD}, d^{PR} and d^{BFGS}. The hybrid direction(2), $0.5d^{PR} + d^{BFGS}$ also gives better performance in comparison with the performances based on d^{SD}, d^{PR} and d^{BFGS}, similarly for $d^{SD} + 0.8d^{PR} + 0.2d^{BFGS}$. For the dimension 16, even the d^{BFGS} gives worse performances than the d^{SD}. For the dimensions 32, 64 and 128, it can be seen that the hybrid directions(1)–(4), with the backtracking technique, give better performances and significant reduction in the number of iterations and function evaluations in almost all choices of the scalar multiples presented.

Example 4.2. *Penalty Function I, (Moré, J.J., et al., 1981). The function f is given by*

$$f(x) = \sum_{i=1}^{m} f_i^2(x), \; m = n+1,$$

where n is the number of variables and

$$f_i(x) = a^{1/2}(x_i - 1), \;\; 1 \leq i \leq n,$$
$$f_{n+1}(x) = \left(\sum_{j=1}^{n} x_j^2 \right)^2 - \frac{1}{4},$$

where $a = 10^{-5}$. The starting point is $x_0 = (\xi_j)$, where $\xi_j = j$. The numerical results are shown in Table 4.2.

Table 4.2. Results for the Penalty Function I

Directions	n	γ	Backtracking	Strong Wolfe	Wolfe	Armijo
			IT/FE	IT/FE	IT/FE	IT/FE
SD	4	1.00	Diverge	Diverge	Diverge	Diverge
PR		.00	Diverge	20 /445	Diverge	Diverge
BFGS		1.00	181 /977	31 /604	116 /1197	56 /319
(1) θPR+γBFGS		.10	171 /879	76 /1446	173 /1005	81 /483
		.20	63 /327	37 /721	101 /634	75 /465
		.30	212 /1134	31 /531	81 /584	46 /294
		.40	56 /293	19 /404	73 /556	36 /244
		.50	36 /200	23 /452	62 /501	54 /360
		.60	40 /216	22 /371	46 /428	49 /306
		.70	172 /909	16 /335	112 /1070	58 /364
		.80	185 /984	29 /571	33 /347	60 /372
		.90	158 /862	18 /411	19 /219	32 /204
(2) γPR+BFGS		.10	176 /958	19 /411	30 /314	40 /252
		.20	179 /987	25 /536	43 /376	48 /300
		.30	172 /957	24 /488	37 /333	50 /315
		.40	31 /178	21 /392	69 /397	37 /266
		.50	38 /221	25 /508	308 /1620	45 /345
		.60	34 /204	31 /547	33 /292	57 /392
		.70	166 /982	17 /289	33 /262	43 /307
		.80	41 /238	27 /458	48 /358	43 /311
		.90	37 /220	21 /364	58 /397	48 /352
(3) SD+PR+BFGS		.00	33 /210	21 /327	38 /304	55 /498

continued on next page

continued from previous page

Directions	n	γ	Backtracking	Strong Wolfe	Wolfe	Armijo
			IT/FE	IT/FE	IT/FE	IT/FE
(4) SD+θPR+γBFGS		.10	117 /658	15 /348	340 /2048	69 /496
		.20	96 /545	56 /858	169 /1025	49 /343
		.30	21 /128	28 /506	315 /1882	63 /445
		.40	31 /197	62 /646	113 /679	54 /371
		.50	32 /188	45 /682	99 /598	51 /351
		.60	31 /189	43 /690	91 /566	30 /231
		.70	159 /918	41 /734	168 /1193	47 /346
		.80	24 /142	19 /368	68 /468	42 /312
		.90	24 /145	38 /523	62 /459	38 /284
SD	8	1.00	Diverge	Diverge	Diverge	Diverge
PR		.00	Diverge	175 /7590	Diverge	Diverge
BFGS		1.00	147 /1382	25 /842	112 /2020	59 /580
(1) θPR+γBFGS		.10	174 /1599	71 /2577	161 /1712	91 /956
		.20	89 /824	57 /1459	212 /2404	85 /947
		.30	173 /1635	27 /938	165 /2210	59 /624
		.40	152 /1428	89 /3021	167 /2366	39 /427
		.50	136 /1268	87 /2958	162 /2273	59 /630
		.60	144 /1373	30 /983	118 /1880	55 /593
		.70	44 /422	87 /2914	110 /1767	54 /571
		.80	146 /1383	30 /992	110 /1760	53 /567
		.90	46 /444	25 /838	93 /1617	52 /550
(2) γPR+BFGS		.10	39 /382	19 /637	98 /1614	64 /669
		.20	155 /1449	16 /632	120 /1748	52 /559
		.30	37 /357	74 /2441	113 /1717	52 /554
		.40	161 /1577	14 /497	183 /1762	49 /536
		.50	313 /2919	93 /2966	172 /2136	48 /530
		.60	134 /1324	99 /3089	109 /1378	56 /624
		.70	134 /1314	31 /967	94 /1321	53 /606
		.80	34 /341	24 /648	111 /1381	53 /609
		.90	35 /360	31 /1068	60 /692	52 /589
(3) SD+PR+BFGS		.00	32 /336	145 /4146	132 /1653	59 /727
(4) SD+θPR+γBFGS		.10	465 /4435	23 /1006	295 /2964	82 /921
		.20	37 /372	71 /2124	178 /1839	67 /743
		.30	47 /473	28 /784	149 /1516	59 /675
		.40	46 /459	61 /1121	237 /2409	62 /719
		.50	39 /388	24 /766	222 /2279	58 /662
		.60	34 /342	20 /735	159 /1799	47 /553
		.70	136 /1332	164 /5413	135 /1739	54 /625
		.80	43 /437	107 /3207	134 /1679	39 /465
		.90	44 /442	105 /3038	144 /1775	47 /547
SD	16	1.00	Diverge	Diverge	Diverge	Diverge
PR		.00	Diverge	155 /12213	Diverge	Diverge
BFGS		1.00	159 /2769	77 /4877	100 /3392	69 /1244
(1) θPR+γBFGS		.10	282 /4851	89 /4656	185 /3530	96 /1828

continued on next page

continued from previous page

Directions	n	γ	Backtracking	Strong Wolfe	Wolfe	Armijo
			IT/FE	IT/FE	IT/FE	IT/FE
		.20	187 /3264	193 /12300	95 /1982	76 /1470
		.30	68 /1201	114 /6171	145 /3515	60 /1157
		.40	158 /2794	77 /4556	64 /1669	63 /1232
		.50	54 /969	85 /5225	138 /3661	54 /1059
		.60	144 /2548	88 /5393	110 /3321	60 /1178
		.70	134 /2348	33 /2131	45 /1287	52 /1004
		.80	147 /2584	74 /4831	35 /1114	60 /1152
		.90	128 /2267	79 /4738	15 /372	43 /834
(2) γPR+BFGS		.10	43 /786	85 /5318	81 /2267	60 /1135
		.20	48 /863	83 /5100	100 /2662	57 /1099
		.30	131 /2324	22 /1434	102 /2919	47 /926
		.40	56 /1020	76 /4516	162 /3134	56 /1089
		.50	131 /2362	91 /5180	121 /2963	37 /761
		.60	116 /2078	30 /1793	45 /1137	55 /1098
		.70	41 /756	83 /4539	34 /827	50 /1020
		.80	119 /2139	79 /4646	100 /2509	56 /1137
		.90	120 /2180	20 /988	62 /1217	52 /1062
(3) SD+PR+BFGS		.00	33 /629	153 /7574	116 /2602	50 /1049
(4) SD+θPR+γBFGS		.10	403 /7112	79 /6090	489 /8829	81 /1593
		.20	236 /4200	64 /3405	205 /3738	65 /1291
		.30	170 /3065	182 /8955	238 /4316	64 /1292
		.40	39 /722	183 /10542	106 /1959	62 /1253
		.50	40 /740	105 /5552	90 /1702	51 /1031
		.60	131 /2356	118 /6418	149 /3059	54 /1095
		.70	135 /2433	180 /10310	115 /2614	52 /1053
		.80	44 /804	94 /4869	122 /2847	38 /796
		.90	128 /2305	32 /1699	54 /1182	52 /1055
SD	32	1.00	Diverge	Diverge	Diverge	Diverge
PR		.00	Diverge	39 /5967	Diverge	Diverge
BFGS		1.00	158 /5315	64 /7206	84 /5968	70 /2407
(1) θPR+γBFGS		.10	138 /4659	188 /19692	171 /6297	97 /3466
		.20	76 /2573	149 /13362	186 /7515	70 /2535
		.30	65 /2227	90 /9909	125 /6080	58 /2131
		.40	142 /4784	33 /3298	52 /2822	57 /2088
		.50	52 /1778	31 /3467	133 /6832	59 /2155
		.60	54 /1873	75 /8600	48 /2609	58 /2138
		.70	112 /3811	25 /2726	35 /2057	51 /1861
		.80	123 /4147	76 /8752	37 /2240	55 /1998
		.90	125 /4228	28 /3277	89 /4971	56 /2023
(2) γPR+BFGS		.10	117 /3953	26 /3186	81 /4874	48 /1760
		.20	131 /4414	70 /8036	94 /5303	56 /2034
		.30	46 /1597	29 /3268	40 /2087	50 /1833
		.40	137 /4642	70 /7338	137 /5430	46 /1712
		.50	36 /1263	23 /2671	100 /5088	50 /1857
		.60	104 /3559	28 /2647	51 /2335	54 /2008

continued on next page

continued from previous page

Directions	n	γ	Backtracking	Strong Wolfe	Wolfe	Armijo
			IT/FE	IT/FE	IT/FE	IT/FE
		.70	109 /3746	87 /8976	99 /4606	51 /1918
		.80	37 /1312	76 /8037	45 /1965	48 /1810
		.90	109 /3784	81 /9153	130 /4973	53 /2005
(3) SD+PR+BFGS		.00	35 /1249	30 /3134	102 /4279	53 /2046
(4) SD+θPR+γBFGS		.10	154 /5229	100 /9332	455 /15591	93 /3364
		.20	59 /2058	163 /16022	184 /6355	73 /2680
		.30	51 /1778	164 /14674	234 /8015	66 /2464
		.40	40 /1415	151 /11175	206 /7076	65 /2405
		.50	143 /4876	121 /11862	166 /6191	58 /2178
		.60	133 /4563	102 /10381	134 /5247	55 /2052
		.70	116 /3980	58 /6055	64 /2502	48 /1805
		.80	107 /3662	31 /2880	62 /2458	50 /1891
		.90	105 /3585	30 /3380	55 /2267	52 /1951
SD	64	1.00	Diverge	Diverge	Diverge	Diverge
PR		.00	Diverge	38 /10093	Diverge	Diverge
BFGS		1.00	74 /4905	78 /18619	44 /7725	116 /7700
(1) θPR+γBFGS		.10	228 /15017	86 /17675	163 /11892	94 /6436
		.20	166 /10954	47 /11276	166 /13282	73 /5065
		.30	44 /2986	78 /17969	122 /11250	61 /4260
		.40	144 /9520	23 /4743	50 /5057	57 /3998
		.50	116 /7684	71 /14866	123 /12331	55 /3877
		.60	141 /9313	26 /5143	90 /9853	50 /3534
		.70	126 /8346	22 /4875	34 /3734	58 /4036
		.80	117 /7775	21 /4621	71 /8611	54 /3768
		.90	89 /5975	24 /5649	83 /9856	45 /3162
(2) γPR+BFGS		.10	81 /5456	21 /4925	72 /8620	63 /4359
		.20	115 /7641	24 /5200	85 /9006	56 /3917
		.30	123 /8161	26 /5533	87 /9235	52 /3648
		.40	124 /8236	24 /5143	151 /10459	54 /3826
		.50	153 /10104	66 /14205	139 /11453	54 /3807
		.60	110 /7347	34 /6327	81 /7490	51 /3612
		.70	42 /2868	70 /15775	85 /8097	47 /3345
		.80	103 /6881	27 /5090	82 /7535	55 /3896
		.90	109 /7327	29 /5535	53 /4070	52 /3693
(3) SD+PR+BFGS		.00	93 /6244	36 /6173	101 /8135	56 /4009
(4) SD+θPR+γBFGS		.10	117 /7806	69 /19155	406 /26915	95 /6571
		.20	178 /11796	40 /7623	181 /12175	78 /5458
		.30	51 /3480	138 /25588	217 /14812	67 /4720
		.40	48 /3265	53 /7896	182 /12444	68 /4782
		.50	116 /7773	47 /8811	92 /6163	58 /4086
		.60	44 /3000	81 /15577	128 /9707	50 /3569
		.70	41 /2803	26 /4354	127 /9545	54 /3823
		.80	119 /7930	31 /5407	51 /3915	53 /3759
		.90	42 /2860	61 /11486	61 /4392	57 /4035

continued on next page

continued from previous page

Directions	n	γ	Backtracking	Strong Wolfe	Wolfe	Armijo
			IT/FE	IT/FE	IT/FE	IT/FE
SD	128	1.00	Diverge	Diverge	Diverge	Diverge
PR		.00	Diverge	124 /67087	Diverge	Diverge
BFGS		1.00	171 /22252	116 /53541	127 /35438	121 /15887
(1) θPR+γBFGS		.10	148 /19327	77 /30936	143 /20466	91 /12176
		.20	174 /22686	52 /17860	78 /12449	74 /9983
		.30	138 /18024	38 /14395	62 /10304	61 /8266
		.40	139 /18165	72 /28648	98 /20891	60 /8157
		.50	99 /12983	63 /27253	51 /10400	51 /6976
		.60	107 /14031	21 /8582	39 /7930	56 /7628
		.70	110 /14415	61 /27762	71 /16227	53 /7219
		.80	46 /6104	24 /10783	26 /6513	60 /8130
		.90	38 /5087	22 /9998	25 /7284	55 /7455
(2) γPR+BFGS		.10	44 /5855	24 /10148	32 /8071	61 /8250
		.20	37 /4944	24 /9223	30 /6643	51 /6967
		.30	50 /6659	57 /24785	71 /14957	51 /6959
		.40	145 /18925	26 /10391	41 /7677	52 /7142
		.50	108 /14164	63 /25844	71 /10518	54 /7375
		.60	103 /13555	69 /26233	102 /17040	57 /7792
		.70	97 /12774	31 /11706	41 /7353	50 /6864
		.80	46 /6168	22 /9103	41 /6531	55 /7553
		.90	43 /5793	31 /11214	45 /7065	52 /7150
(3) SD+PR+BFGS		.00	44 /5966	119 /43626	45 /7092	58 /7968
(4) SD+θPR+γBFGS		.10	128 /16797	67 /33413	351 /45814	127 /16929
		.20	74 /9793	52 /19234	148 /19429	68 /9256
		.30	137 /17955	62 /21778	109 /14506	74 /10035
		.40	48 /6413	39 /15078	92 /12131	68 /9260
		.50	43 /5810	42 /15060	141 /19725	61 /8318
		.60	48 /6421	40 /15201	70 /9452	58 /7932
		.70	43 /5797	48 /16918	115 /17102	52 /7147
		.80	38 /5129	68 /25487	112 /16804	55 /7541
		.90	34 /4595	31 /11851	57 /8266	55 /7548

In Table 4.2, the numerical results show that the performances of the hybrid directions(1)–(4), when implemented with the backtracking technique are better than those single directions, for example, the divergence occurs in the case of d^{SD}. Even when the dimension is 4, the hybrid directions(1)–(4) with almost all choices of scalars give better performances, for instance, $0.5d^{PR} + 0.5d^{BFGS}$. When the dimension is high, for instance in the case $n = 128$, it is very interesting to see that when the single direction d^{PR} is implemented alone, the divergence occurs, but when it is combined with d^{BFGS}, the hybrid direction really gives satisfactory results.

Case 2. In this case, it is observed that when the single direction is implemented alone, the divergence occurs; but when it is combined, the resulting direction behaviour improves. This behaviour is observed from the results obtained from the two standard test problems, the Penalty function II, as given in Example 4.3 and the Biggs EXP6 function, as given in Example 4.4.

Example 4.3. *Penalty Function II, (Moré, J.J., et al., 1981). The function f is given by*

$$f(x) = \sum_{i=1}^{m} f_i^2(x),\ m = 2n,$$

where n is the number of variables and

$$\begin{aligned} f_1(x) &= x_1 - 0.2, \\ f_i(x) &= a^{1/2}\left(e^{\frac{x_i}{10}} + e^{\frac{x_{i-1}}{10}} - y_i\right),\ \ 2 \le i \le n, \\ f_i(x) &= a^{1/2}\left(e^{\frac{x_i - n + 1}{10}} - e^{\frac{-1}{10}}\right),\ \ n < i < 2n, \\ f_{2n}(x) &= \left(\sum_{j=1}^{n} (n - j + 1)x_j^2\right) - 1, \end{aligned}$$

where $a = 10^{-5}$ and $y_i = e^{\frac{i}{10}} + e^{\frac{i-1}{10}}$. The starting point is $x_0 = (\frac{1}{2}, \dots, \frac{1}{2})$. The numerical results are shown in Table 4.3.

Table 4.3. Results for the Penalty Function II

Directions	n	γ	Backtracking	Strong Wolfe	Wolfe	Armijo
			IT/FE	IT/FE	IT/FE	IT/FE
SD	4	1.00	67 /408	35 /489	64 /531	49 /422
PR		.00	31 /192	11 /178	41 /346	35 /299
BFGS		1.00	18 /103	12 /156	13 /101	10 /64
(1) θPR+γBFGS		.10	34 /209	17 /300	118 /955	29 /250
		.20	23 /143	20 /336	53 /443	22 /185
		.30	45 /275	17 /281	55 /456	23 /191
		.40	32 /195	29 /441	47 /360	26 /209
		.50	39 /239	39 /610	45 /342	33 /258
		.60	16 /98	35 /543	21 /162	22 /168
		.70	19 /114	27 /430	34 /249	28 /207
		.80	26 /161	26 /398	18 /130	20 /144
		.90	15 /92	19 /269	19 /132	16 /108
(2) γPR+BFGS		.10	16 /97	17 /255	22 /146	16 /109
		.20	22 /136	25 /435	20 /147	19 /139
		.30	25 /157	21 /330	45 /333	17 /133
		.40	32 /198	30 /483	284 /2003	22 /175
		.50	32 /202	32 /509	44 /356	21 /171
		.60	21 /132	24 /342	38 /298	26 /216
		.70	31 /195	28 /401	42 /340	19 /159
		.80	41 /253	21 /372	23 /194	26 /221
		.90	32 /198	34 /533	34 /291	20 /181
(3) SD+PR+BFGS		.00	27 /184	22 /363	33 /306	25 /248
(4) SD+θPR+γBFGS		.10	11 /77	17 /266	39 /369	23 /222
		.20	19 /129	21 /362	267 /2405	29 /275
		.30	24 /158	19 /345	126 /1138	23 /219
		.40	25 /164	32 /551	53 /489	23 /214
		.50	23 /151	17 /284	66 /601	23 /212
		.60	19 /125	32 /539	39 /362	24 /221
		.70	21 /138	42 /663	60 /538	34 /312
		.80	17 /112	40 /721	47 /406	39 /351
		.90	68 /415	47 /877	50 /437	39 /351
SD	8	1.00	Diverge	173 /5491	Diverge	96 /1274
PR		.00	Diverge	13 /295	Diverge	59 /776
BFGS		1.00	502 /4691	7 /184	259 /4245	152 /1509
(1) θPR+γBFGS		.10	2639 /27733	15 /378	Diverge	27 /361
		.20	580 /6102	17 /468	1971 /25318	45 /590
		.30	509 /5306	13 /395	1820 /23264	35 /455
		.40	652 /6599	15 /361	1153 /14770	40 /515
		.50	687 /6907	89 /2590	892 /11220	42 /533
		.60	506 /5081	584 /15708	799 /9717	16 /210
		.70	276 /2797	420 /10712	615 /7354	45 /548
		.80	401 /4061	271 /6408	408 /5001	21 /261
		.90	263 /2569	125 /3356	550 /6110	147 /1647

continued on next page

continued from previous page

Directions	n	γ	Backtracking	Strong Wolfe	Wolfe	Armijo
			IT/FE	IT/FE	IT/FE	IT/FE
(2) γPR+BFGS		.10	266 /2632	12 /308	530 /6021	29 /340
		.20	278 /2853	229 /6020	484 /5765	31 /375
		.30	274 /2793	293 /8005	797 /9421	152 /1805
		.40	580 /5836	284 /7666	792 /9654	23 /295
		.50	562 /5781	36 /757	525 /6640	40 /523
		.60	226 /2387	56 /1526	743 /9428	44 /572
		.70	261 /2795	80 /2042	852 /10990	31 /412
		.80	402 /4247	41 /1093	797 /10421	18 /249
		.90	376 /3974	49 /1303	1169 /15449	28 /409
(3) SD+PR+BFGS		.00	942 /10371	946 /25920	1207 /17408	31 /469
(4) SD+θPR+γBFGS		.10	Diverge	18 /433	Diverge	42 /596
		.20	2023 /22231	16 /396	Diverge	37 /522
		.30	1770 /19473	18 /436	2445 /33924	23 /327
		.40	1608 /17465	24 /600	1937 /26746	36 /505
		.50	1190 /13068	28 /814	1703 /23429	48 /669
		.60	1241 /13459	43 /1182	1899 /26170	43 /598
		.70	1305 /13854	540 /14085	1396 /19221	61 /837
		.80	1052 /11150	60 /1778	1537 /21152	70 /955
		.90	983 /10465	106 /2949	1386 /19053	73 /991
SD	16	1.00	Diverge	Diverge	Diverge	Diverge
PR		.00	Diverge	158 /8066	Diverge	776 /17079
BFGS		1.00	1237 /21445	248 /13495	710 /22227	127 /2371
(1) θPR+γBFGS		.10	Diverge	2915 /144570	Diverge	1001 /22505
		.20	Diverge	2132 /110312	2857 /62753	404 /9120
		.30	Diverge	67 /3263	Diverge	56 /1192
		.40	1399 /25917	1905 /87373	Diverge	74 /1621
		.50	841 /15639	1195 /53253	2890 /62548	515 /11090
		.60	1289 /23874	1160 /55050	1471 /31305	57 /1256
		.70	460 /8693	569 /26031	1194 /25186	82 /1728
		.80	936 /16878	383 /18777	808 /16654	269 /5603
		.90	281 /5095	439 /21137	499 /10098	214 /4242
(2) γPR+BFGS		.10	535 /9727	425 /19987	745 /15149	193 /3802
		.20	812 /14663	468 /21378	567 /11779	234 /4921
		.30	644 /11944	678 /32779	1129 /23802	106 /2225
		.40	335 /6417	886 /42094	926 /19669	177 /3892
		.50	45 /865	742 /35730	1455 /31482	441 /9513
		.60	712 /13198	1211 /57778	2485 /54358	79 /1758
		.70	933 /17744	955 /43873	874 /19527	61 /1355
		.80	408 /7792	910 /43260	1514 /33583	77 /1774
		.90	1763 /33525	693 /32380	1776 /39670	125 /2852
(3) SD+PR+BFGS		.00	2923 /55566	2383 /124738	Diverge	673 /15831
(4) SD+θPR+γBFGS		.10	Diverge	Diverge	Diverge	1855 /43776
		.20	Diverge	Diverge	Diverge	820 /19261
		.30	Diverge	Diverge	Diverge	635 /14904

continued on next page

continued from previous page

Directions	n	γ	Backtracking	Strong Wolfe	Wolfe	Armijo
			IT/FE	IT/FE	IT/FE	IT/FE
		.40	Diverge	2634 /137245	2793 /64124	330 /7754
		.50	2723 /51769	2188 /115314	1072 /24593	106 /2439
		.60	Diverge	1187 /59308	Diverge	85 /1933
		.70	Diverge	759 /37302	Diverge	98 /2215
		.80	2017 /38354	1356 /64624	Diverge	196 /4418
		.90	1917 /36453	826 /39726	2494 /56399	317 /7177

Example 4.4. *Biggs EXP6 Function, (Moré, J.J., et al., 1981). The function f is given by*

$$f(x) = \sum_{i=1}^{m} f_i^2(x),\ m \geq n, n = 6,$$

where n is the number of variables and

$$f_i(x) = e^{-t_i x_1} - x_4 e^{-t_i x_2} + x_6 e^{-t_i x_5} - y_i,$$

where $t_i = (0.1)i$ and $y_i = e^{-t_i} - 5e^{-10t_i} + 3e^{-4t_i}$. The starting point is $x_0 = (1, 2, 1, 1, 1, 1)$. The numerical results are shown in Table 4.4.

Table 4.4. Results for the Biggs EXP6 Function

Directions	n	γ	Backtracking	Strong Wolfe	Wolfe	Armijo
			IT/FE	IT/FE	IT/FE	IT/FE
SD	6	1.00	Diverge	Diverge	Diverge	Diverge
PR		.00	Diverge	660 /14462	1036 /11030	Diverge
BFGS		1.00	51 /375	21 /557	25 /431	40 /306
(1) θPR+γBFGS		.10	287 /2197	536 /10286	598 /5481	346 /3244
		.20	185 /1424	315 /6534	409 /3778	131 /1200
		.30	236 /1890	254 /4660	328 /3077	83 /720
		.40	162 /1295	175 /3199	259 /2298	109 /958

continued on next page

continued from previous page

Directions	n	γ	Backtracking	Strong Wolfe	Wolfe	Armijo
			IT/FE	IT/FE	IT/FE	IT/FE
		.50	121 /964	112 /1971	151 /1318	109 /950
		.60	100 /796	97 /1893	126 /1090	74 /637
		.70	62 /481	66 /1204	110 /979	56 /465
		.80	50 /379	50 /904	72 /591	49 /400
		.90	41 /308	31 /680	71 /650	42 /329
(2) γPR+BFGS		.10	110 /785	32 /600	63 /543	41 /325
		.20	47 /357	37 /748	63 /556	49 /415
		.30	56 /439	55 /1182	96 /811	45 /373
		.40	63 /501	51 /886	79 /694	58 /512
		.50	82 /654	63 /1179	110 /978	72 /642
		.60	87 /695	87 /1727	212 /1912	61 /573
		.70	86 /693	92 /1982	148 /1359	57 /530
		.80	86 /690	95 /1953	137 /1332	72 /683
		.90	109 /880	94 /1816	138 /1313	96 /921
(3) SD+PR+BFGS		.00	110 /924	143 /3110	236 /2478	149 /1585
(4) SD+θPR+γBFGS		.10	757 /6091	1232 /26067	1409 /14418	690 /7237
		.20	469 /3780	567 /11833	670 /6814	369 /3833
		.30	521 /4179	392 /7960	433 /4385	240 /2476
		.40	396 /3182	274 /6341	356 /3628	138 /1429
		.50	319 /2566	248 /4880	314 /3172	99 /995
		.60	233 /1873	211 /4146	330 /3360	118 /1171
		.70	196 /1577	169 /3728	316 /3154	131 /1302
		.80	154 /1241	173 /3620	259 /2556	115 /1127
		.90	156 /1256	152 /3047	205 /2001	110 /1077

In Table 4.3, the interesting behaviour can be seen from the hybrid directions (1), (2) and (4). That is, the d^{SD} and d^{PR} when performed alone, cause divergence, but they are combined with d^{BFGS}, some reduction in the function evaluation occurs. This can be seen from the case $n = 4$ with $0.1d^{PR} + 0.9d^{BFGS}$ and $d^{SD} + 0.9d^{PR} + 0.1d^{BFGS}$ with backtracking technique. Similarly for the cases $n = 8$ and 16, $0.1d^{PR} + 0.9d^{BFGS}$ and $0.6d^{PR} + d^{BFGS}$ give some reduction in the number of function evaluations. The similar behaviour can also be seen in Table 4.4. The d^{SD} or d^{PR} cause divergence, but the hybrid direction can help reduce the number of function evaluations also. This can be seen from $0.1d^{PR} + 0.9d^{BFGS}$ with backtracking technique.

Case 3. The hybrid directions give worse performances than the performance based on the single direction. The divergence occurs when the test cases are taken from the Brown badly scaled function and the Brown and Dennis function, as shown in Examples 4.5 and 4.6. The coefficients of the variables of the functions used for these test cases are very much different in the magnitude. The numerical results are shown in Tables 4.5 and 4.6. They suggest that for cases in which the coefficients of the variables are too much different in magnitude the hybrid direction cannot handle the case successfully and some scaling has to be done.

Example 4.5. *Brown Badly Scaled Function, (Moré, J.J., et al., 1981). The function f is given by*

$$f(x) = (x_1 - 10^6)^2 + (x_2 - 2 \cdot 10^{-6})^2 + (x_1 x_2 - 2)^2.$$

The starting point is $x_0 = (1, 1)$. The numerical results are shown in Table 4.5.

Table 4.5. Results for the Brown Badly Scaled Function

Directions	n	γ	Backtracking	Strong Wolfe	Wolfe	Armijo
			IT/FE	IT/FE	IT/FE	IT/FE
SD	2	1.00	7 /83	Diverge	Diverge	Diverge
PR		.00	120 /1829	65 /2873	Diverge	Diverge
BFGS		1.00	13 /95	16 /390	12 /205	43 /416
(1) θPR+γBFGS		.10	Diverge	810 /38318	Diverge	Diverge
		.20	Diverge	Diverge	Diverge	Diverge
		.30	24 /386	Diverge	Diverge	Diverge
		.40	Diverge	Diverge	Diverge	Diverge
		.50	67 /1046	Diverge	Diverge	Diverge
		.60	Diverge	Diverge	Diverge	Diverge
		.70	13 /203	Diverge	Diverge	Diverge
		.80	Diverge	Diverge	Diverge	Diverge
		.90	Diverge	Diverge	Diverge	Diverge

continued on next page

continued from previous page

Directions	n	γ	Backtracking	Strong Wolfe	Wolfe	Armijo
			IT/FE	IT/FE	IT/FE	IT/FE
(2) γPR+BFGS		.10	Diverge	Diverge	Diverge	Diverge
		.20	2531 /45557	Diverge	Diverge	Diverge
		.30	934 /16410	Diverge	Diverge	Diverge
		.40	1163 /19106	Diverge	Diverge	Diverge
		.50	Diverge	Diverge	Diverge	Diverge
		.60	28 /447	Diverge	Diverge	Diverge
		.70	Diverge	Diverge	Diverge	Diverge
		.80	Diverge	Diverge	Diverge	Diverge
		.90	Diverge	Diverge	Diverge	Diverge
(3) SD+PR+BFGS		.00	15 /256	Diverge	Diverge	Diverge
(4) SD+θPR+γBFGS		.10	Diverge	2588 /119837	Diverge	Diverge
		.20	Diverge	Diverge	Diverge	Diverge
		.30	Diverge	Diverge	Diverge	Diverge
		.40	Diverge	Diverge	Diverge	Diverge
		.50	Diverge	Diverge	Diverge	Diverge
		.60	Diverge	Diverge	Diverge	Diverge
		.70	Diverge	Diverge	Diverge	Diverge
		.80	Diverge	Diverge	Diverge	Diverge
		.90	Diverge	Diverge	Diverge	Diverge

Example 4.6. *Brown and Dennis Function, (Moré, J.J., et al.,1981). In this case the function f is given by*

$$f(x) = \sum_{i=1}^{m} f_i^2(x),\ m \geq n,\ n = 4,$$

where n is the number of variables and

$$f_i(x) = (x_1 + t_i x_2 - e^{-t_i})^2 + (x_3 + x_4 \sin(t_i) - \cos(t_i))^2,$$

where $t_i = i/5$. The starting point is $x_0 = (25, 5, -5, -1)$. The numerical results are shown in Table 4.6.

Table 4.6. Results for the Brown and Dennis Function

Directions	n	γ	Backtracking IT/FE	Strong Wolfe IT/FE	Wolfe IT/FE	Armijo IT/FE
SD	4	1.00	380 /4472	Diverge	Diverge	495 /10242
PR		.00	50 /686	Diverge	Diverge	406 /8587
BFGS		1.00	20 /148	14 /223	16 /195	27 /272
(1) θPR+γBFGS		.10	109 /1378	Diverge	Diverge	305 /6407
		.20	54 /655	Diverge	Diverge	109 /2180
		.30	60 /706	Diverge	1881 /37722	101 /2047
		.40	85 /1014	Diverge	Diverge	185 /3667
		.50	92 /1141	Diverge	Diverge	317 /6259
		.60	115 /1375	Diverge	Diverge	219 /4347
		.70	91 /1090	Diverge	Diverge	215 /4084
		.80	418 /4604	Diverge	Diverge	136 /2507
		.90	332 /3396	Diverge	Diverge	229 /4042
(2) γPR+BFGS		.10	272 /2652	Diverge	Diverge	173 /2992
		.20	229 /2247	Diverge	Diverge	253 /4851
		.30	147 /1689	Diverge	Diverge	278 /5342
		.40	82 /1052	Diverge	Diverge	243 /4693
		.50	110 /1356	Diverge	Diverge	886 /17615
		.60	79 /973	Diverge	Diverge	685 /13783
		.70	77 /958	Diverge	1665 /33402	50 /1002
		.80	49 /626	Diverge	Diverge	333 /6924
		.90	98 /1288	Diverge	Diverge	122 /2516
(3) SD+PR+BFGS		.00	103 /1269	Diverge	Diverge	145 /3157
(4) SD+θPR+γBFGS		.10	110 /1290	Diverge	Diverge	138 /3009
		.20	109 /1199	Diverge	Diverge	568 /12356
		.30	173 /1976	Diverge	Diverge	514 /11058
		.40	319 /3509	Diverge	Diverge	108 /2279
		.50	154 /1760	Diverge	Diverge	165 /3526
		.60	268 /2940	Diverge	Diverge	101 /2118
		.70	431 /4845	Diverge	Diverge	544 /11450
		.80	464 /5236	Diverge	Diverge	1320 /27522
		.90	386 /4411	Diverge	Diverge	363 /7543

4.3 Discussion

From the implementation of Algorithm 3.2, using the four choices of hybrid directions,

(1) $(1-\gamma)d^{PR} + \gamma d^{BFGS}, \quad \gamma = 0, 0.1, \ldots, 1,$

(2) $\gamma d^{PR} + d^{BFGS}, \quad \gamma = 0, 0.1, \ldots, 1,$

(3) $d^{SD} + d^{PR} + d^{BFGS}$,

(4) $d^{SD} + (1-\gamma)d^{PR} + \gamma d^{BFGS}, \quad \gamma = 0, 0.1, \ldots, 1$,

the following points can be made corresponding to the aims stated in Section 4.1.

1. For this preliminary investigation, the hybrid directions, in particular, the combinations between the conjugate gradient and BFGS directions (Hybrid direction(1)) show some trends for the possibility of speeding up the process of locating the minimizer, in comparison to the search based on a single direction.

2. The backtracking technique shows numerically to be the suitable and efficient way in obtaining the admissible step length along the hybrid directions.

3. The hybrid direction (1) gives the better performances over all especially when it is implemented with the backtracking technique and as the dimension of the problem is higher, the reduction in the number of function evaluations becomes more evident.

Chapter V

Conclusion

This thesis presents an approach for solving an unconstrained minimization problem, min $f(x)$, $f : \mathbb{R}^n \to \mathbb{R}$. The approach is based on the theoretical results on the expanding subspace property of the conjugate gradient method for the convex quadratic function and also the idea of locating a minimizer on the linear variety. The investigation utilizes the line search framework, i.e., for any given $x_0 \in \mathbb{R}^n$ the sequence of estimates of the minimizer of f, $\{x_k\}$ has the form

$$x_{k+1} = x_k + \lambda_k d_k,$$

where d_k is a descent direction of f at x_k and λ_k is an admissible step length along d_k. The approach in constructing the search direction in this thesis is to take a linear combination of some independent search directions. The line search is then carried out along this combined direction. The aim here is to be able to locate a minimizer in a larger region, in particular, on a linear variety, $x_0 + V$, where V is a subspace spanned by the independent search directions in the linear combination.

The preliminary choices of the combined directions in this thesis are limited to the combination of the existing and well-known directions. The two main directions used are the BFGS quasi-Newton and Polak-Ribière conjugate gradient directions. The steepest descent direction is also combined with these two directions to observe the behaviour. Some linear combinations of these directions, or the hybrid directions, are tested on some standard test problems of

Moré, J.J. *et al.* (1981).

The relative numerical results show some promising trends of the hybrid directions in speeding up the process of locating the minimizer. However, this preliminary investigation is limited to choices of the search directions and the scalar multiples in the linear combination. This suggests further investigation and development of a good representative direction in the subspace.

Finally, the approach developed in this thesis can be used as the basis for establishing a parallel numerical method for solving an unconstrained minimization problem, as the search directions can be independently constructed.

References

References

Buckley, A. G. (1978). A combined conjugate-gradient quasi-Newton minimization algorithm. **Math. Programming** 15: 200-210.

Buckley, A. G. (1978). Extending the relationship between the conjugate gradient and BFGS algorithms. **Math. Programming** 15: 343-348.

Byrd, R. H., Nocedal, J., and Yuan, Y-X. (1987). Global convergence of a class of quasi-Newton methods on convex problems. **SIAM J. Numer. Anal.** 24 (5): 1171-1190.

Caprioli, P. and Holmes, M. H. (1998). A parallel quasi-Newton method for Gaussian data fitting. **Parallel Comput**. 24: 1635-1651.

Chen, Z., Fei, P. and Zheng, H. (1995). A parallel quasi-Newton algorithm for unconstrained optimization. **Comput. J.** 55: 125-133.

Conn, A. R., Gould, N. I. M. and Toint, Ph. L. (1991). Convergence of quasi-Newton matrices generated by the symmetric rank one update. **Math. Programming** 50: 177-195.

Dai, Y. H., Han, J., Liu, G., Sun, D., Yin, H. and Yuan, Y-X. (1999). Convergence properties of nonlinear conjugate gradient methods. **SIAM. J. Optimization.** 10 (2): 345-358.

Dai, Y. H. and Yuan, Y-X. (2002). A note on the nonlinear conjugate gradient method. **J. Comp. Math.** 20 (6).

Dennis, J. E., JR., and Moré, J. J. (1974). A characterization of superlinear convergence and its application to quasi-Newton methods. **Math. Comp.** 28(126): 549-560.

Dennis, J. E., JR., and Schnabel, R. B. (1983). **Numerical methods for unconstrained optimization and nonlinear equations**. Prentice-Hall, Inc., Englewood Clifts, NJ.

Fletcher, R. (1987). **Practical methods of optimization**, second edition. John Wiley & Sons, Chichester.

Grapsa, T. N. and Vrahatis, N. M. (1996). A dimension-reducing method for unconstrained optimization. **J. Comp. Appl. Math.** 66: 239-253.

Grippo, L. and Lucidi, S. (1997). A global convergent version of the Polak-Ribière conjugate gradient method. **Math. Programming.** 78 : 375-391.

Kelley, C. T. (1999). **Iterative methods for optimization**. SIAM, Philadelphia.

Li, D-H. and Fukushima, M. (2001). A modified BFGS method and its global convergence in nonconvex minimization. **J. Comp. Appl. Math.** 129: 15-35.

Liao, A. (1997). Modifying the BFGS method. **Operations research letters**. 20: 171-177.

Luenberger, D. G. (1984). **Linear and nonlinear programming**, second edition. Addison-Wesley Publishing Company, Reading, Massachusetts.

Lukšan, L. (1989). Computational experience with improved variable metric methods for unconstrained minimization. **Technical report**. No. V-451.

Lukšan, L. (1991). Computational experience with improved conjugate gradient methods for unconstrained minimization. **Technical report**. No. V-488.

Moré, J. J., Garbow, B.S., and Hillstrom, K .E. (1981). Testing unconstrained optimization software. **ACM Trans. Math. Software** 7(1): 17-41.

Nazareth, L. (1979). A relationship between the BFGS and conjugate gradient algorithms and its implications for new algorithms. **SIAM J. Numer. Anal.** 16 (5): 794-800.

Nocedal, J., and Wright, S. J. (1999). **Numerical optimization**. Springer-Verlag New York, Inc., New York.

Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P. (1986-1992). **Numerical recipes in Fortran 77**: The art of scientific computing, second edition. Vol. 1 of fortran numerical recipes. Cambridge University Press. The United states of America.

Vlček, J. and Lukšan, L. (2002). New variable metric methods for unconstrained minimization covering the large-scale case. **Technical report**. No. V 876.

Vrahatis, M. N., Androulakis, G. S., and Manoussakis, G. E. (1996). A new unconstrained optimization method for imprecise function and gradient values. **J. Math. Anal. appl.** 197(41): 586-607.

Vrahatis, M. N., Androulakis, G. S., Lambrinos, J. N., and Magoulas, G. D. (2000). A class of gradient unconstrained minimization algorithms with adaptive stepsize. **J. Comp. Appl. Math.** 114: 367-386.

Appendix

Appendix A

Terminology

A.1 Types of Solution

Definition A.1. *Let* Ω *be a subset of* $\mathbb{R}^n$*. A point* x^* *is said to be a local minimizer of* f *on* Ω *if there is a neighbourhood* $\mathcal{N}(x^*) \subset \Omega$ *such that*

$$f(x) \geq f(x^*) \;\; for\ all \;\; x \in \mathcal{N}(x^*).$$

Definition A.2. *Let* Ω *be subset of* $\mathbb{R}^n$*. A point* x^* *is said to be a global minimizer of* f *over* Ω *if*

$$f(x) \geq f(x^*) \;\; for\ all \;\; x \in \Omega.$$

A.2 Necessary Conditions

Theorem A.1. *(First-Order Necessary Condition)*

Let x^* *is a local minimizer and* f *is continuously differentiable on an open neighbourhood* $\mathcal{N}(x^*)$*, then* $\nabla f(x^*) = 0$*.*

Theorem A.2. *(Second-Order Necessary Conditions)*

Let x^* *is a local minimizer and* f *is twice continuously differentiable on open neighbourhood* $\mathcal{N}(x^*)$*, then*

1.) $\nabla f(x^*) = 0$,

2.) $\nabla^2 f(x^*) \geq 0$ *(positive semidefinite).*

Theorem A.3. *(Second -Order Sufficient Conditions)*

Let f be twice continuously differentiable on an open neighbourhood $\mathcal{N}(x^*)$. If

1.) $\nabla f(x^*) = 0$,

2.) $\nabla^2 f(x^*) > 0$ *(positive definite).*

Then x^* is a strict local minimizer of f.

A.3 Convex Functions

Definition A.3. *A function f defined on a convex set Ω is said to be convex if, for every $x_1, x_2 \in \Omega$ and every $\theta \in [0,1]$ there holds*

$$f((1-\theta)x_1 + \theta x_2) \leq (1-\theta)f(x_1) + \theta f(x_2).$$

If, for every $\theta \in (0,1)$, and $x_1 \neq x_2$ there holds

$$f((1-\theta)x_1 + \theta x_2) < (1-\theta)f(x_1) + \theta f(x_2),$$

then f is said to be strictly convex.

Theorem A.4.

Let f and g be convex functions on the convex set Ω. Then the function

1.) θf, *for all* $\theta \geq 0$

2.) $f + g$.

are convex on Ω.

Theorem A.5.

Let f be a convex function on a convex set Ω. Then the set $\mathcal{L} = \{x \big| x \in \Omega, f(x) \leq c, c \in \mathbb{R}\}$ is convex.

Theorem A.6.

Let f be continuously differentiable and convex on the convex set Ω. If there is a point $x^* \in \Omega$ such that, for all $y \in \Omega, \nabla f(x^*)^T(y - x^*) \geq 0$, then x^* is a global minimizer.

A.4 Types of Convergence

1) **Convergence of the sequence $\{x_k\}$ in $\mathbb{R}^n$**

Let $x^* \in \mathbb{R}^n, x_k \in \mathbb{R}^n, k = 0, 1, 2, \ldots$. Then the sequence $\{x_k\} = \{x_1, x_2, x_3, \ldots\}$ is said to *converge* to x^* if

$$\lim_{k\to\infty} \|x_k - x^*\| = 0.$$

2) **Q-linear convergence**

The sequence $\{x_k\} = \{x_1, x_2, x_3, \ldots\}$ is said to be *q-linearly convergent* to x^* if there exists a constant $c \in [0, 1)$ and an integer $\hat{k} \geq 0$ such that for all $k \geq \hat{k}$,

$$\|x_{k+1} - x^*\| \leq c\|x_k - x^*\|.$$

3) **Q-superlinear convergence**

The sequence $\{x_k\} = \{x_1, x_2, x_3, \ldots\}$ is said to be *q-superlinearly convergent* to x^* if for some sequence $\{c_k\}$ that converges to 0,

$$\|x_{k+1} - x^*\| \leq c_k\|x_k - x^*\|.$$

4) **Q-quadratic convergence**

The sequence $\{x_k\} = \{x_1, x_2, x_3, \ldots\}$ is said to be *q-quadratically convergent* to x^* if there exist constants $c \geq 0$ and $\hat{k} \geq 0$ such that for all $k \geq \hat{k}$,

$$\|x_{k+1} - x^*\| \leq c\|x_k - x^*\|^2.$$

A.5 Sherman-Morrison-Woodbury formula

If the square nonsingular matrix A in $\mathbb{R}^{n\times n}$ is updated in the following form

$$\bar{A} = A + ab^T,$$

where a and b are vectors in $\mathbb{R}^n$, if $\bar{A}$ is nonsingular, then

$$\bar{A}^{-1} = A^{-1} - \frac{A^{-1}ab^TA^{-1}}{1 + b^TA^{-1}a}. \tag{A.1}$$

This formula can be extended to higher rank updates. Let U and V be matrices in $\mathbb{R}^{n\times p}$ for some p between 1 and n. If

$$\hat{A} = A + UV^T,$$

and $\hat{A}$ is nonsingular, then

$$\hat{A}^{-1} = A^{-1} - A^{-1}U(I + V^TA^{-1}U)^{-1}V^TA^{-1}. \tag{A.2}$$

Appendix B

Fortran Program

The FORTRAN codes of Algorithm 3.2 are presented in this section. The descriptions of the subroutines used in this program are as follows.

- The DABS, DDOT, DFLOAT and DNRM2 subroutines compute the absolute values, dot product and Euclidean norm. They are called from the Sun Performance Library Reference, Basic Linear Algebra Subprograms, Level 1 (BLAS1).
- The INITPT, OBJFCN and GRDFCN subroutines compute the initial point, objective function and its gradient from the standard test problems of Moré, J.J., *et al.* (1981).
- The EXPSLNS, LNSRCH, LINESRCH and LINESRCH1 subroutines compute the step length λ according to the Armijo's conditions, backtracking technique, strong Wolfe and Wolfe conditions, respectively.
- The NCG and BFGS subroutines compute the search direction in the form (2.45) based on the PR choice and (2.31) based on the BFGS update, respectively.

FORTRAN CODES OF ALGORITHM 3.2

```
! THIS PROGRAM IS FOR FINDING THE MINIMIZER OF A GIVEN OBJECTIVE FUNCTION OF
! N VARIABLES.  THE METHOD USED IS TO COMBINE THE QUASI-NEWTON DIRECTIONS(BFGS)
! AND CONJUGATE GRADIENT (POLAK-RIBIERE) DIRECTIONS AND FIT THIS COMBINED DIRECTION
! INTO THE LINE SEARCH FRAMEWORK.
! N : NUMBER OF VARIABLES(DIMENSION)
! ITS : NUMBER OF ITERATIONS,
! ITMAX:MAXIMUM OF ITERATIONS ALLOWED
! FE : TOTAL NUMBER OF FUNCTION EVALUATIONS AND COMPONENT OF THE GRADIENT
! XOLD,X : CURRENT AND NEW ITERATES
! GOLD, D :CURRENT AND NEW GRADIENTS
! DOLD,D : CURRENT AND NEW DIRECTIONS
! DCG :  CONJUGATE GRADIENT DIRECTIONS(POLAK RIBIERE)
! DBFGS : QUASI-NEWTON DIRECTIONS USING THE BFGS UPDATE
! DSD :  STEEPEST DESCENT DIRECTIONS
! DCOMB : HYBRID DIRECTIONS
```

```
! FXO,FX : FUNCTION VALUES AT CURRENT AND NEW ITERATES
! GAMMA : SCALAR MULTIPLES DIRECTIONS OF THE DIRECTIONS IN THE HYBRID DIRECTIONS.
! VALUES OF GAMMA ARE BETWEEN 0 AND 1
! a = 1- GAMMA, b = GAMMA
! NRM* : NORM OF ANY VECTOR *
! H : INVERSE HESSIAN APPROXIMATIONS
! S : THE VARIABLE WHICH CONTAINING THE DIFFERENCE BETWEEN THE NEW AND CURRENT ITERATES
! YY,SS : DOT PRODUCT BETWEEN TWO VECTORS Y AND S RESPECTIVELY.
! NLNS : CHOICES OF LINE SEARCH    1 BACKTRACKING LINE SEARCH (LNSRCH SUBROUTINE)
!                                  2 STRONG WOLFE CONDITIONS  (LINESRCH SUBROUTINE)
!                                  3 WOLFE CONDITIONS (LINESRCH1 SUBROUTINE)
!                                  4 EXPONENTIAL SCHEDULE LINE SEARCH (EXPSLNS SUBROUTINE)
!
! NPROB : PROBLEM NUMBER
!  1 HELICAL VALLEY FUNCTION(3)        10 BROWN BADLY SCALED FUNCTION(2)
!  2 BIGGS EXP6 FUNCTION(6)            11 BROWN AND DENNIS FUNCTION(4)
!  3 GAUSSIAN FUNCTION(3)              12 GULF RESEARCH AND DEVELOPMENT FUNCTION(3)
!  4 POWELL BADLY SCALED FUNCTION(2)   13 TRIGONOMETRIC FUNCTION
!  5 BOX 3-DIMENSIONAL FUNCTION(3)     14 EXTENDED ROSENBROCK FUNCTION
!  6 VARIABLY DIMENSIONED FUNCTION     15 EXTENDED POWELL SINGULAR FUNCTION
!  7 WATSON FUNCTION                   16 BEALE FUNCTION(2)
!  8 PENALTY FUNCTION I                17 WOOD FUNCTION(4)
!  9 PENALTY FUNCTION II               18 CHEBYQUAD FUNCTION
!
! NCOMB : COMBINATION NUMBER    1  aPR + bBFGS
!                               2  aPR + bSD
!                               3  bPR +BFGS
!                               4  SD + PR + BFGS
!                               5  SD + aPR + bBFGS
PROGRAM HYBRIDDIRECT
USE MSFLIB
IMPLICIT NONE
INTEGER    ::  N,NMAX,I,ITS,ITMAX,MAXFE,OUT,OUT1,LDH,FAIL,FE,GE,NPROB,NLNS,T
INTEGER(2)::  IDIREC,ILINE,NDIREC,NLINE,ALLN,NCOMB,IRET*4,SIG*4
INTEGER(2)::  STATUS,CONTROL,LENGTH,RETCODE
RECORD  /MTH$E_INFO/ INFO
CHARACTER*4 :: NAME
PARAMETER (NMAX = 200)
PARAMETER (LDH = NMAX)
PARAMETER (OUT = 13)
PARAMETER (OUT1= 14)
PARAMETER (ITMAX = 3000,MAXFE=90000)
PARAMETER (NDIREC= 31)
PARAMETER (NLINE = 4)
INTEGER  :: IT(NDIREC,NLINE),F(NDIREC,NLINE)
DOUBLE PRECISION :: EPS,TOLX,ALPHA,GAMMA,ZERO,ONE,FACTOR,DCOMB(NMAX)
DOUBLE PRECISION :: FX,FXO,NORMX,NRMG,DDOT,DNRM2,SLOPE,DOLD(NMAX)
DOUBLE PRECISION :: H(LDH,NMAX),GOLD(NMAX),G(NMAX),XOLD(NMAX),X(NMAX)
DOUBLE PRECISION :: D(NMAX),DCG(NMAX),DBFGS(NMAX),DX(NMAX),DSD(NMAX)
PARAMETER (TOLX=1.0D-10,EPS=1.0D-05)
PARAMETER (NPROB =  13)
EXTERNAL  INITPT
EXTERNAL  OBJFCN
EXTERNAL  GRDFCN
EXTERNAL  NCG
EXTERNAL  BFGS
EXTERNAL  LNSRCH
EXTERNAL  LINESRCH
EXTERNAL  LINESRCH1
EXTERNAL  EXPSLNS
EXTERNAL  DNRM2
EXTERNAL  DDOT
INTRINSIC DABS
INTRINSIC DFLOAT
DATA ZERO,ONE,FACTOR  /0.0D0,1.0D0,1.0D0/
!
INTERFACE
    FUNCTION  HAND_FPE (SIGID, EXCEPT)
       !MS$ATTRIBUTES C :: HAND_FPE
       INTEGER*4 HAND_FPE
       INTEGER*2  SIGID, EXCEPT
    END FUNCTION
END INTERFACE
!
OPEN(13,FILE='allresult.dat')
OPEN(14,FILE='output_x.dat')
!
```

```
WRITE(13,*)
    GO TO (401,402,403,404,405,406,407,408,409,410,411,412,&
    & 413,414,415,416,417,418), NPROB
 401 CONTINUE
       WRITE(13,*)'Results for the Helical Valley function'
       GOTO 419
 402 CONTINUE
       WRITE(13,*)'Results for the Biggs EXP6 function'
       GOTO 419
 403 CONTINUE
       WRITE(13,*)'Results for the Gaussian function'
       GOTO 419
 404 CONTINUE
       WRITE(13,*)'Results for the Powell Badly Scaled function'
       GOTO 419
 405 CONTINUE
       WRITE(13,*)'Results for the Box 3-dimensional function'
       GOTO 419
 406 CONTINUE
       WRITE(13,*)'Results for the Variably Dimensioned function'
       GOTO 419
 407 CONTINUE
       WRITE(13,*)'Results for the Watson function'
       GOTO 419
 408 CONTINUE
       WRITE(13,*)'Results for the Penalty function I'
       GOTO 419
 409 CONTINUE
       WRITE(13,*)'Results for the Penalty function II'
       GOTO 419
 410 CONTINUE
       WRITE(13,*)'Results for the Brown Badly Scaled function'
       GOTO 419
 411 CONTINUE
       WRITE(13,*)'Results for the Brown and Dennis function'
       GOTO 419
 412 CONTINUE
       WRITE(13,*)'Results for the Gulf Research and Development function'
       GOTO 419
 413 CONTINUE
       WRITE(13,*)'Results for the Trigonometric function'
       GOTO 419
 414 CONTINUE
       WRITE(13,*)'Results for the Extended Rosenbrock function'
       GOTO 419
 415 CONTINUE
       WRITE(13,*)'Results for the Extended Powell Sigular function'
       GOTO 419
 416 CONTINUE
       WRITE(13,*)'Results for the Beale function'
       GOTO 419
 417 CONTINUE
       WRITE(13,*)'Results for the Wood function'
       GOTO 419
 418 CONTINUE
       WRITE(13,*)'Results for the Chebyquad function'
 419 CONTINUE
!
WRITE(13,*)
WRITE(13,203)'====================================================&
                  &======================================'
WRITE(13,200)'Backtracking','Strong Wolfe','Wolfe','Armijo'
WRITE(13,201)'Directions','n','gamma'
WRITE(13,202)'IT','FE','IT','FE','IT','FE','IT','FE'
WRITE(13,203)'====================================================&
                  &======================================'
WRITE(13,*)
!
! DIMENSION LOOP
!
     ALLN = 2
     T=0
  10 CONTINUE
     T = T+1
     ALLN = 2*ALLN
     IF(NPROB == 4 .OR. NPROB == 10 .OR. NPROB == 16)THEN
        ALLN = 2
```

```
      ELSE IF(NPROB == 1 .OR. NPROB == 3 .OR. NPROB == 5 .OR. NPROB == 12)THEN
         ALLN = 3
      ELSE IF(NPROB == 11 .OR. NPROB == 17)THEN
         ALLN = 4
      ELSE IF(NPROB == 2)THEN
         ALLN = 6
      ELSE
      ENDIF
!
      IF(NPROB == 7)THEN
        IF(ALLN > 16)ALLN = 30
      ENDIF
!
! DIRECTION LOOP
!
      N = ALLN
      IDIREC = 0
      ALPHA = ZERO
   20 CONTINUE
      IDIREC =IDIREC + 1
      IF(IDIREC == 1)THEN
         NCOMB = 2
         GAMMA = ONE
      ELSE IF (IDIREC == 2)THEN
         NCOMB = 1
         GAMMA = ZERO
      ELSE IF (IDIREC == 3)THEN
         GAMMA = 1
      ELSE
         IF(IDIREC == 4)THEN
            GAMMA = 0.1
         ELSE
            GAMMA = GAMMA + 0.1
            IF(IDIREC == 13)THEN
               NCOMB = 3
               GAMMA = 0.1
            ENDIF
            IF(IDIREC == 22)THEN
               GAMMA = ZERO
               NCOMB = 4
            ENDIF
            IF(IDIREC == 23)THEN
               NCOMB = 5
               GAMMA = 0.1
            ENDIF
         ENDIF
       ENDIF
!
! LINE SEARCH LOOP
!
     DO ILINE = 1,4,1
       NLNS = ILINE
!
! INITIAL DATA
!
      ITS= 0
      FE = 0
      GE = 0
      CALL INITPT(N,X,NPROB,FACTOR)
      CALL OBJFCN(N,X,FX,NPROB)
      CALL GRDFCN(N,X,G,NPROB)
      FX0= FX
      FE = FE + 1
      GE = GE + N
      DO I = 1,N
        D(I) = -G(I)
        XOLD(I)= X(I)
        GOLD(I)= G(I)
      ENDDO
!
! CHECK NORM OF GRADIENT G(I)
      NRMG = DNRM2(N,G,1)
      IF(NRMG <= EPS) GOTO 30
!
! MAIN LOOP
!
     DO ITS = 1,ITMAX,1
```

```
!
! THE NEW FUNCTION EVALUATION OCCURS IN LINE SEARCH SUBROUTINE; SAVE THE FUNCTION
! VALUE IN FX FOR THE NEXT LINE SEARCH.
!
       SLOPE = DDOT(N,G,1,D,1)
       IF(SLOPE > ZERO)THEN
          DO I = 1,N
             D(I)=-G(I)
          ENDDO
          SLOPE = DDOT(N,G,1,D,1)
       ENDIF
!
!  SELECTION OF CONDITION ON THE SCALARS ALONG THE SEARCH DIRECTION
!
       IF(NLNS >= 1 .AND. NLNS <= 4)THEN
          GOTO(1010,1020,1030,1040),NLNS
       ELSE
          WRITE(*,*) 'CHOOSE SELECTION OF THE CONDITIONS ON THE SCALARS &
                     &ALONG THE SEARCH DIRECTION 1-4'
          WRITE(*,*)
       STOP
       ENDIF
1010 CONTINUE
       CALL LNSRCH(N,X,FX,D,G,SLOPE,FE,GE,NPROB,FAIL)
       GOTO 1000
1020 CONTINUE
       CALL LINESRCH(N,X,FX,D,G,SLOPE,FE,GE,NPROB,FAIL)
       GOTO 1000
1030 CONTINUE
       CALL LINESRCH1(N,X,FX,D,G,SLOPE,FE,GE,NPROB,FAIL)
       GOTO 1000
1040 CONTINUE
       CALL EXPSLNS(N,X,FX,D,G,SLOPE,NPROB,FE,GE,FAIL)
1000 CONTINUE
       DX(I) = ZERO
       DO I = 1,N
          DX(I)= X(I) - XOLD(I)
       ENDDO
       NORMX = DNRM2(N,DX,1)
       NRMG = DNRM2(N,G,1)
       CALL MATHERRQQ( NAME, LENGTH, INFO, RETCODE)
       IRET = SIGNALQQ(SIG, HAND_FPE)
       CALL GETCONTROLFPQQ (CONTROL)
!
!IF NOT ROUNDING DOWN
!
       IF(IAND(CONTROL, FPCW$DOWN) /= FPCW$DOWN) THEN
          CONTROL = IAND(CONTROL, NOT(FPCW$MCW_RC))     ! CLEAR ALL
                                                        ! ROUNDING
          CONTROL = IOR(CONTROL, FPCW$DOWN)             ! SET TO
                                                        ! ROUND DOWN
          CALL SETCONTROLFPQQ(CONTROL)
       ENDIF
       CALL GETSTATUSFPQQ(STATUS)
!
! CHECK FOR DIVISION BY ZERO
!
       IF(IAND(STATUS, FPSW$INVALID) /= 0) THEN
          WRITE (*,*) 'INVALID.  LOOK                               &
                      &FOR NAN OR SIGNED INFINITY IN RESULTANT DATA.'
       ENDIF
       IF(IRET /= -1)THEN
          WRITE(*,*) 'SET EXCEPTION HANDLER.  RETURN = ', IRET
          GOTO 30
       ENDIF
       IF(NORMX <= TOLX)THEN
          WRITE(*,*)N,' ',ITS,' ',NLNS,' ',NCOMB
          WRITE(*,*)'||X_(k+1)-X_k|| =',NORMX
          GOTO 30
       ENDIF
       IF(NRMG <= EPS)GOTO 30
       DO I = 1,N
          DOLD(I)  = D(I)
          DBFGS(I) = D(I)
          DCG(I)   = D(I)
       ENDDO
!
```

```
! TEST FOR ALL CHOICES OF COMBINED DIRECTIONS
!
       IF(NCOMB >= 1 .AND. NCOMB <= 5)THEN
          GOTO(1001,1002,1003,1004,1005),NCOMB
       ELSE
          WRITE(*,*) ' CHOOSE CHOICES FO FOR ALL COMBINED DIRECTIONS 1-5'
          WRITE(*,*)
       STOP
       ENDIF
1001 CONTINUE
       IF(GAMMA == ZERO)THEN
          CALL NCG(N,G,GOLD,DCG)
       ELSEIF(GAMMA == ONE)THEN
          CALL BFGS(N,ITS,LDH,H,X,XOLD,G,GOLD,DBFGS)
       ELSE
          CALL BFGS(N,ITS,LDH,H,X,XOLD,G,GOLD,DBFGS)
          CALL NCG(N,G,GOLD,DCG)
       ENDIF
       DO I = 1,N
          DCOMB(I)=(ONE-GAMMA)*DCG(I) + GAMMA*DBFGS(I)
       ENDDO
       GOTO 2000
1002 CONTINUE
       IF(GAMMA == ZERO)THEN
          CALL NCG(N,G,GOLD,DCG)
       ELSEIF(GAMMA == ONE)THEN
          DO I = 1,N
             DSD(I)=-G(I)
          ENDDO
       ELSE
          DO I = 1,N
             DSD(I)=-G(I)
          ENDDO
          CALL NCG(N,G,GOLD,DCG)
       ENDIF
       DO I = 1,N
          DCOMB(I)=(ONE-GAMMA)*DCG(I) + GAMMA*DSD(I)
       ENDDO
       GOTO 2000
1003 CONTINUE
       CALL BFGS(N,ITS,LDH,H,X,XOLD,G,GOLD,DBFGS)
       CALL NCG(N,G,GOLD,DCG)
       DO I = 1,N
          DCOMB(I)= GAMMA*DCG(I) + DBFGS(I)
       ENDDO
       GOTO 2000
1004 CONTINUE
       CALL BFGS(N,ITS,LDH,H,X,XOLD,G,GOLD,DBFGS)
       CALL NCG(N,G,GOLD,DCG)
       DO I = 1,N
          DSD(I)=-G(I)
       ENDDO
       DO I = 1,N
          DCOMB(I)= DSD(I) + DCG(I) + DBFGS(I)
       ENDDO
       GOTO 2000
1005 CONTINUE
       IF(GAMMA == ZERO)THEN
          CALL NCG(N,G,GOLD,DCG)
       ELSE IF(GAMMA == ONE)THEN
          CALL BFGS(N,ITS,LDH,H,X,XOLD,G,GOLD,DBFGS)
       ELSE
          CALL BFGS(N,ITS,LDH,H,X,XOLD,G,GOLD,DBFGS)
          CALL NCG(N,G,GOLD,DCG)
       ENDIF
       DO I = 1,N
          DSD(I)=-G(I)
       ENDDO
       DO I = 1,N
          DCOMB(I) = DSD(I) + (ONE-GAMMA)*DCG(I) + GAMMA*DBFGS(I)
       ENDDO
!
! UPDATE NEW DIRECTION
!
2000 CONTINUE
       DO I = 1,N
          D(I) = DCOMB(I)
```

```
          XOLD(I) = X(I)
          GOLD(I) = G(I)
       ENDDO
       FX0 = FX
     ENDDO
!
!  END MAIN LOOP
!
  30 CONTINUE
     WRITE(14,204)NLNS,GAMMA,ITS,(X(I),I=1,N),NRMG,FX
!
     IF(ITS > ITMAX .OR. F(IDIREC,ILINE) > MAXFE)THEN
        IT(IDIREC,ILINE)=  'Div'
        F(IDIREC,ILINE) =  'Div'
     ELSE IF(IRET /= -1)THEN
        IT(IDIREC,ILINE)= ICHAR('0')
        F(IDIREC,ILINE) = ICHAR('0')
        WRITE(13,*)
        WRITE(13,205)GAMMA,IT(IDIREC,1),F(IDIREC,1),IT(IDIREC,2),F(IDIREC,2),  &
                     & IT(IDIREC,3),F(IDIREC,3),IT(IDIREC,4),F(IDIREC,4)
        GOTO 113
     ELSE
        IT(IDIREC,ILINE) = ITS
        F(IDIREC,ILINE) = FE + GE
     ENDIF
ENDDO
!
! END LINE SEARCH LOOP
!
     IF(IDIREC <= 4)THEN
        GOTO(100,102,104,106),IDIREC
 100 CONTINUE
        WRITE(13,*)
        WRITE(13,101)ALLN,GAMMA,IT(IDIREC,1),F(IDIREC,1),IT(IDIREC,2),F(IDIREC,2), &
                     & IT(IDIREC,3),F(IDIREC,3),IT(IDIREC,4),F(IDIREC,4)
        GOTO 113
 102 CONTINUE
        WRITE(13,*)
        WRITE(13,103)GAMMA,IT(IDIREC,1),F(IDIREC,1),IT(IDIREC,2),F(IDIREC,2),  &
                     & IT(IDIREC,3),F(IDIREC,3),IT(IDIREC,4),F(IDIREC,4)
        GOTO 113
 104 CONTINUE
        WRITE(13,*)
        WRITE(13,105)GAMMA,IT(IDIREC,1),F(IDIREC,1),IT(IDIREC,2),F(IDIREC,2),  &
                     & IT(IDIREC,3),F(IDIREC,3),IT(IDIREC,4),F(IDIREC,4)
        GOTO 113
 106 CONTINUE
        WRITE(13,*)
        WRITE(13,107)GAMMA,IT(IDIREC,1),F(IDIREC,1),IT(IDIREC,2),F(IDIREC,2),  &
                     & IT(IDIREC,3),F(IDIREC,3),IT(IDIREC,4),F(IDIREC,4)
        GOTO 113
     ELSE
        IF(IDIREC == 13)THEN
           WRITE(13,*)
           WRITE(13,108)GAMMA,IT(IDIREC,1),F(IDIREC,1),IT(IDIREC,2),F(IDIREC,2),  &
                        & IT(IDIREC,3),F(IDIREC,3),IT(IDIREC,4),F(IDIREC,4)
           GOTO 113
        ELSE IF(IDIREC == 22)THEN
           WRITE(13,*)
           WRITE(13,109)GAMMA,IT(IDIREC,1),F(IDIREC,1),IT(IDIREC,2),F(IDIREC,2),  &
                        & IT(IDIREC,3),F(IDIREC,3),IT(IDIREC,4),F(IDIREC,4)
           GOTO 113
        ELSE IF(IDIREC == 23)THEN
           WRITE(13,*)
           WRITE(13,111)GAMMA,IT(IDIREC,1),F(IDIREC,1),IT(IDIREC,2),F(IDIREC,2),  &
                        & IT(IDIREC,3),F(IDIREC,3),IT(IDIREC,4),F(IDIREC,4)
           GOTO 113
        ELSE
        ENDIF
        WRITE(13,112)GAMMA,IT(IDIREC,1),F(IDIREC,1),IT(IDIREC,2),F(IDIREC,2),  &
                     & IT(IDIREC,3),F(IDIREC,3),IT(IDIREC,4),F(IDIREC,4)
        GOTO 113
     ENDIF
 113 CONTINUE
     IF(IDIREC >= 31)GOTO 40
     GOTO 20
!
```

```
!  END DIRECTION LOOP
!
  40 CONTINUE
     WRITE(13,*)
     IF(NPROB == 4 .OR. NPROB == 10 .OR. NPROB == 16)THEN
        IF(ALLN >= 2)GOTO 50
     ELSE IF(NPROB == 1 .OR. NPROB == 3 .OR. NPROB == 5 .OR. NPROB == 12)THEN
        IF(ALLN >= 3)GOTO 50
     ELSE IF(NPROB == 11 .OR. NPROB == 17)THEN
        IF(ALLN >= 4)GOTO 50
     ELSE IF(NPROB == 7)THEN
        IF(ALLN >= 30)GOTO 50
     ELSE IF(NPROB == 2)THEN
        IF(ALLN >= 6)GOTO 50
     ELSE
        IF(ALLN >= 4000)GOTO 50
     ENDIF
   GOTO 10
!
!  END DIMENSION LOOP
!
  50 CONTINUE
     WRITE(13,203)'====================================================&
                  &====================================='
 101 FORMAT(2X,'SD',9X,I3,2X,F4.2,2X,I5,2X,I7,3X,I5,2X,I7,3X,I5,2X,I7,3X,I5,2X,I7)
 103 FORMAT(2X,'PR',14X,F4.2,2X,I5,2X,I7,3X,I5,2X,I7,3X,I5,2X,I7,3X,I5,2X,I7)
 105 FORMAT(2X,'BFGS',12X,F4.2,2X,I5,2X,I7,3X,I5,2X,I7,3X,I5,2X,I7,3X,I5,2X,I7)
 107 FORMAT(2X,'aPR+bBFGS',7X,F4.2,2X,I5,2X,I7,3X,I5,2X,I7,3X,I5,2X,I7,3X,I5,2X,I7)
 108 FORMAT(2X,'bPR+BFGS',8X,F4.2,2X,I5,2X,I7,3X,I5,2X,I7,3X,I5,2X,I7,3X,I5,2X,I7)
 109 FORMAT(2X,'SD+PR+BFGS',6X,F4.2,2X,I5,2X,I7,3X,I5,2X,I7,3X,I5,2X,I7,3X,I5,2X,I7)
 111 FORMAT(2X,'SD+aPR+bBFGS',4X,F4.2,2X,I5,2X,I7,3X,I5,2X,I7,3X,I5,2X,I7,3X,I5,2X,I7)
 112 FORMAT(18X,F4.2,2X,I5,2X,I7,3X,I5,2X,I7,3X,I5,2X,I7,3X,I5,2X,I7)
 200 FORMAT(27X,A12,5X,A12,8X,A5,11X,A6)
 201 FORMAT(A10,5X,A1,2X,A5)
 202 FORMAT(27X,A2,7X,A2,6X,A2,7X,A2,6X,A2,7X,A2,6X,A2,7X,A2)
 203 FORMAT(A90)
 204 FORMAT(3X,I2,2X,D10.3,1X,I5,1X,100(D15.8),1X,D15.8,1X,D15.8,/)
 205 FORMAT(2X,'PR',10X,F4.2,1X,A5,1X,A7,1X,1X,A5,1X,A7,1X,  &
           & 1X,A5,1X,A7,1X,1X,I5,1X,A7,1X,F5.2)
CLOSE(13)
CLOSE(14)

STOP
END
!
! END HYBRIDDIRECT PROGRAM
!
  FUNCTION HAND_FPE (SIGNUM, EXCNUM)
  !MS$ATTRIBUTES C :: HAND_FPE
  USE MSFLIB
  INTEGER*2  SIGNUM, EXCNUM
  WRITE(*,*) 'IN SIGNUM HANDLER FOR SIG$FPE'
  WRITE(*,*) 'SIGNUM = ', SIGNUM
  WRITE(*,*) 'EXCEPTION = ', EXCNUM
  SELECT CASE(EXCNUM)
     CASE(FPE$INVALID )
          STOP ' FLOATING POINT EXCEPTION: INVALID NUMBER'
     CASE( FPE$DENORMAL )
          STOP ' FLOATING POINT EXCEPTION: DENORMALIZED NUMBER'
     CASE( FPE$ZERODIVIDE )
          STOP ' FLOATING POINT EXCEPTION: ZERO DIVIDE'
     CASE( FPE$OVERFLOW )
          STOP ' FLOATING POINT EXCEPTION: OVERFLOW'
     CASE( FPE$UNDERFLOW )
          STOP ' FLOATING POINT EXCEPTION: UNDERFLOW'
     CASE( FPE$INEXACT )
          STOP ' FLOATING POINT EXCEPTION: INEXACT PRECISION'
     CASE DEFAULT
          STOP ' FLOATING POINT EXCEPTION: NON-IEEE TYPE'
   END SELECT
   HAND_FPE = 1
   RETURN
   END
!
! COMPUTATION ERROR DETECTION
!
   SUBROUTINE MATHERRQQ( NAME, LENGTH, INFO, RETCODE)
```

```
   USE MSFLIB
   INTEGER*2 LENGTH, RETCODE
   CHARACTER(LENGTH) NAME
   RECORD /MTH$E_INFO/ INFO
   RETURN
   WRITE(*,*) "ENTERED MATHERRQQ"
   WRITE(*,*) "FAILING FUNCTION IS: ", NAME
   WRITE(*,*) "ERROR TYPE IS: ", INFO.ERRCODE
   IF((INFO.FTYPE == TY$REAL4 ).OR.(INFO.FTYPE == TY$REAL8)) THEN
     WRITE(*,*) "TYPE: REAL"
     WRITE(*,*) "ENTER THE DESIRED FUNCTION RESULT: "
     READ(*,*) INFO.R8RES
     RETCODE = 1
   ELSEIF ((INFO.FTYPE == TY$CMPLX8 ).OR.(INFO.FTYPE == TY$CMPLX16)) THEN
       WRITE(*,*) "TYPE: COMPLEX"
       WRITE(*,*) "ENTER THE DESIRED FUNCTION RESULT: "
       READ(*,*) INFO.C16RES
       RETCODE = 1
    ENDIF
    END
!
!  START BFGS SUBROUTINE
!
!  THIS SUBROUTINE COMPUTE NEW BFGS DIRECTIONS
!
   SUBROUTINE BFGS(N,ITS,LDH,H,X,XOLD,G,GOLD,DBFGS)
   IMPLICIT INTEGER(I)
   INTEGER N,NMAX,I,J,LDH,ITS
   PARAMETER (NMAX = 200)
   DOUBLE PRECISION TOLX,ZERO,ONE
   DOUBLE PRECISION DDOT,RHO,YHY,SS,YS,YY,DABS
   DOUBLE PRECISION G(N),GOLD(N),HY(NMAX),DBFGS(N)
   DOUBLE PRECISION H(LDH,N),S(NMAX),Y(NMAX),X(N),XOLD(N)
   PARAMETER (TOLX = 1.0D-8)
   EXTERNAL  DDOT
   EXTERNAL  DSYMV
   INTRINSIC DSQRT
   INTRINSIC DMAX1
   INTRINSIC DABS
   DATA ZERO,ONE /0.0D0,1.0D0/
   IF(ITS == 1)THEN
      DO I = 1,N
        DO J = 1,N
           H(I,J) = ZERO
        ENDDO
        H(I,I)= ONE
      ENDDO
    ENDIF
!  COMPUTE THE DIFFERENCE OF NEW AND CURRENT ITERATES
    DO I = 1,N
       S(I) = X(I) - XOLD(I)
    ENDDO
!  COMPUTE THE DIFFERENCE OF NEW AND CURRENT GRADIENTS
    DO I = 1,N
       Y(I) = G(I) - GOLD(I)
    ENDDO
    CALL DSYMV('UPPER TRIANGULAR H',N,ONE,H,LDH,Y,1,ZERO,HY,1)
!  CALCULATE DOT PRODUCTS FOR THE DENOMINATORS.
    YS = DDOT(N,Y,1,S,1)
    YY = DDOT(N,Y,1,Y,1)
    SS = DDOT(N,S,1,S,1)
    YHY= DDOT(N,Y,1,HY,1)
!  SKIP UPDATE IF YS IS NOT SUFFICIENTLY POSITIVE.
    IF(YS > DSQRT(TOLX*YY*SS))THEN
       RHO = ONE/YS
!  THE BFGS UPDATING FORMULA:
       DO I = 1,N
         DO J = I,N
            H(I,J)= H(I,J)-RHO*(S(I)*HY(J)+HY(I)*S(J))   &
                    &+(YHY*RHO**2+RHO)*S(I)*S(J)
         ENDDO
       ENDDO
     ELSE
       DO I = 1,N
         DO J = 1,N
            H(I,J) = ZERO
         ENDDO
```

```
        H(I,I) = ONE
      ENDDO
    ENDIF
!  NOW CALCULATE THE NEXT DIRECTION TO GO, AND GO BACK FOR ANOTHER ITERATION.
    CALL DSYMV('UPPER TRIANGULAR H', N,-ONE,H,LDH,G,1,ZERO,DBFGS,1)

  RETURN
  END
!
! END BFGS SUBROUTINE
!
! START CG SUBROUTINE
!
! NONLINEAR CONJUGATE GRADIENT DIRECTION(POLAK-RIBIERE)
!
 SUBROUTINE NCG(N,G,GOLD,DCG)
 IMPLICIT INTEGER(I)
 INTEGER N,NMAX,I
 PARAMETER (NMAX = 200)
 DOUBLE PRECISION BETA,SCALE,PONE,RST
 DOUBLE PRECISION DDOT,GNEWG,GGOLD,NRMGOLD,NRMGNEW
 DOUBLE PRECISION G(N),GOLD(N),GLC(NMAX),D(NMAX),DCG(N)
 EXTERNAL DDOT
 EXTERNAL DAXPY
 INTRINSIC DABS
 DATA PONE,SCALE  /1.0D-1,-1.0D0/
 DO I = 1,N
    D(I) = DCG(I)
    GLC(I) = G(I)
    DCG(I)=-G(I)
 ENDDO
 NRMGOLD = DDOT(N,GOLD,1,GOLD,1)
 NRMGNEW = DDOT(N,G,1,G,1)
 GGOLD = DDOT(N,G,1,GOLD,1)
! RESTARTED WHEN RST >= 0.1
 RST = DABS(GGOLD)/NRMGNEW
 IF(RST >= PONE)RETURN
 CALL DAXPY (N,SCALE,GOLD,1,GLC,1)
 GNEWG = DDOT(N,G,1,GLC,1)
 BETA = GNEWG/NRMGOLD
! COMPUTE NEW DIRECTION NEXT TO GO,
 CALL DAXPY (N,BETA,D,1,DCG,1)

 RETURN
 END
!
! END CG SUBROUTINE
!
!
! BACKTRACKING LINE SEARCH SUBROUTINE
!
 SUBROUTINE LNSRCH(N,X,FX,DL,G,SLOPE,FE,GE,NPROB,FAIL)
 IMPLICIT NONE
 INTEGER I,N,NMAX,K,FE,GE,NPROB,FAIL
 PARAMETER (NMAX = 200)
 DOUBLE PRECISION C1,ZERO,PONE,HALF,ONE,TWO,THREE
 DOUBLE PRECISION LAMBDA,LAMBLO,LAMBDA0,LAMBDA02
 DOUBLE PRECISION A,B,DISC,RHS1,RHS2,SLOPE,FX,FX0,FX1
 DOUBLE PRECISION G(N),DL(NMAX),X(N),XLC(NMAX)
 EXTERNAL OBJFCN
 EXTERNAL GRDFCN
 INTRINSIC DSQRT
 INTRINSIC DMAX1
 DATA ZERO,PONE,HALF,ONE,TWO,THREE,C1 /0.0D0,1.0D-1,0.5D0,1.0D0,2.0D0,3.0D0,1.0D-4/
 FX0 = FX
 DO I = 1,N
    XLC(I)= X(I)
 ENDDO
 LAMBLO = PONE
 LAMBDA0 = ONE
 FAIL = 0
 K = 0
1 CONTINUE
  K = K + 1
  IF(K >= 50)THEN
     LAMBDA0 = LAMBLO
     FAIL = 1
```

```
  ENDIF
  DO I = 1,N
     X(I) = XLC(I) + LAMBDA0*DL(I)
  ENDDO
  CALL OBJFCN(N,X,FX,NPROB)
  FE = FE + 1
  IF(FAIL == 1)THEN
     CALL GRDFCN(N,X,G,NPROB)
     GE = GE + N
  RETURN
  ENDIF
    IF(FX <= FX0 + C1*LAMBDA0*SLOPE)THEN
       CALL GRDFCN(N,X,G,NPROB)
       GE = GE + N
       RETURN
    ELSE
       IF(LAMBDA0 == ONE)THEN
          LAMBDA = (-SLOPE)/(TWO*(FX -FX0 -SLOPE))
       ELSE
          RHS1 = FX -FX0 -LAMBDA0*SLOPE
          RHS2 = FX1 -FX0 -LAMBDA02*SLOPE
          A = (RHS1/LAMBDA0**2-RHS2/LAMBDA02**2)/(LAMBDA0-LAMBDA02)
          B = (-LAMBDA02*RHS1/LAMBDA0**2+LAMBDA0*RHS2/LAMBDA02**2)/ &
              &(LAMBDA0-LAMBDA02)
          IF(A == ZERO)THEN
             LAMBDA = (-SLOPE)/(TWO*B)
          ELSE
             DISC = B*B -THREE*A*SLOPE
             IF(DISC < ZERO)THEN
                LAMBDA = HALF*LAMBDA0
             ELSE IF(B <= ZERO)THEN
                LAMBDA = (-B+DSQRT(DISC))/(THREE*A)
             ELSE
                LAMBDA = (-SLOPE)/(B+DSQRT(DISC))
             ENDIF
            IF(LAMBDA > HALF*LAMBDA0) LAMBDA=HALF*LAMBDA0
          ENDIF
       ENDIF
    ENDIF
    LAMBDA02 = LAMBDA0
    FX1 = FX
    LAMBDA0 = DMAX1(LAMBDA,PONE*LAMBDA0)
 GOTO 1
 END
!
! END BACKTRACKING LINE SEARCH SUBROUTINE
!
!
! START LINESRCH SUBROUTINE SATISFIES STRONG WOLFE CONDITIONS WITH BISECTION
! INTERPOLATION
!
 SUBROUTINE LINESRCH(N,X,FX,D,G,SLOPE,FE,GE,NPROB,FAIL)
 IMPLICIT NONE
 INTEGER I,J,FAIL,N,NMAX,FE,GE,NPROB
 PARAMETER (NMAX = 200)
 DOUBLE PRECISION DDOT,C1,C2,ZERO,ONE,TWO,LAMBLO
 DOUBLE PRECISION LO,HI,FX,FX0,FLO,FHI,SLOPE,GHI,PONE
 DOUBLE PRECISION X(N),XLC(NMAX),D(N),G(N),GLC(NMAX)
 EXTERNAL DDOT
 EXTERNAL OBJFCN
 EXTERNAL GRDFCN
 EXTERNAL ZOOM
 INTRINSIC DSQRT
 DATA C1,C2,ZERO,PONE,ONE,TWO  /1.0D-04,1.0D-01,0.0D0,1.0D-01,1.0D0,2.0D0/
 DO I = 1,N
    XLC(I) = X(I)
    GLC(I) = G(I)
 ENDDO
 FAIL = 0
 LO = ZERO
 LAMBLO = PONE
 FX0 = FX
 FLO = FX0
 HI = ONE
 J = 0
2 CONTINUE
    J = J + 1
```

```
    DO I = 1,N
       X(I) = XLC(I) + HI*D(I)
    ENDDO
    CALL OBJFCN(N,X,FX,NPROB)
    FHI= FX
    FE = FE + 1
    IF(FAIL == 1)THEN
       CALL GRDFCN(N,X,G,NPROB)
       GE = GE + N
       RETURN
    ENDIF
    IF((FHI > FX0 + C1*HI*SLOPE) .OR. (FHI >= FLO .AND. J > 1) )THEN
        FX = FX0
        DO I = 1,N
           X(I) = XLC(I)
           G(I) = GLC(I)
        ENDDO
        CALL ZOOM(N,LAMBLO,LO,HI,X,FX,D,G,SLOPE,FE,GE,NPROB,FAIL)
        RETURN
     ENDIF
     CALL GRDFCN(N,X,G,NPROB)
     GHI = DDOT(N,G,1,D,1)
     GE = GE + N
     IF(DABS(GHI) <= -C2*SLOPE)THEN
        RETURN
     ENDIF
     IF(GHI >= ZERO)THEN
        FX = FX0
        DO I = 1,N
           X(I) = XLC(I)
           G(I) = GLC(I)
        ENDDO
        CALL ZOOM(N,LAMBLO,LO,HI,X,FX,D,G,SLOPE,FE,GE,NPROB,FAIL)
        RETURN
     ENDIF
     LO  = HI
     FLO = FHI
     HI = TWO*HI
     IF(J >= 50)THEN
        HI = LAMBLO
        FAIL = 1
     ENDIF
 GOTO 2
 END
!
! END LINESRCH SUBROUTINE
!
!
! SUBROTUINE ZOOM COMPUTE STEP LENGTH WITH BISECTION INTERPOLATION
!
 SUBROUTINE ZOOM(N,LAMBLO,LO,HI,X,FX,D,G,SLOPE,FE,GE,NPROB,FAIL)
 IMPLICIT NONE
 INTEGER I,K,N,NMAX,FAIL,FE,GE,NPROB
 PARAMETER (NMAX = 200)
 DOUBLE PRECISION  C1,C2,ZERO,TWO,LO,HI,ATRY,BISECT,LAMBLO
 DOUBLE PRECISION  FX,FX0,FTRY,FLO,DABS,DDOT,SLOPE,GTRY
 DOUBLE PRECISION  X(N),XLC(NMAX),D(N),G(N)
 EXTERNAL  DDOT
 EXTERNAL  OBJFCN
 EXTERNAL  GRDFCN
 INTRINSIC DABS
 INTRINSIC DSQRT
 DATA C1,C2,ZERO,TWO /1.0D-04,1.0D-01,0.0D0,2.0D0/
 FAIL = 0
 FX0 = FX
 FLO = FX0
 DO I = 1,N
    XLC(I) = X(I)
 ENDDO
 K = 0
3 CONTINUE
    K = K + 1
    IF(LO .NE. HI) THEN
       BISECT = (LO + HI)/TWO
       ATRY = BISECT
       IF(K >= 50)THEN
          ATRY = LAMBLO
```

```
            FAIL = 1
         ENDIF
         DO I = 1,N
            X(I) = XLC(I) + ATRY*D(I)
         ENDDO
         CALL OBJFCN(N,X,FX,NPROB)
         FTRY = FX
         FE = FE + 1
         IF(FAIL == 1)THEN
            CALL GRDFCN(N,X,G,NPROB)
            GE = GE + N
            RETURN
         ENDIF
         IF((FTRY > FX0 + C1*ATRY*SLOPE) .OR. (FTRY >= FLO) )THEN
            HI = ATRY
         ELSE
            CALL GRDFCN(N,X,G,NPROB)
            GTRY = DDOT(N,G,1,D,1)
            GE = GE + N
            IF(DABS(GTRY) <= -C2*SLOPE)THEN
               RETURN
            ENDIF
! CHECK WHICH WAY TO LOOK NEXT
            IF(GTRY*(HI - LO) >= ZERO)THEN
               HI = LO
            ENDIF
            LO  = ATRY
            FLO = FTRY
         ENDIF
      ENDIF
   GOTO 3
   END
!
! END ZOOM SEARCH SUBROUTINE
!
! START LINESRCH1 SUBROUTINE SATISFIES STRONG WOLFE CONDITIONS WITH BISECTION
! INTERPOLATION
!
   SUBROUTINE LINESRCH1(N,X,FX,D,G,SLOPE,FE,GE,NPROB,FAIL)
   IMPLICIT NONE
   INTEGER I,J,FAIL,N,NMAX,FE,GE,NPROB
   PARAMETER (NMAX = 200)
   DOUBLE PRECISION DDOT,C1,C2,ZERO,ONE,TWO,LAMBLO
   DOUBLE PRECISION LO,HI,FX,FX0,FLO,FHI,SLOPE,GHI,PONE
   DOUBLE PRECISION X(N),XLC(NMAX),D(N),G(N),GLC(NMAX)
   EXTERNAL DDOT
   EXTERNAL OBJFCN
   EXTERNAL GRDFCN
   EXTERNAL ZOOM
   INTRINSIC DSQRT
   DATA C1,C2,ZERO,PONE,ONE,TWO  /1.0D-04,1.0D-01,0.0D0,1.0D-01,1.0D0,2.0D0/
   DO I = 1,N
      XLC(I) = X(I)
      GLC(I) = G(I)
   ENDDO
   FAIL = 0
   LO = ZERO
   LAMBLO = PONE
   FX0 = FX
   FLO = FX0
   HI = ONE
   J = 0
 4 CONTINUE
      J = J + 1
      DO I = 1,N
         X(I) = XLC(I) + HI*D(I)
      ENDDO
      CALL OBJFCN(N,X,FX,NPROB)
      FHI= FX
      FE = FE + 1
      IF(FAIL == 1)THEN
         CALL GRDFCN(N,X,G,NPROB)
         GE = GE + N
         RETURN
      ENDIF
      IF((FHI > FX0 + C1*HI*SLOPE) .OR. (FHI >= FLO .AND. J > 1) )THEN
          FX = FX0
```

```
            DO I = 1,N
               X(I) = XLC(I)
               G(I) = GLC(I)
            ENDDO
            CALL ZOOM(N,LAMBLO,LO,HI,X,FX,D,G,SLOPE,FE,GE,NPROB,FAIL)
            RETURN
         ENDIF
         CALL GRDFCN(N,X,G,NPROB)
         GHI = DDOT(N,G,1,D,1)
         GE = GE + N
         IF(DABS(GHI) <= -C2*SLOPE)THEN
            RETURN
         ENDIF
         IF(GHI >= ZERO)THEN
            FX = FX0
            DO I = 1,N
               X(I) = XLC(I)
               G(I) = GLC(I)
            ENDDO
            CALL ZOOM(N,LAMBLO,LO,HI,X,FX,D,G,SLOPE,FE,GE,NPROB,FAIL)
            RETURN
         ENDIF
         LO  = HI
         FLO = FHI
         HI = TWO*HI
         IF(J >= 50)THEN
            HI = LAMBLO
            FAIL = 1
         ENDIF
     GOTO 4
     END
!
! END LINESRCH1 SUBROUTINE
!
!
! SUBROTUINE ZOOM1 COMPUTE STEP LENGTH WITH BISECTION INTERPOLATION
!
     SUBROUTINE ZOOM1(N,LAMBLO,LO,HI,X,FX,D,G,SLOPE,FE,GE,NPROB,FAIL)
     IMPLICIT NONE
     INTEGER I,K,N,NMAX,FAIL,FE,GE,NPROB
     PARAMETER (NMAX = 200)
     DOUBLE PRECISION  C1,C2,ZERO,TWO,LO,HI,ATRY,BISECT,LAMBLO
     DOUBLE PRECISION  FX,FX0,FTRY,FLO,DABS,DDOT,SLOPE,GTRY
     DOUBLE PRECISION  X(N),XLC(NMAX),D(N),G(N)
     EXTERNAL  DDOT
     EXTERNAL  OBJFCN
     EXTERNAL  GRDFCN
     INTRINSIC DABS
     INTRINSIC DSQRT
     DATA C1,C2,ZERO,TWO /1.0D-04,1.0D-01,0.0D0,2.0D0/
     FAIL = 0
     FX0 = FX
     FLO = FX0
     DO I = 1,N
        XLC(I) = X(I)
     ENDDO
     K = 0
    5 CONTINUE
        K = K + 1
        IF(LO .NE. HI) THEN
           BISECT = (LO + HI)/TWO
           ATRY = BISECT
           IF(K >= 50)THEN
              ATRY = LAMBLO
              FAIL = 1
           ENDIF
           DO I = 1,N
              X(I) = XLC(I) + ATRY*D(I)
           ENDDO
           CALL OBJFCN(N,X,FX,NPROB)
           FTRY = FX
           FE = FE + 1
           IF(FAIL == 1)THEN
              CALL GRDFCN(N,X,G,NPROB)
              GE = GE + N
              RETURN
           ENDIF
```

```
        IF((FTRY > FX0 + C1*ATRY*SLOPE) .OR. (FTRY >= FLO) )THEN
           HI = ATRY
        ELSE
           CALL GRDFCN(N,X,G,NPROB)
           GTRY = DDOT(N,G,1,D,1)
           GE = GE + N
           IF(DABS(GTRY) >= C2*SLOPE)THEN
              RETURN
           ENDIF
! CHECK WHICH WAY TO LOOK NEXT
           IF(GTRY*(HI - LO) >= ZERO)THEN
              HI = LO
           ENDIF
           LO  = ATRY
           FLO = FTRY
        ENDIF
     ENDIF
  GOTO 5
  END
!
! END ZOOM1 SEARCH SUBROUTINE
!
! START EXPONENTIAL SCHEDULE LINE SEARCH SUBROUTINE
!
  SUBROUTINE EXPSLNS(N,X,FX,DL,G,SLOPE,NPROB,FE,GE,FAIL)
  IMPLICIT NONE
  INTEGER I,M,N,NMAX,K,FE,GE,NPROB,FAIL
  PARAMETER (NMAX = 200)
  DOUBLE PRECISION PONE,ONE,TWO,LAMBDA,LAMBDA0
  DOUBLE PRECISION FX,FX0,SLOPE,LAMBLO
  DOUBLE PRECISION G(N),DL(N),X(N),XLC(NMAX)
  EXTERNAL OBJFCN
  EXTERNAL GRDFCN
  DATA PONE,ONE,TWO  /1.0D-01,1.0D0,2.0D0/
  FX0 = FX
  DO I = 1,N
     XLC(I)= X(I)
  ENDDO
  LAMBLO = PONE
  LAMBDA0= ONE
  LAMBDA = LAMBDA0
  FAIL = 0
  K = 0
  M = 1
 6 CONTINUE
     K = K + 1
     DO I = 1,N
        X(I) = XLC(I) + LAMBDA*DL(I)
     ENDDO
     CALL OBJFCN(N,X,FX,NPROB)
     FE = FE + 1
     IF(FAIL == 1)THEN
        CALL GRDFCN(N,X,G,NPROB)
        GE = GE + N
        RETURN
     ENDIF
     IF(FX  <= FX0 + (ONE/TWO)*LAMBDA*SLOPE)THEN
        CALL GRDFCN(N,X,G,NPROB)
        GE = GE + N
        RETURN
     ELSE
        M = M + 1
        LAMBDA = LAMBDA0/(TWO**(M-1))
        IF(K >= 50)THEN
           LAMBDA = LAMBLO
           FAIL = 1
        ENDIF
     ENDIF
  GOTO 6
  END
```

Curriculum Vitae

FIRST NAME: Phaichayon

LAST NAME: Sirisathienwatthana

SEX: Male

NATIONALITY: Thai

DATE OF BIRTH: January 01, 1977

MARITAL STATUS: Single

EDUCATION BACKGROUND:

Bachelor of Education (Mathematics). March 2000, Rajabhat Kumphaengphet Institute, Kumphaengphet, Thailand.

WORK EXPERIENCE:

Teaching Assistant, Trimester 1 and 2/Academic Year 2002, School of Mathematics, Suranaree University of Technology, Nakhon Ratchasima, Thailand.