Notes on Optimization

These notes were prepared for an Optimization course that I taught in IIT Guwahati in 2006. They may be useful to some students, as they provide a concise summary of many of the basic concepts. Simple exercises mentioned here may be useful for students to test their understanding.

Some texts are suggested in these notes, but the material is standard and available in many books.

Narayan Rangaraj
narayan.rangaraj@iitb.ac.in

Text – Optimization Concepts and Applications in Engineering, Ashok D. Belegundu and Tirupathi R. Chandrupatla, Pearson Education, Delhi, 1999 – **B and C**

Supplementary texts –
For LP - *Linear Programming and Network Flows, [2nd edition], M.S.Bazaraa, J.J.Jarvis and H.D.Sherali, Wiley, 1990 or Operations Research, H. Taha*
For non-traditional optimization - *Modern Heuristics for Combinatorial Optimization, C. Reeves, Orient Longman, 1993*
For Engineering Applications – *Optimization for Engineering Design : Algorithms and Examples, K. Deb, Prentice Hall India, 1995* – **Deb** or **K.Deb**
Mathematical Background – *Introduction to Optimization, Edwin K.P.Chong and S.H.Zak* - **C and Z**
General Background, Geometric Programming, etc. – *Optimization: Theory and Applications, S.S.Rao, Wiley Eastern, 1984*
Numerical methods – *Numerical Optimization, Jorge Nocedal and Stephen J. Wright, Springer, 1999*

Introduction to optimization

Optimization in this course will consist of minimization or maximization (it doesn't matter) of a "well defined" function of several (but finite number of) variables, perhaps subject to some constraints. Continuous optimization relies on the use of derivatives (slopes or gradients) and higher order quantities like second derivatives (curvature and its generalizations), to model a function's behavior as the parameters change.

Most of continuous optimization will rely on either a functional form of the objective that has to be optimized, or at least that there are smooth properties of a function that can be evaluated efficiently at various points. Note that derivatives of various orders can be approximated by finite differences when required.

We will also consider discrete optimization, which deals with a (usually finite) discrete set of feasible points. Here, notions of derivatives and model functions are not directly

useful, but notions of descent, improvement and local/global optimality are useful here, too.

What this course will NOT consider: Multi-objective optimization, fuzzy function theory, probabilistic concerns both of objective and constraints and equilibrium or game theoretic models. Some of these can be treated as fairly natural, but non-trivial extensions of the theory that we will see here.

Some nice things that you may learn from this course

- Linear programming as a link technique between continuous and discrete optimization
- Linear programming duality and sensitivity, its extensions to Nonlinear problems and the interpretations there
- (Convex) Quadratic programming (QP) is as easy as linear programming
- Quasi Newton methods based on approximations of second order information would be among the most efficient general purpose nonlinear programming techniques
- Sequential QP would form among the most robust general purpose tools for constrained optimization
- Integer programming problems are sometimes solvable using linear programming subproblems

Self study

Optimization of a function of one variable

This consists of derivative and nonderivative methods that are quite nice and intuitive and useful. Please read them. As subroutines for larger (dimensional) problems, it may be enough to find the first (local) optimum point in a given direction and that too, approximately.

See Chap 2 in Belegundu and Chandrupatla, and also in K.Deb)

Basic linear programming

Graphical method for 2 variable problems, extreme point properties of a solution (notion of a basic feasible solution) and the basic simplex method will be taken as the starting point.

See Chap 4.1-4.4 of B and C and Appendix in Deb

Some exercises

- Formulate a procedure for minimizing a function f of n variables. The only information available is to be able to evaluate f(x) given x. It is believed that the function is a smooth one (differentiable, etc.), but no formula is available. Assume that some "reasonable" starting solution is available.

- On a directed network of nodes (set N) and directed arcs (set A) with each arc going from some node to another, and a distance function defined on each arc, describe using proper notation, the problem of finding the shortest path between two designated nodes s and t. Try to also formulate this problem as a linear programming problem. Note that LP is a continuous problem.

- Show rigorously how maximization and minimization problems are equivalent. You have to first formulate a rigorous statement about this! The crux is that if you can do one, you can do the other.

Some preliminaries on norms, derivatives and later on, linear algebra, especially of positive definite matrices, are available as intro chapters or detailed appendices in any book on optimization or even O.R. and on books on (real) analysis and linear algebra. A basic understanding of this, along with appropriate notation for vectors and matrices is essential for this course.

Norms on vectors

For defining derivatives for functions of several variables, for describing algorithms and their convergence to optimal points and even to characterize optimal points in the first place, we need notions of norms on vectors (and later, on matrices). A norm is a function $|| x ||$ defined on n-dimensional vectors x, that measures the magnitude of vector x, that satisfy the following conditions:

1. $|| x || >= 0$ for all vectors x and $|| x ||= 0$ only for the zero vector x
2. $|| ax || = |a| || x ||$ for all vectors x and scalars a and
3. $|| x || + || y || >= || x+y ||$ for any two vectors x and y.

A set of functions satisfying these conditions are the p-norms on vectors, defined as -

$|| x ||_p$ = p-th root of $\Sigma_i | x_i |^p$

The two norm (p=2) is the usual Euclidean norm and the one norm is the rectilinear or Manhattan norm and p = infinity corresponds to the max norm, i.e. $|| x || = \max_i | x_i |$.

For most of our purposes, these norms (and some others) are equivalent in the sense that a sequence of vectors that converges to zero under one norm does so under another.

Derivatives

A derivative is a linear operator that captures the local behaviour of a function. When evaluated at a point x, it provides an estimate of the function value at points near x, in a quantifiably approximate way.

For a function of one variable, the notion of a derivative is a constant evaluated at each point. This notion extended to a function f of n variables $f(x_1, \ldots, x_n)$ results in the definition of a gradient vector $\nabla f(x)$, the vector of partial derivatives with respect to the

co-ordinate variables. The i-th component of $\nabla f(x)$ is the partial derivative of f(x) w.r.t. x_i.

The second derivative generalization is that of the Hessian matrix, defined as the (symmetric) matrix of cross derivatives, evaluated at some point.

Characterization of optima and introduction to algorithms

Characterization of optima

It is not possible to design efficient algorithms for optimization unless the characterization of optima is exact and easily computable or verifiable. For unconstrained minima/maxima of differentiable functions, the optimum is characterized by $f'(x) = 0$ for a function of one variable, and $\nabla f(x) = 0$ (the zero vector) for a function f defined on n-variables. [Verify this, using the Taylor series expansion for a function of n-variables.]

Existence: For optimization over R^n, the Weierstrass theorem ensures that a continuous function over a closed, bounded set will have an optimum point in the set, i.e. it makes sense to find an optimal solution. In some applications, unbounded solutions are also of interest, and in Linear Programming, unbounded solutions can also be characterized fully, using extreme directions, along which the objective function increases/decreases indefinitely. But for most practical problems, an unbounded solution usually implies that some constraints are missing. If nothing, physical conditions on the design parameters (variables) would result in box constraints to make the feasible set bounded, usually.

Examples: Please read K. Deb's text or other texts for some examples of linear and non-linear and discrete optimization problems drawn from various fields of engineering.

Local and global optima: Analytical methods for finding optimal solutions usually find only a local optimum. The reason is that they are based on models of functions, derivatives and extrapolations (expansions) based on those, all of which are valid only locally. It is necessary to understand how to find a good, local solution first for the reasons that a global minimum is also a local minimum and any algorithm for a global minimum will benefit from a fast procedure to find a local minimum. Practically speaking, some problems deal with minimization of convex functions, where the local min is also a global min or a good starting point is known, from prior experience or heuristically, which will ensure convergence to a globally optimal solution.

However, over time, people are formulating challenging and new optimization problems, where prior information is not available and where there is a genuine difficulty in obtaining good, global solutions. Many non-traditional methods aim to find such solutions. The performance of these methods is generally valid only statistically speaking and the efficacy is determined by extensive numerical experimentation. But they form an increasingly powerful and acceptable class of methods for optimization.

Most of the notation henceforth refers to optimization of a function f defined on a set K, which is a subset of R^n. Most of the illustrations are for minimization.

Global optimum: The vector x* is a global minimum of f over K if f(x*) <= f(x) for all x in K. This is difficult to verify/certify, in general, other than by exhaustive enumeration/search.

Local optimum: The vector x* is a local minimum of f if there is a neighbourhood N of x* such that f(x*) <= f(x) for x in N. The neighbourhood N is defined using some parameter ε, as $\{x : \|x-x^*\| <= \varepsilon \}$. Note that a norm is involved in this definition and hence also in the definition of a local min.

The necessary condition $\nabla f(x^*) = 0$ is required for x* to be a local minimum. Note also that we are implicitly assuming that all points in the neighbourhood N are feasible, i.e. belonging to set K. If this is not so, the condition for a local minimum needs to be modified. So this is usually stated as the condition for an unconstrained minimum of f. It also holds when x* belongs to the interior of set K (i.e. when we can define a neighbourhood N of x* with all elements in N being in set K).

Question: Is a solution obtained through the Simplex method of LP a local or a global minimum?

For discrete optimization problems, the notion of a global minimum is the same (i.e. a solution that is at least as good as all other vectors), but the notion of a local minimum is less clear and is somewhat subjective.

Exercise: For the traveling salesman problem (TSP) on n nodes – you can consider the directed version, which will even allow the i-j distance to be different from the j-i distance, define a (feasible) solution in the following two ways and characterize optimality in each case. Please make sure you define the input data/parameters for the TSP, for this will help you in defining the objective function explicitly in terms of the representation that you select.

Method 1: $a = [a_1, \ldots, a_n]$ is a permutation of the integers 1, …, n. a_i represents the i-th city in the tour. How many solutions are there in this representation? Are all solutions captured by this? Many tours are equivalent in this definition. How many distinct tours are there?

Method 2: X = n x n matrix of 0's and 1's. $x_{ij} = 1$ if city j appears in the i-th position in a tour. Constraints are required on the n x n x_{ij} variables to define feasible tours. Define these. Map a solution vector in Method 1 to one in Method 2 and vice versa.

Try to define locally optimal solutions for each representation, by defining an appropriate neighbourhood. Note that you would have to define a norm on the solution space to capture the notion of "nearby" solutions.

Descent algorithms:

One large class of iterative methods for nonlinear optimization belong to the following scheme;

General descent algorithm:
Start with some vector x^0

At the k-th step, given x^k, if $\nabla f(x^k)$ is not equal to the zero vector)

1. Find a descent direction d^k
2. Find a step size α_k so that $x_{k+1} = x_k + \alpha_k d^k$ is a "better" point

Various methods can be defined depending on how d^k and α_k are determined.

Motivation for descent based methods:

From the expansion $f(x+h) = f(x) + \nabla f(x)^T h$ + a small error term (for h with "small" norm, the error term goes to zero faster than $||h||$), if we use this at the point $x = x^k$, we see that choosing h to be = -- $\alpha_k \nabla f(x^k)$ would give $x^k + h$ with smaller function value than x^k. The parameter α_k has to be small enough to make the approximation a valid one.

In fact, any h which gives $\nabla f(x^k)^T h < 0$ is valid and this gives rise to a set of methods. A general way of writing a descent direction in this framework is d_k = -- $B_k \nabla f(x^k)$. This gives;

1. The steepest descent method for the choice $B_k = I$
2. The Newton method for the choice $B_k = \nabla^2 f(x^k)$
3. The Quasi-Newton method for the choice B_k a suitable (positive definite) approximation to $\nabla^2 f(x^k)$

Note that 1 and 3 always give a descent direction (for $\nabla f(x^k)$ not equal to zero), whereas 2 gives a descent direction for $\nabla^2 f(x^k)$ positive definite.

Note: You would need to review the definition of a positive definite matrix. A symmetric matrix B is positive definite if $h^T B h > 0$ for all non-zero h. For real symmetric matrices, this is equivalent to all the eigenvalues of B being positive (and several other equivalent conditions). The positive definite matrices that we are referring to are the second derivative (Hessian) matrices or their approximations.

Step size determination:

In exact line searches (i.e. where α_k is chosen so as to minimize $f(x^k + \alpha_k d^k)$, the parameter can be determined by any one of the techniques for one dimensional minimization. Since this is just an iterative step in a larger algorithm, inexact line searches, where α_k is chosen based on various criteria (basically, sufficient decrease of f

and sufficiently large values of α_k), work quite well. There are a number of inexact line search rules.

First order necessary condition for a (local) minimum

We re-state a condition for a local min of a real-valued (continuously differentiable) function f defined over a set K in R^n, as follows:

If the vector x* is a local minimum of f, then the condition $\nabla f(x^*)^T d >= 0$ must hold for all feasible directions d. A feasible direction d at $x^* \varepsilon K$ is a vector d such that $(x^* + \alpha d) \varepsilon K$ for all α in some interval $[0, \alpha']$. If the set K is all of R^n, or if the point x* belongs to the interior of set K (which notion you can make precise), then since all directions d are feasible, the only way the condition can be satisfied is for $\nabla f(x^*) = 0$ **[verify this]**.

This is only a necessary condition (the same condition holds for a local maximum also, as well as for points of inflexion or saddle points). A point where $\nabla f(x^*) = 0$ is called a stationary point. Most analytical techniques would attempt to find a stationary point (actually, will be able to progress only if not at a stationary point).

This condition can be verified by using the first-order approximation for a function f around the point x. If $\nabla f(x)^T d < 0$, we would have $f(x + ah) < f(x)$ for some sufficiently small a. This would say that x is not a local min. In the unconstrained case, if $\nabla f(x)$ is nor equal to zero, a choice of h that will give descent is the vector $-\nabla f(x)$. This forms the basis for the steepest descent algorithm, where this direction is combined with a line search step (usually stated as an exact line search).

Steepest descent method (also called Cauchy method)

The steepest descent method is easy to implement and is very robust. It is globally convergent under very general assumptions (i.e. it converges – to a stationary point, which is usually a local minimum) starting from anywhere. For a quadratic function with circular contour lines, it finds the minimum in one step (but with a line search), since the steepest descent direction points to the minimum at every point.

The method performs poorly if the contours of the function are ellipsoids with skewed axes, where the method zigzags, resulting in unacceptably poor performance. This notion is made precise (for quadratic functions of the form $c^T x + \frac{1}{2} x^T Q x$) by the concept of a condition number of the second derivative matrix (the constant matrix Q). For positive definite matrices Q, the condition number is the ratio of the largest eigenvalue to the smallest eigenvalue. The bigger this number, the slower the rate of convergence (see Chong and Zak for nice proofs and insights on the behaviour of this class of methods).

Improved descent methods

There are a number of improvements possible over the basic steepest descent method. One is to do an inexact line search, which may reduce the overall computational effort

(note that the normal convergence results for the steepest descent method are in terms of the number of iterations involving computation of the gradient, and an exact line search is assumed, which may actually consume quite a bit of the computational effort in practice).

But the main source of improvement is in terms of better directions for search. We note that if the gradient vector $\nabla f(x)$ is not equal to zero, there are actually a number of descent directions one can use (other than the negative gradient vector, i.e. $-\nabla f(x)$). In particular, a direction $-B\nabla f(x)$, is guaranteed to give a descent direction at x for a positive definite matrix B. It is convenient to take B as an approximation that will capture some second order information of the function at that point. We will return to this idea after discussing the Newton method.

Two methods that use better directions than the steepest descent direction, are conjugate direction methods and conjugate gradient methods. Both of these are best understood for quadratic functions and suitably generalized to general non-linear functions.

Conjugate direction and conjugate gradient methods

For a quadratic function $c^Tx + \frac{1}{2} x^TQx$, the directions $d_1, \ldots, d_n$ are said to be conjugate with respect to (the symmetric matrix) Q if $d_i^TQd_j = 0$ for i different from j. A set of conjugate directions forms a better set to search over and it can be shown that a line search done sequentially in a set of n conjugate directions will minimize a quadratic function. Note that there are a number of sets of conjugate directions for a given Q.

Rather than defining the set of conjugate directions right in the beginning, an iterative scheme to generate them using gradient information can be proposed, which is the conjugate gradient method. This forms the basis for quite a powerful, general-purpose technique for general non-linear functions (with a suitably generalized interpretation of conjugate directions) and you can refer to B and C, Deb, C and Z and other books for more details.

Second order methods

The most powerful techniques for finite dimensional optimization on R^n, are those based on Newton's method. When these work, there are very few competing methods. They are based on second derivatives (curvature) and quadratic model functions to iteratively generate search directions. The method provides a balance between computational work done and speed of the algorithm (time to find a "good" solution). Even when they do not work (for reasons we will explain), the motivating ideas help us define good approximations that work well.

Before studying the method, we need to look at second order necessary and sufficient conditions for optimality. Define $\nabla^2 f(x^*)$ as the n x n matrix of partial derivatives of f (i.e. $\nabla^2 f(x^*)\,|_{(i,j)}$ is the second partial derivative $\partial^2 f/\partial x_i\,\partial x_j$ evaluated at x^*.) This is a real-valued symmetric matrix for functions of our interest.

Second order necessary and sufficient conditions

The vector x* is a minimum of twice continuously differentiable function f only if $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive semidefinite.

This again is easy to see from the two term expansion of the Taylor series of f around x*, viz. $f(x^* + h) = f(x^*) + \nabla f(x^*)^T h + \frac{1}{2} h^T \nabla^2 f(x^*) h$ + small term that goes to zero faster than $\| h \|^2$. Since $\nabla f(x^*) = 0$, for x* to be a local min (i.e. for small h), the third term on the RHS must be >=0, which is what positive semidefiniteness means.

Note that $\nabla^2 f(x^*)$ is a real, symmetric matrix. Such a matrix is positive semi-definite (positive definite) if and only if all its eigenvalues - which are guaranteed to be real - are non-negative (positive). There are other checks for positive definiteness, involving the various determinants of the principal submatrices, but the eigenvalue test is the most convenient.

It is known **[pl revise this from your linear algebra course]** that a real symmetric matrix is diagonalizable through a factorization as follows $\nabla^2 f(x^*) = U^T \Sigma U$, where is a diagonal matrix containing the eigenvalues of the LHS matrix and U is an orthogonal matrix of unit dimension eigenvectors corresponding to the n-eigenvalues of $\nabla^2 f(x^*)$. You can use this factorization to verify the correspondence between eigenvalues and positive semi-definiteness (positive definiteness).

So the condition for local optimality at x* for a twice continuously differentiable function f (for simplicity, we just state the unconstrained version) is that $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive semidefinite. If $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite, in the same setting, then these conditions are sufficient and x* is indeed a local minimum.

Remark: The fact that there is no necessary and sufficient condition of this type can be verified by examining the following functions defined on R, x^4, x^3, $-x^4$ etc. at x = 0, where in each case, the second derivative (in this case a number) is zero at x = 0, which could be a minimum, maximum, neither a minimum nor a maximum, etc.

Some exercises

- K.Deb has suggested exploring the Himmelblau function $(x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$ for which you should find all stationary points and local minima and also explore the performance of the steepest descent and other methods from various starting points.

- A simple problem that you should be able to solve completely is of the following type (B and C prob 3.4). A cardboard box of dimensions l, w and h has to be designed with volume 1.67 ft3, and to minimize the amount of cardboard material (there are flaps on the top and bottom with dimensions length/2 and width/2 to enable the box to be closed completely). Find dimensions that would achieve this.

- A problem related to optimization is that of finding the roots of nonlinear functions or solving a system of nonlinear equations. Consider the function f which takes vectors in R^n to vectors in R^n. Representing f as $f(x) = [f_1(x), f_2(x), \ldots, f_n(x)]$, formulate an optimization problem and relate the stationary points x* of that optimization problem to roots of f, i.e. points which satisfy $f(x) = 0$.

For systems of linear equations, you would recall the existence and uniqueness results for a system $Ax-b = 0$, for different m x n dimension matrices A **[pl revise this material]**. Extend this understanding to the system of non-linear equations $f(x) = 0$.

- A problem that is quite important in engineering is the (linear) least squares problem, which you may have encountered in curve fitting and other applications. The problem is of the form $\text{Min}_x \|Ax - b\|_2$ for some given matrix A and constant vector b (relate this to the curve fitting and regression models that you may be familiar with – i.e. where). Write down necessary conditions for optimality and also find a way to find the optimal solution.

This problem has numerous applications in statistics, signal processing and other areas.

- Some problems of infinite dimension can be discretized and solved using the methods above (see example 3.12 (iv) of B and C and others).

Newton's method

The pure Newton method (without line search) for finding the minimum of a function f defined on R^n is based on constructing a quadratic approximation of the function at every stage, and taking the minimum of that quadratic function as the next iterate as follows:

Given the iterate xk, construct the model function
$f(x^k + h) = f(x^k) + \nabla f(x^k)^T h + \frac{1}{2} h^T \nabla^2 f(x^k) h$
and find h to minimize the RHS, This is given by $h^k = -\nabla^2 f(x^k)^{-1} \nabla f(x^k)$, so that the next iterate x^{k+1} is given by $x^k + h^k$, i.e. $x^k - \nabla^2 f(x^k)^{-1} \nabla f(x^k)$.

Local convergence

It can be shown that near a minimum the Newton direction is not only a descent direction, but is well scaled and does not need a line search to be performed and the convergence to the minimum is quadratic (i.e. the errors decrease at a quadratic rate, which is very fast). But this is valid only near a minimum (i.e. if x^k is "close" to x*). This is called local convergence.

Drawbacks of the Newton method

The two major drawbacks of the Newton method are the following:

- We may not get descent (and convergence) if we start far away from the minimum x*. This can be addressed by taking a positive definite matrix that is "close" to the real inverse of the second derivative matrix in the Newton method and then doing a line search at each iteration.

- The second difficulty is that it is numerically expensive and tedious to compute the second derivative matrix (n^2 function evaluations) and its inverse at every stage. Since anyway, we are developing iterative methods, approximations to this would suffice, provided we get global convergence eventually.

Quasi Newton methods address all the above concerns.

Some suggestions for independent project work.

Those who are interested in using Mathematica can explore the book Practical Optimization Methods using Mathematica by M.Asghar Bhatti, Springer, 2000 in the library (or other references) and present a paper based on that.

Those interested in Matlab exercises can follow the book of Chong and Zak and also see the Matlab tool-box either on the local network or some documentation at
http://www.mathworks.com/access/helpdesk/help/pdf_doc/optim/optim_tb.pdf

The book by Belegundu and Chandrupatla has a CD with Fortran code and K.Deb's book also has sample code for several implementable optimization methods.

Those who are interested in applications may please note that modeling the problem appropriately is an important and non-trivial step. Modeling, at the least, involves the following steps:

- Deciding on a set of design parameters
- Quantifying an appropriate objective function for the problem (in terms of the design parameters and perhaps other constants)
- Identifying and quantifying constraints (in terms of the design parameters and perhaps other constants)

  Note that in problems with many objectives, this could be a multi-step exercise where some of the objectives could be turned into constraints either with target values or with bounds. If this is to be done systematically, one way to do it is through goal programming.)

Modified Newton methods

In engineering design problems, function evaluation itself can be expensive. Derivative calculations, especially if done numerically are expensive (but required, at least to verify optimality!). Second derivative calculations in this context are a large computational

burden (n^2 function evaluations to construct the Hessian matrix of second derivatives, and then solving a system of linear equations to get the Newton direction turns out to be a big effort). If we add to this the fact that after all this trouble, the Newton direction (far away from a minimum) may not even be a descent direction, then it is clear that some modifications are called for.

[Try to check even with simple functions of one/two variable(s) that [a] the Newton direction may be a descent direction, but may not be well-scaled and [b] with functions of two variables that it may not even be a descent direction when a point is far from the minimum. Constructing such examples is an illuminating exercise in itself.]

Levenberg-Marquardt method: One modification is the Levenberg-Marquardt modification, where a positive multiple λ of the identity matrix is added to the second derivative matrix to get a positive definite matrix $B_k = \nabla^2 f(x^k) + \lambda I$, which is then inverted to get the descent direction $-B_k^{-1}\nabla f(x^k)$. Note that taking λ to be very large positive essentially amounts to the steepest descent direction and taking λ to be zero gives the Newton method. We essentially keep track of the eigenvalues of $\nabla^2 f(x^k)$ at every stage and see that it remains positive definite, so as to give descent. If not, we add a suitable multiple as above and proceed. This also converges globally, and close to the minimum, essentially reverts to the Newton formula. Note that eigenvalues are computed through efficient factorizations and not by finding the roots of the characteristic polynomial, which is an unstable numerical computation. This gives descent, and a robust algorithm, but the problem of computing $\nabla^2 f(x^k)$ and its inverse at every stage remains.

Conjugate direction methods: Recall that conjugate direction methods worked well (starting with finite n step convergence for quadratic functions and suitable extrapolations for non-linear functions) without having to do second derivative calculations and matrix inversion. They are in between the steepest descent method (very few calculations at every step, but can take a large number of iterations and can have unacceptable numerical performance) on the one hand and Newton's method (considerable effort at each step, but locally very effective in the number of steps required to converge) on the other.

Motivating ideas for Quasi Newton methods: Quasi Newton methods, which are currently the most robust and effective algorithms for unconstrained optimization, are based on the following set of ideas.

- If B_k is positive definite, the direction $-B_k^{-1}\nabla f(x^k)$ is always a descent direction at x^k, and we can perhaps get global convergence (i.e. convergence starting anywhere) by searching in those directions.
- As long as B_k approximates the second derivative matrix at least asymptotically, the method is likely to work well locally (i.e. fast convergence).
- For a quadratic function, a set of conjugate directions, when searched sequentially, gives the optimum solution in at most n iterations.

- In terms of numerical computations for the inverse of a matrix (remember that eventually a system of equations is solved using the second derivative matrix to get the Newton direction, which is equivalent to inverting that matrix), the following formula is used for a low rank update to a matrix [Sherman-Morrison-Woodbury formula]
  $[A + uv^T]^{-1} = A^{-1} + (1/1+k)\, A^{-1}uv^T A^{-1}$, where $k = v^T A^{-1}u$
  [You can verify this by direct multiplication.]

  Note that if A^{-1} is known, this is much faster than computing $[A + uv^T]^{-1}$ directly.
  This is a rank one update (uv^T is a rank one matrix) of the original matrix A. In particular, $A + uu^T$ is a symmetric rank one update.
- If B_k (approximation to second derivative matrix at step k) is updated by a small rank correction to get B_{k+1} (approximation to second derivative matrix at step k+1), then B_{k+1}^{-1} can be computed easily by the above argument.

Quasi Newton methods put all these ideas together to construct approximations B_k to the Hessian matrix at each stage. Note that some updates work on B_k and update B_k and then find its inverse, whereas some work directly on the inverse of the second derivative approximation (usually called H_k, in text books).

[Refer to Chong and Zak, Nocedal and Wright, Fletcher and other books in your library for more details on Quasi Newton methods and their numerical properties.]

Quasi Newton condition

Consider two successive iterates x_k and x_{k+1}, and let $s = x^{k+1} - x^k$ and $t = \nabla f(x^{k+1}) - \nabla f(x^k)$ for convenience. If the matrix B_{k+1} is to approximate the second derivative matrix at stage k+1, it should ideally satisfy the following condition:

Quasi-Newton condition : $B_{k+1}\, s = t$

Quasi-Newton condition for the inverse : $H_{k+1}\, t = s$

In matrix vector terms, the difference in the gradient for a difference in the variable should be captured by the second derivative. ['Dividing' by the lhs vector, which is strictly forbidden in vector arithmetic, may give you the appearance of a familiar derivative like formula based on finite differences]. All Quasi Newton formulas for B_{k+1} (or for the inverse H_{k+1}) satisfy this QN condition. This still gives a number of options for B_{k+1} or H_{k+1} [note that we only have n equations for the n^2 entries of B or H].

The next condition is that it should be easy to compute the inverse. This gives rise to a set of efficient updates (i.e. either H_{k+1}, given H_k, or B_{k+1} which is easy to invert, given B_k).

Updates for Hk or Bk

Symmetric rank one update : $B_{k+1} = B_k + a(uu^T)$ for some vector u and scalar a, which are determined using the QN condition. This may sometimes work, but in general may not preserve positive definiteness and so may not have descent at every stage, so is not a favoured method.

Symmetric rank two updates : $B_{k+1} = B_k + a(uu^T) + b(vv^T)$ for some vectors u and v and some scalar constants a and b. This now gives a wide choice of methods and two of the most popular are the DFP (Davidon Fletcher Powell) update, which is written directly in terms of H and the BFGS (Broyden Fletcher Goldfarb Shanno) and combinations of these called the Broyden family of updates.

DFP : $H_{k+1} = H_k + (1/s^Tt)\, ss^T - (1/t^TH_kt)\, [H_kt][H_kt]^T$
BFGS : $B_{k+1} = B_k + (1/s^Tt)\, tt^T - (1/s^TB_ks)\, [B_ks][B_ks]^T$

In the BFGS, the inverse updating formula has now to be applied twice to get H_{k+1}, which can be done explicitly. The resulting updates for H_{k+1} in both cases are far more efficient than computing the inverse from scratch. These updates all preserve symmetry and positive definiteness, and result in conjugate directions (in the quadratic case) and satisfy the QN condition. The BFGS seems to be the best general-purpose update in practice.

General QN algorithm for minimizing a function f

Start with x^0 and $H_0 = I$ (approximation to the inverse of the Hessian)
At step k,
1. $d^k = -H_k \nabla f(x^k)$
2. Find α_k so as to (exactly or approximately) minimize $f(x^k + \alpha_k d^k)$
3. $x^{k+1} = x^k + \alpha_k d^k$
4. Update H_{k+1} (as per various methods discussed above)

Continue until a termination condition is satisfied.

Linear Programming

In case you need some more problems to work on, try the following:

- Find the minimum of the function of one variable $f(x) = \max\{4-3x, 2x-5\}$, by verifying the first order conditions for a local minimum. Note that f is not differentiable at all points.

- [From C and Z] Find a solution of the equation $x^3 - 12.2x^2 + 7.45x + 42 = 0$ (one solution is approximately equal to 11).

- [From C and Z] Find a minimum of the function $f(x) = x^4 - 14x^3 + 60x^2 - 70x$ (one solution is approximately 0.8) – using Golden Section search or Fibonacci search or any derivative based method.

You can use MATLAB to plot these functions and verify your calculations and to code your algorithms as well.

Notes on Linear Programming

LP is part of self study. You are expected to know the basic formulation, the algebraic simplex method (preferably with matrix vector notation and using simplex tableau or other schemes – any O.R. or optimization book will contain this material such as Belegundu and Chandrupatla, Deb (Appendix), Chong and Zak (chapters 15-17), S.S.Rao etc.). A very quick summary of the important results is listed here.

LP in standard form

A linear programme (in standard form) is an optimization problem of the form

$\text{Min}_x\ c^T x$ subject to $Ax = b,\ x >= 0$

where A is a given m x n matrix (m <= n), b is a given m-vector (the RHS vector) and c is a given n-vector (the cost vector). The n-vector x is to be determined.

Note: Maximization can be handled just as easily as minimization, and the case of inequality constraints and unrestricted variables can be transformed to problem instances of the standard form.

Feasible region

The feasible region $K = \{x : Ax = b, x >= 0\}$ is a convex set in R^n, i.e. $y \varepsilon K$, $z \varepsilon K$ implies that $\{\alpha y + (1-\alpha)z\} \varepsilon K$ for all $\alpha \varepsilon [0,1]$. K is a polyhedral set (could be unbounded) in R^n. A point $x \varepsilon K$, where the only way $x = \alpha y + (1-\alpha)z$ with y,z e K, is for x=y=z, is called an extreme point of K. These correspond to vertices of the polyhedral set.

An m x m invertible submatrix B of the constraint matrix A is called a basis of A. If $B^{-1}b >= 0$, then the solution $x = [x_B, x_N] = [B^{-1}b, 0]$ is called a basic feasible solution **(bfs)** of the LP. The variables corresponding to x_B and x_N are called basic and non-basic variables respectively. **You can verify that such basic feasible solutions account for all extreme points or vertices of the feasible region.**

A vector d is called a direction of the constraint set K, if $x \varepsilon K$ implies that $x + \lambda d \varepsilon K$ for all $\lambda >= 0$. Similar to extreme points, one can define extreme directions (i.e. those that cannot be written as a strict convex combination of two other directions).

Basis representation

It can be shown than any vector $x \varepsilon K$ can be written as a convex combination of the extreme points and a non-negative combination of the feasible directions. This means that if x^i are the extreme points of K and d^j are the extreme directions of K, then any $x \varepsilon K$ can be written as $x = \Sigma_i \lambda_i x^i + \Sigma_j \mu_j d^j$, with $\Sigma_i \lambda_i = 1$, and $\lambda_i >= 0$ and $\mu_j >= 0$.

A solution to an LP, if one exists, is guaranteed to be found at one of the extreme points of K, the feasible region. This is the rationale for the Simplex method of Linear Programming, which goes from one basic feasible solution to another, improving at every stage, till an optimum solution is found.

Algebraic rationale for the simplex method

Let B be a basis corresponding to the bfs $[x_B, x_N]$. Using the constraint equations $Ax = b$ (i.e. $Bx_B + Nx_N = b$), we can write the basic variables x_N in terms of the non basic variables x_B as $x_B = B^{-1}b - B^{-1}Nx_N$.

Substituting this in the objective function $z = c^Tx = c_B^T x_B + c_N^T x_N$, we get the expression $z = c_B^T B^{-1}b + (c_N^T - c_B^T B^{-1}N) x_N$

The sign of the coefficients of x_N tell whether the current basic is optimal or not (if any coefficient is negative, then the objective function can be decreased by increasing the value of x_N from its current value of zero, and so that basis cannot be optimal). **Note that some textbooks refer to the negative of this coefficient and so the sign convention for optimality is the reverse.** These coefficients are called the reduced cost coefficients of each of the variables. Note that the same definition applied to the basic variables gives zero as the reduced cost.

This also tells us how to go to a better solution. Pick a variable whose coefficient above is negative and increase it till feasibility is retained (using the constraint equation and watching for the first basic variable to hit zero – this is implemented by a simple ratio of coefficients called the ratio test in textbooks). This actually then gives a new bfs, and the process continues.

Therefore, we give an informal summary of the Simplex Method below.

Simplex method : for $\text{Min}_x\ c^Tx$ subject to $Ax = b$, $x >= 0$

- Start with a bfs $[x_B, x_N]$ – this is done through a phase 1 procedure, as required **[see below for the phase 1 LP]**

- Check if the current bfs is optimal by checking the sign of the coefficients of the objective function when all the basic variables are eliminated (using the constraint eqns)

- If the current bfs is not optimal, propose a variable to enter the basis (based on the sign of the reduced costs calculated above). A leaving variable is identified as the one that first goes to zero by increasing the entering variable (using the ratio test).

- A new basis is computed by the pivoting operation.

Refer to a proper text-book for the details.

Phase 1 Simplex

A preliminary result is that a LP in standard form has a bfs if it is feasible. [Note that the feasible region $\{(x_1,x_2)$ s.t. $x_2 >= 0\}$ seen as a feasible region in R^2 does not have an extreme point, but in standard form, this system does have a bfs.]

The Simplex method requires a starting bfs. The following is one method to get a bfs (if one exists) for the system $\{Ax = b, x >= 0)$.

Formulate the LP

Min $\mathbf{1}^T x_a$
s.t. $Ax + x_a = b$
$x, x_a >= 0$, where **1** is the m-vector of all ones. It is assumed that $b >= 0$, as usual. The variables x_a are called the artificial variables, one for each constraint.

This LP always has the bfs $[x, x_a] = [0,b]$, so the simplex method for this problem can be initiated.

The main result is that if the solution to this LP is with objective function value > 0, then the original LP has no feasible solution and if the solution to this LP occurs with function value = 0, then (provided the artificial variables are not in the basis, with value = 0, which can happen for a degenerate phase 1 LP).

The phase 1 LP can be computationally combined with the calculations of Phase 2, which is what is done in practice.

Actual implementation of the simplex method is through the revised simplex method, which generates columns corresponding to the entering variable, when required, and keeps a copy of the basis inverse through updations.

Termination and performance

The simplex method (attributed to Dantzig (1947) approx), is guaranteed to terminate in one of three ways:

- Certifying an unbounded optimal value (by identifying a feasible direction in which the objective function can be indefinitely improved),
- Certifying infeasibility of the problem (in phase 1 of the procedure) or
- Terminating in an optimal solution.

The only care that has to be taken is to do with cycling in the case of degenerate solutions (where a number of bfs's correspond to a single extreme point and the solution keeps going from one to other with no improvement). This can be avoided by one of several anti-cycling rules. The simplex method performs well in practice and has been able to

solve large problems in practice, especially when specialized to particular problem structure and using matrix computational enhancements.

However, the simplex method is known to have poor worst-case performance and is not a theoretically satisfactory algorithm. Two classes of algorithms that are theoretically attractive (polynomial time) are the Ellipsoid algorithm, proposed by Khachiyan (1979) and Interior Point algorithms, first proposed by Karmarkar (1984). The interior point algorithm has been enhanced in several ways and now forms a viable alternative to the simplex method. The interior point method relies on a non-linear optimization approach to LP, but with computational enhancements to do with scaling of the feasible region to achieve large step sizes, and other measures.

Success of LP

The success of linear programming as an optimization tool can be attributed to many factors. One is that in a reasonable range of operating values, the abstractions for costs and resource usages can be taken to be linear with respect to operating variables. Even some non-linear but convex cost expressions can be written in piecewise linear form which are then amenable to linear programming formulations. A second reason is that there are now powerful general-purpose codes available to solve 'large' linear programming formulations in a numerically stable ways in such a way that engineers and managers can interpret the results usefully. These codes are based either on simple (extreme point) methods or on barrier (interior point) methods. For LPs with special structure (such as network flow LPs), very large problems can be set up and solved very effectively. A third reason is that the supporting theory of convex analysis and duality is so strong and elegant that a lot of additional mileage is available from a linear programming formulation and its solution. These include sensitivity information and various enhancements to computation and interpretation of solutions. Put together, these make LP among the most powerful tools of optimization theory and also in the practice of Operations Research.

As part of this course, we will see some aspects of LP that are interesting, useful and intriguing, apart from the basic material which you should read up on, in any textbook on Optimization and/or O.R. These notes are one step ahead of the current lectures, but you may please read them now.

Special structure LPs : Network flow LPs

Probably the most important sub-class of practical LPs are Network Flow LPs, which admit a surprisingly large number of specializations to interesting applications. These are defined below.

The underlying structure is that of a directed network. The specification is in terms of a set N of nodes, a set of arcs A, with each arc j having a head node i_1 and a tail node i_2. The main variables are flow variables x_j, defined on each arc j. Typically, flow variables satisfying bound constraints on arcs and satisfying a flow balance condition at nodes are

to be decided so that costs associated with the flow are minimized. In addition, the following data is required to specify the Min cost flow network flow LP.

A cost c_j on each arc j, lower and upper bounds l_j and u_j on each arc j, and the flow divergence or net supply b_i at each node i.

With these, the min cost network flow or the optimal distribution problem is as follows:

$$\text{Min } \Sigma\, c_j\, x_j$$
$$\text{subject to: } \Sigma_{\{j \text{ outgoing at } i\}}\, x_j \;-\; \Sigma_{\{j \text{ incoming at } j\}}\, x_j = b_i \text{ (each i)}$$
$$\text{and } l_j <= x_j <= u_j \text{ (each j)}$$

For more details on such problems, apart from O.R. books, you can consult the book Linear Programming and Network Flows by Bazaraa, Jarvis and Sherali or one of several books on network flows and combinatorial optimization (notice that some of the problems discussed here have a combinatorial flavour, such as the shortest path problem and the assignment problem). Linear programming as a sub-problem in solving large combinatorial problems is one of the most important aspects of LP, in the last 20 years.

Exercises:

- By properly defining the various constants, show that the problem of finding the shortest path from a node s to a node t on a directed network (with distances defined on each arc) can be formulated as a min cost LP. One question which you should think about and perhaps answer later on, is how the (extreme point) solution to this LP relates to the known (discrete search) techniques for this problem. Also formulate the longest path problem on a (acyclic) network, which comes up in Project Management as the critical path, can also be formulated as an LP.

- What is the condition to be satisfied by constants bi at each node to make the network flow LP a meaningful one? Relate this condition and other correspondences to two well known problems from the O.R. literature, namely the Transportation (or Transhipment) problem and the Assignment problem. These are discussed in any standard O.R. book and also in Belegundu and Chandrupatla. Verify that these two problems are special cases of the Min cost flow LP.

- Note that for arbitrary data, the network flow LP may not have a feasible flow. For the usual shortest path problem, transportation and assignment problem verify that the formulation is feasible. What happens to the longest path formulation with positive costs with cycles in the network?

- You can see that the constraint matrix of the min cost flow LP is of a structure that admits easy computation of basic feasible solutions (starting from a feasible one). The constraint matrix is such that each m by m sub-matrix (actually any sub-matrix) that is invertible, has determinant 1 or -1 – these matrices are called

totally unimodular. Verify this if you can (you can prove it by induction or other methods). What do bfs's of this LP correspond to on the network?

- How would you model a network flow problem on an undirected network?

Linear programming and duality

Corresponding to any linear programming, there is another one in the background which is closely connected with it. What a minimization problem (with constraints) achieves as a minimum can be interpreted as the maximum of another problem. This is quite a deep principle, which holds not just for LPs but for a larger class of optimization problems. [One example of this is the following geometric problem. Given a convex set K and a point c outside it, both say in R^n – although again this sort of result is valid in a more general setting – try find a point in K that is closest to c (in the Euclidean norm). This minimum norm problem turns out to be intimately related to the following maximization problem. Construct a (hyper)plane that separates c and K (the hyperplane is defined as some d^Tx and separation is achieved by saying that $d^Tx < 0$ for x in K and $d^Tc >= 0$, i.e. K and c lie on opposite sides of the hyperplane). Find the maximum distance between c and such a hyperplane. This maximization problem turns out to have the same value as the earlier minimization problem.]

Weak duality

One way to derive this is the following. Consider the LP
$Min_x\ c^Tx$ subject to $Ax = b$, $x >= 0$ called the primal problem P.

Multiply constraint i by a constant wi and add the resulting set of equations. You can see that the RHS quantity $\Sigma\ w_ib_i <= c^Tx$ for any feasible x, provided that for each index j, the following inequalities hold $\Sigma\ w_i\ a_{ij} <= c_j$. You can see this by direct arithmetic component-wise and even more easily by matrix vector arithmetic. Note that this bound on the objective function value is true for any x and is therefore true for the optimal solution x* to the primal problem. The quantity given by wTb is a bound on the primal objective function valid with any w satisfying the given constraints $w^TA <= c$ and therefore $\{Max_w\ w^Tb$ s.t. $w^TA <= c\} <= c^Tx^*$ for feasible x* (i.e. it gives a lower bound. The problem in w is also an LP (but with unconstrained variables w).

This result is called weak duality. The problem $\{Max_w\ w^Tb$ s.t. $w^TA <= c\}$is called the Dual LP, and the variables w are called the dual variables. Using this itself, one can show that if the primal P is unbounded, the dual D is infeasible and vice-versa. Note that both primal and dual can be infeasible (both cannot be unbounded from the previous statement).

Strong duality

The obvious question of interest is whether there is indeed a w that achieves this bound (i.e. satisfies the constraints and has $w^Tb = c^Tx^*$). This answer is 'yes' for linear

programmes and some other optimization problems, but not true in general. This can be proved in many ways, and the most direct way for us is if we believe in the finite termination of the Simplex method (in the absence of degeneracy), then at optimality, the coefficients of the basic and non-basic variables satisfy the condition that $(c_j - c_B^T B^{-1} Aj) >= 0$ all j, where A_j is the j-th constraint column of A. Note that for j belonging to the basis, this is zero by definition and for the non-basic variables, this is precisely the optimality test that we saw in the Simplex method.

Now defining $w^T = c_B^T B^{-1}$ constructively gives us a feasible dual vector that satisfies $w^{*T} b = c^T x^*$ **[verify this]**.

The main duality result is that for a linear programme that has an optimal solution, the dual LP also has an optimal solution and with the same objective function value as the primal problem.

Sometimes the dual problem is easier to solve as it may have fewer constraints or may have special structure (e.g. separability).

Dual variables and Complementary slackness

There are several interesting applications and interpretation of dual variables and the dual LP. The values of the dual variables at optimality are of particular significance as they capture the rate of change of the objective function $c^T x^*$ with respect to small changes in the RHS variables b. This is of interest if b_i represents the availability of resource i or other interpretations of the constraint RHS. This result can be seen intuitively from the expression $c^T x^* = z^* = w^{*T} b$ and so taking the partial derivative of z* w.r.t. bi gives the desired interpretation.

The other important linkage of the primal and dual problems is through complementary slackness. For the standard pair of LPs where both the primal and dual have optimal solutions (x* and w*), the complementary slackness condition is that $w^{*T} A_i < c_i$ implies that $x^*_i = 0$, i.e. the i-th constraint in the dual is satisfied with inequality only for zero variables and conversely that $x^*_i = 0$ implies that i-th constraint in the dual is binding. Note that both these conditions are one-way conditions and not if and only if conditions.

A set of feasible vectors x and w are optimal if and only they satisfy the complementary slackness conditions.

Exercises

- You can test your understanding of complementary slackness and dual variables and some of the relevant notation and transformations by writing the dual LP to the inequality constrained LP $Min_x\ c^T x$ subject to $Ax >= b$, $x >= 0$.

- Write the dual of the transportation LP and interpret the dual variables.

- Write the dual of the shortest path LP and interpret the dual variables.

Sensitivity analysis

For the LP in standard form, $\text{Min}_x\ c^T x$ subject to $Ax = b$, $x >= 0$, the questions of interest are the following. How does the optimal solution vary with change in the data parameters A_{ij}, c_j and b_i (especially the latter two, because the 'technology' matrix A does not change as often as prices and resource availabilities and demands – in planning applications). What range of values keeps the same basis, but with different values of the variables at optimality? These are useful in design decisions and some managerial decisions, as data are often not known with certainty or are likely to change with time. These questions are well understood and answered in the case of linear programming.

Subsidiary decisions are whether an LP optimal solution remains optimal if a new constraint is added **(if the original solution satisfies the new constraint, it is optimal. Why?)** and how to restore optimality and feasibility if not, and whether a solution remains optimal if a new variable is added to an LP.

LP and convex programming

A few more notes on Linear Programming and related matters are below.

Dual simplex

The primal simplex method retains feasibility throughout (starting from an initial bfs, got from Phase 1 simplex, as required) and keeps on improving the objective function (i.e. optimality). A dual strategy to this can be followed, which satisfies dual feasibility (i.e. the primal optimality condition) throughout and keeps on improving the dual objective function value (roughly speaking, the primal feasibility condition) till termination. This is called the dual simplex method.

[See a text book for the method. The ratio test is now to get a new basis while retaining dual feasibility.]

This is especially useful when you have an LP of the form Min $c^T x$ s.t. $Ax >= b$, $x >= 0$, where all the b_i and c_j are non-negative, because in this case, $x = 0$ gives a dual feasible solution **[Verify this]**. Such LPs arise for example in cost minimization problems subject to demand satisfaction constraints. Recall that the diet problem primal was of this type.

Another place where the dual simplex logic is useful is when an LP is solved to optimality and then a constraint is added (which may be violated by the optimal solution). Again in this case, we have dual feasibility **[verify this]** and usually, a couple of iterations of the dual simplex are enough to restore primal feasibility (and therefore the solution to the original problem). In relaxation-based approaches, where all constraints are not specified at the beginning, but are introduced by and by (a viable approach for some large problems), this is very useful.

Convex functions

Recall the definition of a convex set. The underlying assumption in most of the discussion so far is that the domain of the function f to be optimized is a convex set. Recall that in the definitions of neighbourhood, or even during line searches, we are implicitly assuming that the function is defined on all line segments between two points that we identify. In particular, if the domain is constrained through specified functions (e.g. $g(x) <= 0$ or $h(x) = 0$), we will see later how to handle optimality conditions and algorithms.

For now, it is important to note that local optimality of f is not equivalent to global optimality, in general. One case where it is so is when f is a convex function. The epigraph epi(f) of a real-valued function f defined on convex set K, a subset of Rn, is the n+1 dimensional set {y,x} satisfying $y >= f(x)$. Then, f is called a convex function if epi(f) is a convex set.

For a convex function, apart from the definition above (which links up the terms convex sets and convex functions through the epigraph) there are several equivalent definitions. Some of them are:
f is convex if

1. $f(\lambda x + (1-\lambda)y) >= \lambda f(x) + (1-\lambda)y$ for $0 <= \lambda <= 1$ and $x,y \in K$.
2. (If f is differentiable) $f(y) >= f(x) + \nabla f(x)^T(y-x)$, for $x,y \in K$.
3. (If f is twice-differentiable) $\nabla^2 f(x)$ is positive semi-definite for all $x \in K$.

Note that linear functions are convex.

The main result is that for a convex function f (defined over a convex set K), a local minimum x* is also a global minimum. **[Try to show this].**

Convex programming

If the set K is specified explicitly using constraint functions $g_i(x) <= 0$, then if the g_i functions are convex and equality functions $h_j(x) = 0$, where the h_j functions are linear, then K is a convex set **[verify this]**. Minimization of a convex function over such a convex set is called a convex programming problem and the global minimum result holds here as well. LP is of course an example of this.

This nice fact regarding global minima (which is due to the structure of the function and the constraint set) has nothing to do with any of the algorithms we have studied, which still are aimed at finding local minima. Convex feasible sets make it easier to develop algorithms where we retain feasibility throughout.

Some odd things to ponder about in case you have trouble sleeping

- What is a simplex? [as in 'simplex' method]. Hint – it is not the opposite of 'complex'. It is also not hostel slang for 'easy' although by now, I am sure you would find the simplex method easy.

- Concave functions are defined in a manner similar to convex functions (e.g. f is concave if –f is convex, or by defining the hypo-graph similar to the epigraph). However, there is nothing like a concave set. A set is convex or it is not convex.

- Why is a Linear Programme called so (i.e. why is it called a 'Programme')? Hint – it has nothing to do with a computer 'programme'.

Constrained optimization

Constrained optimization, for us, deals with the minimization (or maximization) of an objective function f(x) – say x is an n-dimensional vector, subject to x belonging to a constraint set K. The set K is defined by constraint equalities ($h_j(x) = 0$, j = 1, …, n) and constraint inequalities ($g_i(x) <= 0$, i = 1, ..., m). **[Try to think of situations where constraints are NOT specified in this manner and what we would be able to do then.]**

Note that $\{x : g_i(x) <= 0\}$ defines a convex set for g_i convex. For this reason, equality constraints are generally meaningful only when they are linear. For non-convex feasible regions, algorithms are difficult to design, because feasibility of different iterates itself could be a difficult thing to achieve. The ideas here are relevant to non-convex feasible regions, but the strongest results are for convex programming problems, where a convex function is minimized over a convex set. Here, apart from feasible direction algorithms, we have global optimality and a well-developed duality theory.

Some simple examples

To test your intuition and basic understanding of constrained optimization problems, try your hand at the following problems.

Min $(x_1 – 2)^2 + (x_2 – 2)^2$ subject to $x_1 + x_2 = 1$
Min $(x_1 – 2)^2 + (x_2 – 2)^2$ subject to $x_1 + x_2 <= 1$
Min $(x_1 – 2)^2 + (x_2 – 2)^2$ subject to $x_1 + x_2 <= 1$
Min $x_1 + x_2$ subject to $x_1^2 + x_2^2 – 2 = 0$
Min $x_1 + x_2$ subject to $x_1^2 + x_2^2 – 2 <= 0$

Looking at these examples, see if you can derive a set of first order optimality conditions, which any candidate point x* must satisfy. At regular time intervals (e.g. every two days), recall that LP is an example of constrained optimization, so you should constantly re-interpret the results for LP.

Equality constraints

Consider a problem with a single equality constraint. Min f(x) s.t. h(x) = 0. Verify that for x* to be optimal, we must have $\nabla f(x^*) - \mu \nabla h(x^*) = 0$ for some μ. Verify that any point x* along with some μ that satisfies $\nabla f(x^*) - \mu \nabla h(x^*) = 0$ cannot satisfy $\nabla f(x^*)^T d < 0$ and $\nabla h(x^*)^T d = 0$ for any d. In fact, if the condition $\nabla f(x^*) - \mu \nabla h(x^*) = 0$, is not satisfied, it is easy to see that, we can find a d so that $f(x^*+d) < f(x^*) + \nabla f(x^*)^T d$ (i.e. where $\nabla f(x^*)^T d < 0$) and $\nabla h(x^*)^T d = 0$, so the point x*+d is feasible and has lower objective function value, so x* cannot be optimal.

This argument can be extended to multiple equality constraints ($h_j(x) = 0$, j = 1, …, n) by saying that $\nabla f(x^*) - \Sigma_j \mu_j \nabla h_j(x^*) = 0$. One side is clear (if it satisfies this condition then the first order feasibility/optimality condition is satisfied). The other part, that one can actually find a better point x* + d if the condition holds, requires a regularity assumption (e.g. linear independence of the constraints at x*).

Inequality constraints

Consider a single inequality constraint, i.e. Min f(x) s.t. g(x) <= 0. If x* is a candidate solution, what condition must it satisfy?

First of all, notice that if x* satisfies $g(x^*) < 0$, then for x* to be optimal, we must have the familiar first order optimality condition, $\nabla f(x^*) = 0$. This is because we can always restrict ourselves to a neighbourhood of x* where we remain feasible with respect to g(x) (using the first order expansion $g(x^*+d) = g(x^*) + \nabla g(x^*)^T d$, which is valid for small ||d||, since $g(x^*) < 0$, we can ensure $g(x^*+d) < 0$ in the entire neighbourhood). This means that we can essentially ignore the constraint g as far as local optimality is concerned, and we need the unconstrained optimality condition $\nabla f(x^*) = 0$ to hold. **[Verify the details.]**

If a candidate point x* satisfies $g(x^*) = 0$, then directions d which satisfy $\nabla g(x^*)^T d <= 0$ will retain feasibility for small movements in that direction. Also, if such a direction d also gives decrease in f, i.e. $\nabla f(x^*)^T d < 0$, then we have a direction (and therefore a point in a neighbourhood of x*) which is both feasible and better than x* (i.e. less value of f). This would mean that x* is not optimal. Therefore for x* to be optimal, there cannot be a vector d in set $K = \{d : \nabla g(x^*)^T d <= 0, \nabla f(x^*)^T d < 0\}$. We can show that, in general, this means that there must be a non-negative λ, such that $\nabla f(x^*) + \lambda \nabla g(x^*) = 0$. If this condition holds, then notice that multiplying through by any d will disqualify it from lying in K, ie. Set K is empty. The other way also can be shown.

For more than one inequality constraint, we can show that the first order necessary condition is $\nabla f(x^*) + \Sigma_i \lambda_i \nabla g_i(x^*) = 0$, with the added conditions that the λ_i are >= 0, and only those constraints will appear with non-zero values which are binding, i.e for which $g_i(x^*) = 0$. This is usually written as the complementary slackness condition $\lambda_i g_i(x^*) = 0$.

Again, one direction is easier to show, that is, if x* (along with some multipliers) satisfy

the condition above (and of course x* is feasible), then one can show – using arguments like the ones above – that there cannot be any direction d which retains feasibility as well as descent in f. This means that x* is locally optimal. To show that a locally optimal x* must satisfy these conditions requires an additional assumption/condition discussed below.

Karush Kuhn Tucker (KKT) conditions

The first order necessary conditions for optimality in constrained optimization problems involving both equality and inequality constraints are usually stated as the KKT (Karush Kuhn Tucker) conditions as below. For the problem Min f(x) s.t. ($h_j(x) = 0$, j = 1, …, n) and ($g_i(x) <= 0$, i = 1, ..., m), if x* is optimal, the following conditions must hold (in the presence of a regularity condition to be explained later – this is the slightly annoying part of this topic – but do not worry too much about it at this point!). In what follows, there is a multiplier λ_i corresponding to each inequality constraint i and a multiplier μ_j corresponding to each equality constraint j.

$\nabla f(x^*) + \Sigma_i \lambda_i \nabla g_i(x^*) + \Sigma_j \mu_j \nabla h_j(x^*) = 0$
$g_i(x^*) <= 0$ all i
$h_j(x^*) = 0$, all j
$\lambda_i g_i(x^*) = 0$ all i
and $\lambda_i >= 0$ all i.

The second and third conditions are just feasibility conditions on x*. The fourth refers to the non-negativity condition of the multipliers corresponding to the inequality constraints and the fifth condition is the complementary slackness condition.

[To start with, apply these conditions to our standard form LP and see what you get. Also apply them to each of the simple sample problems stated in the beginning.]

Constraint qualification and regularity conditions

The necessary conditions for constrained problems are not quite as neat as in the unconstrained case. An example will illustrate the difficulty. Consider the problem Min x_1 subject to the constraints $x_2 <= x_1^3$ and $x_2 >= 0$. You can verify that [0,0] is the solution to this problem, but that the KKT conditions cannot hold at that point. We will discuss this later.

These notes finish some basic material on necessary and sufficient conditions for constrained optima. The remaining part of the course will consist of Quadratic Programming, use of QP to solve general nonlinear constrained problems, and finally, use of some non-traditional techniques to solve optimization problems. I will look at course project proposals if they are given to me on Monday, not after that.

Theorems of the alternative

To show the validity of the Karush Kuhn Tucker first order conditions for optimality of constrained optimization problems, there are a number of possible approaches. One of them is based on a set of (classical) results called theorems of the alternative. Examples of this are Gordan's theorem and Farkas's lemma. They are basically of the form that a certain system of (linear) equalities or inequalities has a feasible solution if and only if some other system has no solution.

One version of Farkas's lemma refers to the following two systems (exactly one of which has a solution):
System 1 : $\{x : Ax = b, x >= 0\}$
System 2 : $\{y : y^T A <= 0, y^T b > 0\}$

These theorems have nice geometrical interpretations, and some of them can be seen as separation theorems in convex analysis. For example, in the above system, consider the columns of A as vectors. The first system says that vector b lies in the positive cone generated by the columns of A (here x_i refers to the weight of column i of A). The second system says that we can find a plane that separates the columns of A and the b vector (then y represents the normal vector that defines the separating hyperplane, which makes an obtuse angle with the columns of A and an acute angle with b).

Alternative interpretations of the KKT conditions

Another way to look at the KKT condition is to linearise the objective function and constraints – which is valid if a constraint qualification holds (to be discussed) and apply linear programming duality. Conversely, one can derive LP duality as an application of the KKT conditions.

If we define the Lagrangean function $L(x,\lambda,\mu) = f(x) + \lambda^T g(x) + \mu^T h(x)$, the main KKT condition can also be viewed as the gradient (w.r.t. x) of the Lagrangean being set equal to zero for optimality, rather than just the gradient of the objective function.

The Lagrange multipliers λ and μ at optimality play the same roles as the optimal dual variables in Linear Programming and have the interpretation of shadow prices of resources (i.e. marginal change in optimal objective function value for unit change in right hand side of constraints). Apart from other things, this is consistent with the complementary slackness condition that the marginal change is zero for a constraint that is not binding at optimality.

Constraint qualifications

For a constraint set K, define a feasible sequence at x* as a sequence of vectors $\{x^k\}$ such that x^k not equal to x*, lim x^k = x* and x^k belonging to K for k sufficiently large. Then a limiting direction of this sequence is lim $(x^k - x^*)/\| x^k - x^* \|$ in a suitable norm.

The local condition for optimality (say local minimum) of f at x* over set K is that $\nabla f(x^*)^T d >= 0$ for all limiting directions d. **[Verify this from the first order expansion**

for f]. This condition is difficult to check. What is more reasonable is the following. Suppose K is defined by the constraint set $\{g_i(x) <= 0, i = 1, \ldots, n$ and $h_j(x) = 0, j = 1, \ldots, m\}$. The linearised set of feasible directions at a feasible point x* is the following: {d such that $\nabla h_j(x^*)^T d = 0$ and $\nabla gi(x^*)^T d <= 0$ for active constraints i}. A constraint qualification is said to hold at x* if these sets are equal (i.e. the set of limiting directions and the linearised set of feasible directions).

It is easy to see that if the KKT conditions hold, (local) optimality must hold. A constraint qualification (or regularity condition) basically allows us to derive a set of multipliers whenever optimality holds. The proof of that requires a version of the implicit function theorem or the use of generalized inverses.

Constraint qualifications are of several different types. Two which are commonly applicable are (i) Linear independence of active constraint gradients at x* or (ii) Linear constraints at x*. There are many others, and you can refer to a book on non-linear programming for details.

Examples where constraint qualification does not hold

$\{(x_1,x_2)$ s.t. $x_2 >= 0$, $x_2 <= x_1^3\}$. For this set at [0,0], the direction [1,0] is the only limiting direction but the set of linearised feasible directions is $\{d: d_2 = 0\}$. This set also contains [-1,0], which is not a limiting direction. In such cases, the KKT conditions for an optimization problem over this set may not hold. **[Verify this, and see what the implications are for, say, $f(x) = x_1 + x_2$, which attains its minimum at [0,0].** Another example is $K = \{(x_1,x_2)$ s.t. $x_1 >= 0$, $x_2 >= 0$, $x_2 – (1- x_1)^3 <= 0\}$.

An example with a different flavour is $K = \{(x_1,x_2)$ s.t. $(x_1^2 + x_1^2) = 1$, $(x_1 + 1)^2 + x_2^2 = 4\}$. **[Exercise: Try finding an objective function for which the KKT conditions will fail for this example.]**

Despite this technicality, the KKT conditions are very useful to characterize optimality in a majority of cases.

Examples

Try the following examples for practice. Note that in all cases, we are looking for locally optimal solutions.

- Min $(x_1 – 3/2)^2 + (x_2 – 1/8)^4$ s.t. $x_1 + x_2 – 1 <= 0$, $x_1 – x_2 – 1 <= 0$, $x_2 – x_1 – 1 <= 0$, $- x_1 – x_2 – 1 <= 0$
- Max $x_1x_2x_3$ s.t. $x_1 x_2 + x_2 x_3 + x_3 x_1 = A/2$ (this is the problem of maximizing the volume of a box of fixed area)
- Max x^TQx s.t. $||x||_2 = 1$
- Min $–2 x_1 + x_2$, s.t. $x_2 – (1- x_1)^3 <= 0$ and $1 – x_2 – 0.25 x_1^2 <= 0$
- Min $x_1 x_2$ s.t. $x_1^2 + x_2^2 = 1$
- Min $–0.1(x_1-4)^2 + x_2^2$ s.t. $1 – x_1^2 – x_2^2 <= 0$
- LP in standard form (and its dual)

- Symmetric form of the LP (and its dual)
- Min ½ $x^T Q x - d^T x$ s.t. $Ax = b$ (assume that Q is symmetric positive definite)
- Min ½ $x^T Q x - d^T x$ s.t. $Ax <= b$ (assume that Q is symmetric positive definite)

Second order conditions

Second order necessary and sufficient conditions are stated here. The most convenient way is to state them in terms of the Hessian of the Lagrangean function at optimality. For a point x* and the associated set of multipliers λ^*, μ^* which satisfy the first order conditions, the matrix $\nabla^2_{xx} L(x^*, \lambda^*, \mu^*)$ is positive semidefinite on an appropriate set of directions. This set of directions is defined as $\{d : \nabla h_j(x^*)^T d = 0$ for equality constraints $h_j = 0$, $\nabla g_i(x^*)^T d = 0$ for i defining active constraints and for which the multipliers $\lambda_i^* > 0$ and $\nabla g_i(x^*)^T d >= 0$ for i defining active constraints for which the multipliers $\lambda_i^* = 0\}$. See Chong and Zak and other books on nonlinear optimization for the details.

In the inequality constrained case, when the multipliers λ^* at optimality are unique and if strict complementarity holds, this leads to the checkable condition that $Z^T \nabla^2_{xx} L(x^*, \lambda^*) Z$ is positive semidefinite, where Z is a full rank matrix spanning the null space of the active constraint gradients at x* (this matrix can be computed).

Generally speaking, sufficient conditions are when the Hessian (i.e. second derivative matrix) of the Lagrangean is positive definite over the appropriate subspace (set of directions).

Note that it is not necessary that just the Hessian of the objective function be positive semi-definite over the set of directions and that the given condition is a weaker condition than that.

Quadratic Programming and Sequential Quadratic Programming

The Quadratic Programming (QP) problem is the following:
Min ½ $x^T Q x + d^T x$
s.t. $Ax >= b$
i.e. the minimization (or maximization) of a quadratic function of n variables subject to linear inequality constraints.

This formulation includes equality constraints as well (in fact, we will see that problem first). This problem in general, has a structure similar to Linear Programming – you will see that if you write down the KKT conditions for LP and QP.

An unconstrained QP makes sense (unlike in LP). Analysing the unconstrained QP tells us that QP can have local optima that are not global optima (also stationary points that are neither minima nor maxima). This will of course carry over in the constrained case as well. So QP will have all the characteristics of general non-linear problems, but with the feature that the convex QP (with positive definite matrix Q) will admit LP like solutions. The constraint region of QP is convex.

QP is important in its own right as well as for the reason that it forms that basis for the most successful techniques for general non-linear programming problems (Sequential Quadratic Programming and variants are generally regarded currently as the best methods for general NLP).

Equality constrained QP (EQP)

Equality constrained QP (min Min ½ $x^T Q x + d^T x$ s.t. $Ax = b$, with Q an n dim matrix, A an m x n constraint matrix, with $m < n$) is easy to solve mathematically, but has some subtleties computationally speaking. Verify that if Q is non singular, and A is full rank (i.e. rank m), then the equality constrained QP amounts to solving an m+n dimension square, nonsingular system of linear equations for the x values and the multipliers. Alternatively, we can think of the m constraints being used to eliminate m of the variables and then solving the resulting unconstrained quadratic function minimization.

Example: $3x_1^2 + 2\,x_1x_2 + x_1x_3 + 2.5\,x_2^2 + 2\,x_2\,x_3 + 2\,x_3^2 - 8\,x_1 - 3\,x_2 - 3\,x_3$ s.t. $x_1 + x_3 = 3$, $x_2 + x_3 = 0$.

The solution of this problem is $x^* = [2, -1, 1]$. **[Verify that this satisfies the KKT conditions. How do you interpret the multipliers in this case? Using the multipliers, decide whether the solution is optimal if the first constraint was $x_1 + x_3 <= 3$? For the case $x_1 + x_3 >= 3$?]**

QP is therefore a finite problem, in that the solution can be found by enumerating a finite number of possibilities with finite computations for each possibility. This is seen by allowing a certain number of constraints to be active (including the possibility of none of the constraints being active) and solving the equality constrained QP in each case. Of course, this would not be a practical way to solve QPs, but it is (like) LP at least a finite problem.

If the Q matrix is not positive definite (for a minimization problem), the objective function is not convex. Then combinatorially speaking, QP is a hard problem, and essentially has to be solved by (clever) enumeration, or by heuristics (for large problems). Descent-based methods are not guaranteed to give us the global minimum.

If we allow x to be constrained to integer values (in particular, 0-1 values), then QP includes the hardest known problems in discrete optimization, including the traveling salesman problem and variants.

QP and LCP

A unifying framework which includes QP, LP and some other problems is the Linear Complementarity Problem (LCP). This is stated as follows:

Given a matrix M and a vector q, find a vector z, such that the following conditions are met. $z >= 0$, $q + Mz >= 0$ and $z^T(q + Mz) = 0$. It is easy to define this problem (i.e. define appropriate M and q) so that LP and QP are special cases of LCP. **[Try this.]** The complementarity condition says that either $z_i = 0$ or $(q + Mz)_i = 0$, so trying out various possibilities (each involving a linear system to be solved) will give a solution. A procedure for LCP and therefore QP (Lemke's algorithm) is given in Belegundu and Chandrupatla.

Active set methods for inequality constrained convex QP

The active set method for convex QP is somewhat similar to the Simplex method for LP. Assume that Q is positive semidefinite.

We start with a feasible solution to the system $Ax <= b$ (by using phase one simplex, if required). We maintain a list of constraint indices that are active (where $[Ax]_i = b_i$). Solve the equality constrained QP with these constraints only. This is written as follows: With respect to the current iterate x^k, which has a certain set of active constraints, we can define a direction p^k to be found, which is restricted to that subspace of active constraints. The relevant EQP is

Min_p ½ $p^TQp + (Qx^k + d)^Tp$

s.t. $a_i^Tp = 0$ for all the constraints i that are active at x^k

Here, note that p is the variable in this QP. The term a_i is the i-the constraint row and xk is the current iterate (so the term $Qx^k + d$) is a known one. This EQP is just the minimization of the original objective function, but written in terms of $p = x - x_k$.

If the optimal solution to EQP is $p^k = 0$, look at the multipliers for all the (active) constraints. If they are all non-negative, then the given solution is optimal (verify then that all the KKT conditions are satisfied). If any of the multipliers are negative, say corresponding to constraint i is the least one, drop that constraint from the active set and re-solve the EQP.

If the direction p^k is not equal to zero, do a line search along p^k (this is a very simple line search to ensure that all the linear inequality constraints are satisfied – similar to the ratio test in LP). Re-calculate the active set of indices at this point and continue. Note that step size = 1 is like taking the Newton step that gives the local minimum in the appropriate direction.

Example (from Nocedal and Wright): Min $(x_1 - 1)^2 + (x_2 - 2.5)^2$
s.t. $x_1 - 2\,x_2 + 2 >= 0$, $-x_1 - 2\,x_2 + 6 >= 0$, $-x_1 + 2\,x_2 + 2 >= 0$, $x_1 >= 0$, $x_2 >= 0$.

Starting with the feasible point [2,0], we can start with the active set of constraints and the various iterates of the algorithm will finally terminate at the solution [1.4,1.7]. **[Verify the details. For this example, try different objective functions, including one that gives a minimum in the interior of the feasible region, and verify that the method works. Also try different starting points for this example. Try to write an**

argument for the finite convergence of this algorithm, in the absence of degeneracy – like in LP.]

Belegundu and Chandrupatla provide a version of the active set method for QP, for the special case Q = I, which arises in their general sequential quadratic programming algo. In this special QP, the dual QP is particularly simple to solve (having only non-negativity constraints) and their active set method is specialized to that case.

Exercise: Try the QP Min ½ $x^T x$ s.t. Ax <= b (here x is an n-dimensional variable, A is a given m by n matrix and b is a given m vector). Consider also the dual QP to this one, Max – ½ $y^T(AA^T)y$ – $b^T y$ s.t. y >= 0. Let f_1 and f_2 be the primal and dual objective functions.

Show that the KKT conditions of (P) and (D) are the same.
State and prove a weak duality result for f_1 and f_2 (using techniques similar to those used in LP). Therefore show that if x_0 and y_0 are feasible for (P) and (D) and $f_1(x_0) = f_2(y_0)$, then x_0 and y_0 are optimal for (P) and (D).

Exercise: Solve Min $x_1^2 + 2x_2^2 - 2\,x_1 - 6\,x_2 - 2\,x_1\,x_2$ s.t. ½ x_1 + ½ x_2 <= 1, - x_1 + 2 x_2 <= 2, x_1, x_2 >= 0.

Try different starting points, including the interior of the feasible region.

General non-linear optimization problems

This topic is a big one by itself and a large number of techniques have been proposed in the literature. Here, we only mention a few and give some details on one of the most successful ones, namely Sequential Quadratic Programming (SQP). We also give some introduction to penalty function based methods. Three other methods that are successful in various problem settings are summarized in Belegundu and Chandrupatla (Rosen's gradient projection method for linear constraints, Zoutendijk's method of feasible directions and the generalized reduced gradient method).

Merit functions and penalty functions

A useful concept in constrained optimization problems is that of a merit function. Keeping in mind that there is an objective function (which needs to be minimized, say) and constraints which need to be satisfied, we need to keep both aspects in mind. A merit function is a composite function that includes the objective function and constraints violations, if any, and that can be used to measure the progress of an algorithm and also as a means to perform a line search during iterations.

For the equality constrained nonlinear optimization problem Min f(x) s.t. $h_j(x) = 0$, the quadratic penalty function, $f(x) + (1/\mu)\ \Sigma_j\ h_j(x)^2$ has been proposed as a merit function (for a given parameter $\mu > 0$). This can also used as an objective function in a penalty function based approach, where $(1/\mu)$ represents the penalty parameter for violating the

constraint $h_j(x) = 0$. Typically, this unconstrained problem is solved for a sequence of mk values going down to zero, with each solution used as a starting point for the next problem. It can be shown that the solutions to the unconstrained problem will converge to the constrained solution. Note that typically, all the solutions of the unconstrained problems will be infeasible (will not satisfy h(x) = 0, and that we get a feasible and optimal solution only in the limit.

Exercise: Try the penalty approach for the problem Min $x_1^2 + x_2^2$ s.t. $x_1 + x_2 = 1$, and then for the problem Min $x_1 + x_2$ s.t. $x_1^2 + x_2^2 - 2 = 0$.

Exercise: Think of a way to extend the logic of penalty functions to inequality constrained problems.

Another merit function that is very important is the l_1 merit function, which is defined as $f(x) + (1/\mu)\ \Sigma_j\ |h_j(x)| + (1/\mu)\ \Sigma_i\ |g_i(x)|^+$, where $|g_i(x)|^+$ is max(gi(x),0). This is an exact penalty function because for an appropriate choice of the penalty parameter μ, a single minimization of this will give the solution to the constrained problem. However, this function is not differentiable at all points.

There are other merit functions which are defined based on penalty parameters and also the multipliers or dual variables (suitably defined).

Techniques for general nonlinear problems

In solving general nonlinear programming problems by iterative means, it is seen that subproblems that involve linear constraints are tractable. Nonlinear constraints (even quadratic), are difficult to handle, and themselves would require iterative procedures to solve. It turns out that although linear objective function with linear constraints (which result in LP subproblems) are tractable, they are weak approximations and are generally not effective enough (somewhat akin to steepest descent methods for unconstrained problems). (Convex) quadratic objective with linear constraints are possible to solved efficiently. This is the SQP scheme. Nonlinear (convex) objective functions with linear constraints offer some hope of solutions in reasonable time. These methods also have shown some success.

Sequential quadratic programming SQP

There are many ways to motivate the class of algorithms falling under the SQP scheme. One of the ways is as follows. Consider the inequality constrained NLP Min f(x) s.t. $g_i(x) <= 0$ and some iterate x^k. Replace the objective function and each (active) constraint by a quadratic approximation around x^k and try to find a d^k that gives a locally improved solution (like the Newton scheme).

This gives the quadratic optimization problem
$\text{Min}_d\ f(x^k) + \nabla f(x^k)^T d\ + \frac{1}{2}\ d^T \nabla^2 f(x^k) d$
s.t. $g_i(x^k) + \nabla g_i(x^k) T d\ + \frac{1}{2}\ d^T \nabla^2 g_i(x^k) d <= 0$ for active constraints i

This is difficult to solve (this is not a QP as it has quadratic constraints), but we attempt to write the KKT conditions for this problem. **[Please do this]**. They will include some optimal multiplier values for each constraint.

We then consider a related problem, namely
$\text{Min}_d\ f(x^k) + \nabla f(x^k)^T d\ + \frac{1}{2}\ d^T \nabla^2 f(x^k) d + \Sigma_i\ \lambda_i d^T \nabla^2 g_i(x^k) d$
s.t. $g_i(x^k) + \nabla g_i(x^k)^T d\ <= 0$ for active constraints i

We see that the KKT conditions for this problem are the same as those for the earlier problem (and this is a QP with linear constraints) **[Verify these details]**. The catch of course is that the λ_i values are not known beforehand. But in an iterative scheme, these values are used from the previous iteration (starting with an ad-hoc or intelligent initial estimate).

Once the direction finding QP is solved for d^k, a line search is done (on an appropriate merit function) so that the objective function is reduced, without giving up too much on the feasibility. Note that all iterates need not satisfy the original constraints.

In practical implementations, the second order term in the QP objective function above (which is really the Hessian of the Lagrangean) is approximated by an appropriate positive definite matrix (akin to Quasi Newton methods). Belegundu and Chandrupatla give a version of this where the identity matrix I is used throughout as the approximation to the Lagrangean of the Hessian. This has the advantage that the resulting QP is easier to solve as it has special dual structure, as they illustrate.

Code based on SQP is available in B and C and also in Matlab and other software. B and C also give interesting interpretations of the QP direction finding subproblem.

Discrete Optimization Problems

As an example of discrete optimization problems, try the following problem.

The slot assignment problem

Try to formulate and solve the time-slot assignment problem described below. Courses 1,2,3,4,5,6 and 7 are running next semester. Instructor K is teaching courses 4 and 5 and the same room is required for courses 4 and 6 (so those pairs of courses cannot run in the same time slot). Three time slots 1, 2 and 3 are available for the seven courses. Based on student pre-registration information, the following pairs of courses have some number of students interested in both courses:

CoursePair Number of students

Course pairs	Number of common registrants
1-2	2
1-3	4

1-4	1
2-4	2
2-5	4
3-4	5
3-6	4
5-6	3
5-7	4
6-7	4

Allot time slots 1,2 and 3 to courses 1, …, 7 so as to minimize the number of student clashes. Your procedure should be generalizable to larger instances of this problem (i.e. finally, you should propose a general algorithm and apply it to this problem as an illustration).

Example: A simple, single-pass algorithm could be this. Number the slots 1, …, m. Take each course from 1, …, n in turn. To each course, allot the first slot that will avoid any conflict with slots already allotted. If conflict cannot be avoided, choose a slot that will minimize conflict with courses already allotted.

For this problem, this will lead to the solution:
Course 1 : Slot 1
Course 2 : Slot 2
Course 3 : Slot 1
Course 4 : Slot 3
Course 5 : Slot 1
Course 6 : Slot 1
Course 7 : Slot 2
leading to a total conflict of 3.
This can be encapsulated as the solution [1, 2, 1, 3, 1, 1, 2]

We can try local improvements, by considering a slot change for each course, one at a time. But this leads to no improvement in this case.
However, the solution [1, 2, 3, 2, 3, 1, 2] is better, with total conflict 1.

Now consider the structure of this problem. Let i = 1, .., n be the courses that have to be offered. Slot numbers j = 1, …, m are available. Let $n(i_1,i_2)$ be the number of students wanting to register for the course pair (i_1, i_2). In terms of this, formulate a decision variable for the allotment of slots to courses and an objective function and constraints to capture the decision that you have to make. Also model the instructor constraint (i.e. two courses which are being taught by the same instructor are not to be assigned the same time slot).

Suppose the vector $z = [z_1, z_2, \ldots, z_n]$ represents a possible solution, where z_i represents the slot assigned to course i. How many possible solutions are there to this problem, in general?

Define a neighbour of z as another slot assignment where the slot assignment of one course is different. Develop a local descent method and then a simulated annealing method based on this definition of a neighbourhood of a given solution. Apart from the usual parameters of a simulated annealing algo (the acceptance probability and the cooling schedule), you would have to also define a way of sampling the neighbourhood to get a candidate solution and also a way to include other constraints (like the instructor constraint). Implement this procedure on this example.

Can you think of any other neighbourhood structure that would be useful for this problem? For any negihbourhood structure that you define, are all solutions 'connected' (i.e. is it possible to reach the optimum solution starting from anywhere, through a sequence of neighbours)? If so, what is the maximum number of steps that would be required in the most optimistic case? Thus may give some idea of the number of uphill moves that may have to be made!

How would you generate a good starting solution to this problem?

Formulate a genetic algorithm approach to this problem (including the encoding of the solution space).

As a follow up thought (not part of this course), you can make a plan for how such a decision would be implemented in practice. Some points to consider are: what decisions are taken prior to the taking up of this one, what data to collect for this decision, how to pre-process that data to set up this decision problem, how to decide on the objective function and constraints, how to present the results of this decision making to concerned people, etc. Some of these would obviously require some technical work in programming and interfacing with other decision systems (e.g. the registration process in the university) and some would require some (subjective) judgements based on talking to different people. The real life version of this problem is quite complicated.

Simulated Annealing

Basic descent algorithm.

In what follows, we give a general description of a descent algorithm and a simulated annealing algorithm. This could be applied to any function f, defined on discrete or continuous domains and with no assumptions on differentiability, convexity or other properties of f.

Given a function f to be minimized, start with iterate x^0.
Define a neighbourhood $N(x^0)$ and select a $x \in N(x^0)$
Accept x if $f(x) < f(x^0)$ and then redefine $x^0 = x$
Repeat
Stop when there is no such x, in which case x^0 is a locally optimal solution

[Propose a modification of this when the solution is restricted to set K.]

Notes:

- We need to define the notion of a neighbourhood. This is done using norms in a continuous space and needs to be done differently for a discrete solution space.
- We need to define a way of getting an x from $N(x^0)$. This can be done enumeratively or through some randomized procedure.
- To conclude that there is no x that improves f, we need a computable mechanism **[what is the mechanism in the usual differentiable case?]**. For the discrete case, when the neighbourhood is of small finite (low order polynomial in n, the dimension of the decision vector), we do it by direct enumeration in some manner.
- Tabu search methods retain some memory of how an x^0 was arrived at and try to avoid repetitive explorations of the same region, by restricting the x that can be selected.

Simulated annealing (SA) algorithm

This can be viewed as a modification of the basic algorithm, where ascent is occasionally permitted, with probability inversely proportional to the extent of ascent, and in any case getting smaller and going to zero as the algorithm progresses. These in fact form the major parameters of the SA algorithm, viz., the probability function and the cooling schedule, both of which are controlled through a parameter referred to as the Temperature, in keeping with the analogy with annealing.

The prototype algorithm is the following:

Given a function f to be minimized, choose parameters t_0 and a function α
Start with iterate x^0.
Define a neighbourhood $N(x^0)$ and select a x e $N(x^0)$ at random
Accept x if $f(x) < f(x^0)$ and then redefine $x^0 = x$
Accept x if $f(x) > f(x^0)$ and the following condition is satisfied
    Pick random p in [0,1] and the current value of t (initially t_0)
    Check if $p < \exp(f(x0) - f(x))/t)$
Repeat
Redefine $t = \alpha(t)$
Stop when there is no such x, in which case x^0 is a locally optimal solution, or when an iteration count is reached.

Notes:

Apart from the earlier ones, which are still relevant, we make the following points.

- The acceptance function for an uphill move is derived from principles of statistical thermodynamics, but any simpler function can be chosen provided it decreases with the temperature parameter and if it is inversely proportional to the extent of the function ascent.

- It is important to select the sample point at random (and not guided solely by a descent criterion). This random selection can be done in different ways. For a continuous solution space (say $x^0 \varepsilon R^n$), the simplest one is to perturb the co-ordinates of x^0 by some random amount drawn from a symmetric distribution centred around zero. For discrete solution spaces, the randomization has to be defined specific to the setting.
- The function $\alpha(t)$ is a decreasing function of t and is usually selected as at for a in the range 0.8 to 0.99. Other cooling schedules have been successfully used.

Convergence of the SA algorithm: An (non-rigorous) argument for the convergence of the simulated annealing algorithm can be as follows. Discretize the solution space to create a set of states representing the feasible region (to a desired accuracy). From any point here, a neighbouring point has a chance of being selected. If it is a better point, it is accepted, and if it is a worse point, it is accepted with a certain probability. In summary, every neighbour has a certain probability of being reached from a given point. **For the examples that you have studied, verify that the entire search space is connected, in the sense that each point (in particular, the global minimum, that we are looking for) can be reached from any starting point.**

Now, the progress of the algorithm can be modeled as a stochastic process which is like a Markov chain. The main characteristic being that the transition from one state to another is determined by probabilities independent of how the state was reached – this is not the case in Tabu search, which is why a different analysis would be needed there. This Markov chain will have asymptotic behaviour converging to stationary probabilities distributed (uniformly?) over the global minima of the function. This means that the probability of finding the state of the algorithm at these global minima is high as the algorithm progresses. Therefore the convergence is of a probabilistic type and cannot be guaranteed. This is why the SA algo is run with different starting points, sampled at random, to increase the chances of converging to the true, global minimum.

The travelling salesman problem (TSP)

This is a standard example of combinatorial optimization, which is 'difficult' to solve optimally, in reasonable time. Note that as with most discrete problems, it is possible to solve such problems optimally, in a finite manner through enumeration. But it is widely believed that there is no 'efficient' algorithm that will be able to solve large size versions of this problem. So heuristics and other randomized procedures are acceptable for such problems, and it provides a good benchmark to test various procedures.

The problem is to find a sequence of visiting n cities (and return to the starting city) with the objective of minimizing the total cost of travel. The input data is a cost matrix C, where the (i,j) entry is the cost of going from city i to city j.

Consider the following formulation. Define the variable x_{ij} = 1 if city i follows city j on the tour, and zero otherwise. Then solve the assignment problem
Min $\Sigma_i \Sigma_j c_{ij} x_{ij}$

s.t. $\Sigma_i x_{ij} = 1$ all j, and $\Sigma_j x_{ij} = 1$ for all i, with $x_{ij} \varepsilon \{0,1\}$.
Why is this NOT a valid formulation of the TSP?

Formulate a correct optimization problem (using appropriate notation) for the TSP and develop an SA algorithm and GA version to solve it.

The traveling salesman problem (TSP), apart from being of theoretical interest, is a cornerstone problem of combinatorial optimization. It is the prototypical NP-hard problem, which is not known to have an 'efficient' polynomial (i.e. effort increasing as a polynomial function of the problem size) algorithm for its exact solution. The decision versions of such problems are called NP-complete problems, indicating that if any such problem has a polynomial solution, then so do a whole bunch of similar problems (none of whom have a known polynomial algorithm for their solution). For details of this terminology and its extended implications, you would need to refer to a book on complexity theory or combinatorial optimization. Note that NP does not stand for non-polynomial, but for Non-deterministic Polynomial.

Some problems can be cast directly in TSP language.

- Routing problems on networks, where total routing costs are distance dependant. An example is the processing sequence in PCB manufacture or VLSI fabrication.
- Scheduling problems. An example is that of minimizing overall makespan for sequence dependant setup times on equipment such as paint mixing machines.
- Some analytical problems (see page 291 in Belegundu and Chandrupatla).

Many problems are very similar to the TSP, such as the Vehicle Routing Problem, which has obvious practical implications in distribution problems (milk runs, courier, post, etc.) and the Hamiltonian circuit problem in graph theory.

A large number of heuristics are available for TSP, ranging from simple ones to very sophisticated one. **Try to think of one or two on your own and read about some. They may work quite well on small problems.** A direct math programming formulation of the TSP in terms of a linear integer programming problem is possible, but not very useful. Such a formulation for some other problems (such as the VRP) is very cumbersome and may not be worth the effort.

The knapsack problem

This is another fundamental problem in combinatorial optimization. This is convenient to express as a linear integer programme as follows (for given constants v_i and c_i):
Max $\Sigma_i v_i x_i$
s.t. $\Sigma_i c_i x_i <= B$
$x_i \varepsilon \{0,1\}$.

Exercise: Show that the LP relaxation of this problem has an easy solution. One obvious way of trying to solve this problem is to round off the solution of the LP relaxation. Try to construct a problem instance where this will perform poorly.

Convince yourself that the value of the objective function in the LP relaxation will be an upper bound for the optimal solution for the knapsack problem.

Formulate an exact algorithm for solving the knapsack problem.

The term knapsack comes from the interpretation where the decision variable represents the choice of what items to pack in a knapsack (rucksack) so as to maximize value to the hiker (item i has value v_i) subject to a weight (or budget constraint). This too, occurs in a number of settings, including as sub-problems in many larger problems.

One of the exact solution methods is through branch and bound. This is explained in a number of books. Briefly speaking, the branch and bound method for integer constrained (usually with linear constraints) optimization problems is as follows:

At every stage, define a set of problems that capture the different possible values of all or some of the variables. For example, where all variables take on values 0 or 1, one set of problems could be ones where variable x1 takes on value 0 and another set where x1 takes on value 1. At every node defining such a subset of problems, it is often possible to get a bound on the optimum value for all problems in that set (e.g. by relaxing the constraints for all the variables that are not yet fixed, which then yields a simple LP). Comparing this bound with the best known (integer) solution at that point may allow us to discard that set of solutions without any further exploration. For open solutions, the bounds may provide a criterion for further branching and exploring promising nodes. For details, see a text book (e.g. Belegundu and Chandrupatla, chapter 8).

Try this procedure for the knapsack problem. This is closely connected, in this case, with dynamic programming algorithms for this problem.

Set covering problem

A problem similar to the knapsack problem, but of more general nature is the set covering problem. A simple version of this is the following: Given the covering matrix A of zeros and ones, where a_{ij} represents whether or not item i covers requirement j (say m rows and n columns), solve

$\text{Min } \Sigma_i x_i$
$\text{s.t. } \Sigma_i a_{ij} x_i >= 1, \text{ all j}$
$x_i \varepsilon \{0,1\}$

This minimizes the number of items required to 'cover' all demands. As with many combinatorial problems, the set covering problem admits a number of heuristics. One set of rules that can reduce the problem size substantially is the following:

Row reduction: If the row k in matrix A dominates the row l in matrix A in terms of the position of the ones (i.e. $A_{kj} = 1$ implies $A_{lj} = 1$, then one of the rows of A can be deleted from consideration. **Which one?**

Column reduction: If the column u in matrix A dominates the column v in matrix A in terms of the position of the ones (i.e. $A_{iu} = 1$ implies $A_{iv} = 1$, then one of the columns of A can be deleted from consideration. **Which one?**

Solitary 1: If there is a solitary 1 in column v, the constraints can be substantially simplified. **How?**

Exercise: Show that the set covering problem with the matrix A having three rows [1,1,0], [0,1,1] and [1,0,1] is not easily solvable by LP relaxation or by the heuristics above. Generalize this example to larger dimensional ones.

The weighted set covering problem is a natural generalization of the form
Min $\Sigma_i c_i x_i$
s.t. $\Sigma_i a_{ij} x_i >= 1$, all j
$x_i \varepsilon \{0,1\}$
for given weights c_i.

Genetic algorithms

Another method for finding globally optimal solutions is by the use of Genetic Algorithms. A self contained reference for that is Chapter 6 of K.Deb's book, Optimization for Engineering Design, which should be read by those interested (this chapter also includes material on Simulated Annealing). Belegundu and Chandrupatla also has a quick section on it and Chong and Zak have a more theoretically oriented chapter on this topic.

The first step is to decide on an encoding scheme to represent the variables to the desired accuracy using a (usually binary) string. The length of the string represents the desired accuracy. If there are bounds on the variables, a simple scheme is to use the string 00…0 for each variable at its lower bound and 11…1 at its upper bound and intermediate values suitably interpolated **[see K.Deb for details. Note that the formulation discussed there has some box constraints on the variables to make the domain a finite one for realistic encoding.]**.

An initial set of solutions is selected by a random mechanism which favours desirable solutions (as of now). The desirability of a solution is evaluated by a fitness function (which includes the objective function, and some penalty on violated constraints, if any). This sampling of the population is done by an appropriate weighted probability mechanism, referred to as the roulette wheel (creating a probability distribution where better solutions have a higher probability of being selected). **[Make sure you understand this simple mechanism and how to implement it.]**

From the initial pool at any stage, a new set of solutions is generated through various operators, of which the most commonly used are the crossover and mutation operators. These randomly perturb the solutions to get new ones (which explore new parts of the

search space). These can be defined in many ways, and can be used to control the progress of the algorithm. Termination criteria is in terms of the number of iterations and keeping track of the best available solutions at any time.

Crossover tries to combine two solutions to get a third, different one. The points in the sequence of bits where the crossover can take place can be controlled. Mutation takes one solution and generates a different one through random perturbation of that solution. Parameters of a typical GA implementation would include the probabilities associated with the crossover and mutation operations.

For combinatorial optimization problems, the coding is most often in terms of 0-1 variables, which map naturally to chromosomes. For the n-city TSP, for example, we could have a string of n^2 0-1 variables indicating the position of the i-th city in the tour or a string of n variables (each taking on integer values from 1, …, n) representing the order in which the cities are visited. Notice that for the TSP, in the second representation, a set of n circular permutations are actually the same, as far as a tour is concerned. A fitness function is evaluated for each chromosome.

The original GA is attributed to Holland and associates in the 1970's. In that, the initial population of M chromosomes was selected at random, and a crossover was effected by choosing one parent with a high fitness function and another parent chosen at random, to get a new pair of offspring.

Convergence analysis of the GA is not easy and relies on defining schema (which represent groups of solutions) and the average fitness of the population at every step and the specific reproduction plan that is chosen. For details, you would have to refer to more specialized references, beyond the scope of this course.

Try the function $f(x) = x^3 - 60x^2 + 900x + 100$ with x restricted to integer values between 0 and 31 to find the maximum, using a genetic algorithm. You can use the ordinary binary representation of integers as an illustrative coding for this one dimensional problem [e.g. the string 10011 would represent x = 19, and the string 00101 would represent x = 5]. Choose 5 strings at random and implement crossover and mutation and see how the algorithm progresses.

Try the Himmelblau function $f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$ with the box constraints $0 <= x_1, x_2 <= 6$, and also the constrained version with $(x_1 - 5)^2 + x_2^2 - 26 >= 0$, and code it yourself, to understand how the method works.

As a further exercise, also try the simulated annealing algorithm on the same function and explore the following issues:

What are the initial points for which the algorithm terminates at the global minimum?
What is the sensitivity of the algorithm to the cooling schedule and the initial parameter?

Come up with a scheme for GA for the slot timetabling problem and implement it.

Performance of Genetic Algorithms

It is difficult to come up with general results about the performance of GA as there are a number of different ways of implementing GA, starting with options in encoding the solution space and variables and the different ways of generating starting populations, the selection mechanisms and the methods used for 'evolution' of new generations of solutions. But some broad principles are as follows.

We restrict the analysis to simple, single-point crossover and mutation as the two mechanisms of evolution. For convenience, we use the binary encoding of solutions to illustrate the arguments. The framework used is that of a schema, which is a subset of chromosomes, where some of the values in the chromosome take on a fixed value. For example, the schema * * 1 * 0 * * represents a collection of chromosomes, which includes the chromosomes 1 0 1 0 0 0 1 and 1 1 1 1 0 1 0 and several others.

The order of a schema is the number of fixed entries and the length of a schema is the distance between the first and last fixed positions in the schema (the order of the schema above is 2 and the length is also 2). A third quantity of interest is the fitness ratio, which is the average fitness of a schema divided by the average fitness of the population (overall). The following three results can then be verified.

[Note that each generation can be thought of as a time step in the algorithm, so some references use time instead of generation. In what follows, P_c is the probability that a given chromosome is selected for crossover and P_m is the probability that it is selected for mutation, while defining the next generation.]

Result 1: In the selection plan, if a parent is selected in proportion to its fitness, then the expected number of instances of schema H in generation k+1 is f(H,k) N(H,k) where f is the fitness ration of H in generation k and N is the number of instances of H in generation k.

Result 2: If P(H,k) is the probability that a schema H is represented in the generation k, and if crossover is applied in generation k with a probability P_c, then
$P(H,k+1) >= [1 – P_c\lambda(H)] [1 – P(H,k)] / (n – 1)$, where n is the length of the chromosome and λ is the length of the schema.

Result 3: If P(H,k) is the probability that a schema H is represented in the generation k, and if mutation is applied in generation k with a probability P_m, then
$P(H,k+1) >= [1 – P_m\mu(H)]$, where n is the length of the chromosome and μ is the order of the schema.

Putting these together, we get a lower bound on the presence of schema H in generation (k+1) if crossover and mutation are applied.

This gives the Schema Theorem for the expected number of representatives of schema H in generation k+1 as:
$E(H,k+1) >= [1 - P_c\lambda(H)] / (n - 1) [1 - P(H,k) - P_m\mu(H)]\ f(H,k)\ N(H,k)$

This basically says that short, low order schemata combine with each other to form better and better solutions. This itself is not enough to ensure anything, but it is expected to work well in practice, which it does, empirically in some applications.

Dependence on encoding: As can be seen, the success depends on the way the solution space is encoded. As an illustration, supposing the good schema are such that the length is long (the number of entries between the first and last fixed elements of the schema), then crossover less likely to favour the continuation of this schema in subsequent generations. Therefore even the ordering of variables in a multi-variable problem matters significantly, as it effects the encoding scheme and therefore the schemata which capture "desirable" solutions. It is this that makes GA somewhat of an experimental science (but which is quite successful sometimes)!

Introduction to Lagrangean relaxation

Taking the weighted set covering problem as an example, we illustrate the principle of Lagrangean relaxation. Consider the following example, from Beasley (in Reeves' book).

Min $2x_1 + 3x_2 + 4x_3 + 5x_4$
s.t. $x_1 + x_3 >= 1$
$x_1 + x_4 >= 1$
$x_2 + x_3 + x_4 >= 1$
$x_i \varepsilon \{0,1\}$ all i.

Make sure you are able to relate this to the notation of the weighted set covering problem above.

The related problem
Min $2x_1 + 3x_2 + 4x_3 + 5x_4 + \lambda_1 (1 - x_1 - x_3) + \lambda_2 (1 - x_1 - x_4) + \lambda_3 (1 - x_2 - x_3 - x_4)$
s.t. $x_i \varepsilon \{0,1\}$ all i.
if solved, gives a lower bound on the original problem value for any positive values of the λ multipliers. **Verify this using arguments similar to weak duality in LP. Also numerically try this for a few positive values of the multipliers and compare with the true optimal value for the original problem (which you can easily get, at least by enumeration).**

This problem for specific values of the multipliers is easy to solve (compared to the original problem). **Try it.** A solution of such a problem which also satisfies the original feasibility and complementary slackness conditions would solve the original problem.

Lagrangean relaxation attempts to construct a sequence of such problems and iteratively come up with better bounds which can either be used in a branch and bound procedure or solve the problem optimally, directly. The procedure relies on a clever updating of the values of the multipliers to get the best value of the relaxed problem. Subgradient optimization is one technique used in this context.

ME 609 – Optimization methods in engineering Quiz I - 1 hour
No clarifications. Answer as many questions as you can (at least 4).

Partial/full credit only when logic is explained.

1. Find the maximum of the function $f(x) = 10 + x^3 - 2x - 5\exp(x)$ in the interval [-5,5] to within an accuracy of 0.3.

2. Consider the unconstrained optimization problem on R^n, $\text{Min}_x \|Ax - b\|^2$ where A is a given m x n matrix (m >= n) and b is a given m-vector (the norm is the 2 norm defined on m-vectors). Show that this problem has a unique (global) minimum.

3. For a continuously differentiable function $f : R^n \rightarrow R$, show that the descent directions at a point x form a convex set (i.e. if d and e are descent directions at x, then so is the vector $\lambda d + (1-\lambda)e$ for $\lambda \in [0,1]$).

4. Find all stationary points of the function $f(x) = x_1^2 - x_2^2$ and classify them as local minima, local maxima or neither.

5. What are the steepest descent and Newton directions for minimizing the function $x_1^2 + 10x_2^2$ starting from the point [1,1]? How would the two methods converge for this function (i.e. whether they will converge and at what rate)?

Quiz 1 : Questions and brief solutions

1. Find the maximum of the function $f(x) = 10 + x^3 - 2x - 5e^x$ in the interval [-5,5] to within an accuracy of 0.3.
Sol: One can solve this problem by various methods. Here the solution is given using Newton's method.
max f(x) is min of –f(x). Let $F(x) = -10 - x^3 + 2x + 5e^x$
$\therefore F'(x) = -3x^2 + 2 + 5e^x$ and $\therefore F''(x) = -6x + 5e^x$
From Newton's method, $x^{(k+1)} = x^{(k)} - \dfrac{F'(x^{(k)})}{F''(x^{(k)})}$.
Assume $x^{(0)} = 0$, $F(x^{(0)}) = -5$
$x^{(1)} = 0 - \dfrac{7}{5} = -1.4$, $F(x^{(1)}) = -8.8230$

$$x^{(2)} = -1.4 - \frac{-2.64}{9.633} = -1.126, \qquad F(x^{(2)}) = -9.2027$$

$$x^{(3)} = -1.126 - \frac{-0.182}{8.378} = -1.1043, \qquad F(x^{(3)}) = -9.2047$$

Within the desired accuracy, the function has a maximum at x = -1.1043 with function value 9.2047.

2. Consider the unconstrained optimization problem on R^n, $Min_x \|AX - b\|^2$ where A is a given m x n matrix (m>=n) and b is a given m-vector (the norm is the 2 norm defined on m-vectors). Show that this problem has a unique (global) minimum.

Sol: $Min_x \|AX - b\|^2$, i.e. $Min_x (AX - b)^T (AX - b)$.

$Min_x f(x) = X^T A^T AX - 2b^T AX + b^T b$.

This is a quadratic function.

$\nabla f(x) = 0$, gives $A^T AX = 2A^T b$, Which has a unique solution if $A^T A$ is non-singular

$\nabla^2 f(x) = \frac{1}{2} A^T A$. This is always positive definite for $A \neq 0$. Hence the problem has a unique (global) minimum.

3. For a continuously differentiable function f: $R^n \rightarrow R$, show that the descent directions at a point x from a convex set (i.e. if d and e are descent directions at x, then so is the vector $\lambda d + (1-\lambda)e$ for $\lambda \in [0,1]$).

Sol: If d is a descent direction at x, then $\nabla f(x)^T d < 0$

If e is a descent direction at x, then $\nabla f(x)^T e < 0$

Consider the direction $h = \lambda d + (1-\lambda)e$.

Then, $\nabla f(x)^T h = \nabla f(x)^T [\lambda d + (1-\lambda)e]$

$= \lambda \nabla f(x)^T d + (1-\lambda) \nabla f(x)^T e < 0$

$\therefore$ h is also a descent direction. i.e. the set of descent directions is a convex set.

4. Find all stationary points of the function $f(x) = x_1^2 - x_2^2$ and classify them as local minima, local maxima or neither.

Sol:

$$f(x) = x_1^2 - x_2^2$$

$$\nabla f(x) = \begin{bmatrix} 2x_1 \\ -2x_2 \end{bmatrix}$$

For a stationary point, $\nabla f(x) = 0$. So $[x_1, x_2] = [0,0]$ is the only stationary point.

For classifying a stationary point, the characteristic matrix should be checked for its definiteness at that point. The eigen-values of the second derivative matrix at [0,0] are 2

and –2, which are neither all positive or zero nor all negative or zero. Which means the matrix is neither positive semi definite nor negative semi definite. Therefore (0, 0) is neither a maximum nor a minimum of the given function.

5. What are the steepest descent and Newton directions for minimizing the function $x_1^2 + 10x_2^2$ starting from the point [1, 1]? How would the two methods converge for this function (i.e. whether they will converge and at what rate)?

Sol:

$$f(x) = x_1^2 + 10x_2^2$$

The steepest descent direction

$$d = -\nabla f(x)\big|_{(x_1,x_2)}$$

$$= -\begin{bmatrix} 2x_1 \\ 20x_2 \end{bmatrix}_{(1,1)}$$

$$= \begin{bmatrix} -2 \\ -20 \end{bmatrix}$$

Newton's direction

$$d = -\left[\nabla^2 f(x)\right]^{-1}_{(x_1,x_2)} \left[\nabla f(x)\right]_{(x_1,x_2)}$$

$$\left[\nabla^2 f(x)\right]^{-1}_{(x_1,x_2)} = \begin{bmatrix} 2 & 0 \\ 0 & 20 \end{bmatrix}^{-1}_{(1,1)}$$

$$= \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{20} \end{bmatrix}$$

$$\left[\nabla f(x)\right]_{(x_1,x_2)} = \begin{bmatrix} 2x_1 \\ 10x_2 \end{bmatrix}_{(1,1)} = \begin{bmatrix} 2 \\ 20 \end{bmatrix}$$

Therefore Newton's direction

$$d = -\begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{20} \end{bmatrix} \begin{bmatrix} 2 \\ 20 \end{bmatrix}$$

$$= \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

Convergence:

As the minimum of the function is at (0, 0), when started from (1, 1) Newton's method converges in one step while the steepest descent method takes infinite steps along a linear tube.

ME 609 : Optimization methods in Engineering : Quiz 2 : 23-2-2006 : 8-9 a.m.
Open handwritten notes only (no borrowing of notes/calculators etc.)

1) a) For the linear programme (P) Min $c^T x$ s.t. $Ax = b$, $x >= 0$ (with the usual notation), show that if P is unbounded, then the dual LP is infeasible.
b) Where in the simplex algorithm is infeasibility detected?
[For extra credit, you can relate the 'certificate' for infeasbility of the dual LP, to a 'direction' of the primal LP along which the objective function is unbounded].

2) a) Consider the symmetric rank 1 update $H_{k+1} = H_k + \alpha uu^T$, where H is the approximation to the inverse of the Hessian matrix at step k. What is the condition that α and u must meet so as to satisfy the Quasi Newton condition?
b) Why is the symmetric rank 1 update not adequate to design an effective approximate Quasi Newton algorithm?

3) For the Himmelblau function, $f(x_1,x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$, for the starting point [0,0], find the Newton direction, and verify that it is not a descent direction. Using the result that the eigenvalues of a real symmetric matrix $(A + \lambda I)$ are those of A with λ added to each of them, how will you find a good descent direction that is "close" to the Newton direction?

4) a) Show that the feasible region of an LP in standard form is a convex set.
b) Consider the extreme point-extreme direction representation of the feasible region of the standard form LP. Is the representation of any feasible point in terms of a convex combination of its extreme points and non-negative combination of its extreme directions a unique one?

Quiz 2 – brief solutions

a) For the linear programme (P) Min $c^T x$ s.t. $Ax = b$, $x >= 0$ (with the usual notation), show that if P is unbounded, then the dual LP is infeasible.

The dual is Max $w^T b$ s.t. $w^T a <= c^T$. Any feasible w for the dual, say w', gives a bound $w'^T b$ on the optimal value of the primal (P), by the weak duality inequality which says that $c^T x >= w^T Ax >= w^T b$ for any pair x (feasible for P) and w (feasible for D). Since P is unbounded, there can be no feasible w for D.

b) Where in the simplex algorithm is infeasibility detected?

In Phase 1 simplex (e.g. using artificial variables).

[For extra credit, you can relate the 'certificate' for infeasbility of the dual LP, to a 'direction' of the primal LP along which the objective function is unbounded].

2 a) Consider the symmetric rank 1 update $H_{k+1} = H_k + \alpha uu^T$, where H is the approximation to the inverse of the Hessian matrix at step k. What is the condition that α and u must meet so as to satisfy the Quasi Newton condition?

The QN condition is that H_{k+1}[difference in gradients] = [difference in x values]. i.e. $H_{k+1}s^k$
$= t^k$, where $s^k = \nabla f(x^{k+1}) - \nabla f(x^k)$ and $t^k = x^{k+1} - x^k$
Applying this to $H_{k+1} = H_k + \alpha uu^T$ gives $u = t^k - H_k s^k$ and $\alpha = 1/s^{kT}(t^k - H_k s^k)$

b) Why is the symmetric rank 1 update not adequate to design an effective approximate Quasi Newton algorithm?

It may not be possible to achieve both the QN condition and preserve the positive definiteness condition (for descent).

3) For the Himmelblau function, $f(x_1,x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$, for the starting point [0,0], find the Newton direction, and verify that it is not a descent direction.

Straightforward. Recall that descent means that the vector makes an obtuse angle with the gradient vector at that point.

Using the result that the eigenvalues of a real symmetric matrix $(A + \lambda I)$ are those of A with λ added to each of them, how will you find a good descent direction that is "close" to the Newton direction?

Some of the eigenvalues of $\nabla^2 f(x_k)$ are negative. Let the most negative one be λ_1, then add a multiple $(-\lambda_1 + \varepsilon)I$ to the Hessian matrix to get a 'sufficiently' positive definite matrix which is close to the Hessian matrix.

4 a) Show that the feasible region of an LP in standard form is a convex set.

Take y and z satisfying the constraints Ax = b, x >= 0 and show that a convex combination of y and z also satisfy the same constraints.

b) Consider the extreme point-extreme direction representation of the feasible region of the standard form LP. Is the representation of any feasible point in terms of a convex combination of its extreme points and non-negative combination of its extreme directions a unique one?

No.

Mid sem exam and solutions (some brief, some not so brief)

Q.1 The diet problem P is Min $c^T x$ s.t. $Ax >= b$, $x >= 0$, where x represents the vector of quantities of foods procured, c, the vector of unit costs, b the vector of nutrition requirements and $A = [a_{ij}]$ the matrix comprising the quantities of i-th nutrient in the j-th food. Assume that $c > 0$ and $b > 0$. Consider its dual D with variables w.

(a) What is the interpretation of the complementary slackness conditions $w^T(Ax - b) = 0$ and $x^T(c - A^T w) = 0$? **[5 marks]**

Sol The dual is max $b^T w$ s.t. $w^T A <= c^T$, $w >= 0$.

The complementary slackness conditions $w^T(Ax-b) = 0$ can be interpreted as follows. The (shadow) price is zero for a nutrient that is anyway met above requirement in the optimal mix and the requirement of a nutrient is met exactly in the cost optimal plan if the shadow price is positive.

The complementary slackness condition $x^T(c-A^T w) = 0$ can be interpreted as follows. If the i-th food is consumed at positive levels in the optimal mix, the fairness condition on the i-th food price is satisfied with equality. If the i-th fairness condition in the dual price problem is satisfied with inequality at optimality, then the i-th food is not used in the optimal food mix in the primal.

1 (b) Show that a pair of vectors (x,w) which satisfy (i) the feasibility conditions of P and D and (ii) the complementary slackness conditions above will actually be optimal for P and D. Suggest a feasible solution (x,w) to this system, but without the complementary slackness conditions (i.e. only (i) above). [If you like, you can do **(b)** for the standard form LP and its dual rather than the diet problem and its dual]. **[10 marks]**

Sol For any feasible x' for P and w' for D, we have $c^T x' >= w'^T A x' >= w'^T b$. If equality holds in these conditions, then x' and w' are optimal for the respective problems, since they have achieved their respective bounds (i.e. $c^T x >= (c^T x' = w'^T A x' = w'^T b) >= w^T b$ for any other feasible x and w.

Now if we have x' and w' satisfying the complementary slackness conditions as above, then we have exactly these conditions satisfied (i.e. $c^T x' = x'^T A w' = b^T w'$).

1(c) Suppose the diet problem has been solved to optimality. Now one more nutrient is considered (i.e. one more constraint is added). How would you solve this new problem (i.e. without doing it from scratch)? **[5 marks]**

Sol If the dual problem has been solved to optimality, and another constraint is added, the original solution is checked to see if it is still feasible. If so, the solution continues to remain optimal (and it is feasible). If not, apply one or more iterations of the dual simplex method, which retains optimality and tries to obtain feasibility (note that the original solution provides a starting solution for the dual simplex method).

Q2. (a) Show that a set K belonging to R^n is bounded in the 1-norm if and only if it is bounded under the infinity norm. [Bounded means that there is a number M so that $||x|| <= M$ for all x in the set.] **[10 marks]**

Sol K is bounded in the infinity norm. This means that there is an M such that $\max_i |x_i| <= M$. This means that $\Sigma_i |x_i| <= nM$, for all x in K. So there is a finite number ($a = nM$) so that $||x||_1$ is less than a for all x in K, so it is bounded in the one norm.

Conversely, if K is bounded in the 1-norm, then $||x||_\infty <= ||x||_1 <= M$ for some M, so it is bounded in the infinity norm.

2(b) For vectors x in R^n, consider the function $f(x) = \text{sqrt}(x^TQx)$ for a positive definite matrix Q. Does f define a norm on vectors? **[10 marks]**

Sol This question implicitly assumes that Q is a symmetric positive definite matrix. For matrices with real number entries, our definition of positive definiteness is perhaps applicable even for non-symmetric matrices, but in all our discussions (including the eigenvalue characterization of positive definiteness, we have taken Q to be real symmetric matrices).

For $f(x) = \text{sqrt}(x^TQx)$ to be a norm, it has to satisfy the conditions that $f(x) = 0$ for $x=0$ (true here), $f(x) > 0$ for x not equal to 0, (true), $f(ax) = |a|f(x)$ for scalar a and any x (true) and finally the triangle inequality $f(x+y) <= f(x) + f(y)$. This last inequality is also in fact, true. This can be shown in a number of ways. One way is this.

Consider the square of both LHS and RHS, ie $(x+y)^TQ(x+y)$ and $x^TQx + y^TQy + 2\text{sqrt}(x^TQx)(y^TQy)$. If we can show that $x^TQy <= \text{sqrt}(x^TQx)(y^TQy)$, then the inequality follows. This is actually the Cauchy Schwarz inequality (akin to the usual inner product inequality $|x^Ty| <= ||x||^2 ||y||^2$ which you would be familiar with in a different form – for example, the fact that $\text{Cos}\ \theta <= 1$). If you are not familiar with this, try the inequality $(x-\alpha y)^TQ(x-\alpha y) >= 0$ for all vectors x,y and scalars α. This is certainly true for $y = 0$ and for y not equal to zero, take $\alpha = (x^TQy)/(y^TQy)$ and the triangle inequality will follow. There are many other ways to show the required inequality. You will be given sufficient credit for going some way along this argument.

Q3. Consider the quadratic function $f(x) = ½\ x^TQx – b^Tx$ where Q is given as and b is $[-1\ 1]^T$. For a real symmetric matrix Q, a set of (non-zero vectors) $\{d_i\}$ is called Q-conjugate if $d_k^TQd_j = 0$, for k not equal to j. A set of conjugate directions for Q in f above is $d_1 = [1,0]^T$ and $d_2 = [-3/8,3/4]^T$.

3(a) For this example, show that starting from [0,0], minimizing along these (two) conjugate directions will give a stationary point of f. **[5 marks]**

Sol The (local/global) minimum of the function f is got by solving $\nabla f(x^*) = 0$, which gives $x^* = [-1, 3/2]$. As usual, for a quadratic function, the minimum itself is directly obtainable by analysis, but the method is developed and then subsequently applied to more general non-linear functions.

Starting with $x^0 = [0,0]$, minimizing along the two directions d^1 and d^2 gives the points $x^1 = [-1/4, 0]$ with $\alpha_1 = -1/4$, and $x^2 = [-1, 3/2]$ with $\alpha_1 = 2$.

3(b) Implement (at least one step of) any QN method that you know for minimizing this function starting from [0,0]. **[5 marks]**

3(c) Show that any set of Q-conjugate directions is linearly independent. **[5 marks]**

Sol Let $\{ d_1, \ldots, d_n\}$ be a set of Q-conjugate directions. If they are linearly dependent, then $\Sigma_i \lambda_i d_i = 0$ where at least one λ_k is not zero. Take that index k and multiply through by $Q d_k$. Since the directions are conjugate, the LHS will be $\lambda_k d_k Q d_k$, which cannot be zero. Contradiction.

3(d) How will you use the conjugate direction (or a Quasi Newton) method to solve a square system of equations Ax = b, where A is positive definite? **[5 marks]**

Sol Notice that solving Ax = b is equivalent to minimizing the quadratic function ½ $x^T Ax$ – $b^T x$. So apply the conjugate direction minimization technique for minimizing that function. This technique is actually relevant for solving general systems of equations Ax=b, not just for A positive definite.

Q4. Write down the LP for the shortest path problem. Interpret the dual LP to this, including the dual variables and constraints and the objective function. **[5 marks]**

Sol The LP for the shortest path problem from s to t on a directed network N (with node set N and arc set A) is - Min $\Sigma_i c_i x_i$ subject to $\Sigma_{i \text{ outgoing from s}} x_i = 1$, $\Sigma_{i \text{ incoming to t}} x_i = 1$, $\Sigma_{i \text{ outgoing from j}} x_i - \Sigma_{i \text{ incoming to j}} x_i = 0$ for j not equal to s or t, and $0 <= x_i <= 1$

[Note: i refers to arcs and j to nodes].

The dual of this is

Max $w_t - w_s$ subject to s.t. $w_{j2} - w_{j1} <= c_i$ where i goes from j1 to j2.

[Note: One constraint for each arc i]. In this, the values w_i can be thought of as node potentials and the objective function refers to the max potential difference between s and t, subject to the fact that the potential difference between any two nodes is no more than the cost (distance) between the two end points of the arcs. Note that the w_i's are unconstrained and it is only the differences in the w_i's that matter (both in the objective function and in the constraints). Therefore we can arbitrarily set one of them (say w_s) to zero.

Q5. An open box is made from a rectangular sheet of metal (170 cm x 90 cm) by cutting out h x h dimension squares at the corners (these squares are then discarded), bending the flaps and welding/joining the edges. Every cubic cm of volume of the open box would bring in a profit of Rs 5 and every cm of welding length costs Re 1 and every sq cm of lost area costs Rs 2.

(a) With this information, what is the value of h for maximum profit? You would need to justify the maximal nature of the proposed solution. **[10 marks]**

Sol The problem can be cast as an unconstrained optimization problem (a slightly different version of this problem is solved in Belegundu and Chandrupatla and this problem appears as an exercise). The solution is straightforward. Objective function in terms of h is 5(p-2h)(q-2h)h – 4h – 2($4h^2$), where the first term is the revenue from the volume of the box, the second term is the cost from joining and the third term the cost of material loss. This cubic expression in h has to be maximized. Set derivative equal to zero and take the positive root(s) of the resulting quadratic equation (since they are physically meaningful). Check if second derivative is negative at the proposed solution.

Q5(b) What are the assumptions you would need to make so that this problem is meaningful (in particular, what would you have to assume so that the data for this problem makes sense)? Where does the cost of the material of the box appear in this formulation? **[5 marks]**

Sol The main assumptions are that there is unlimited demand – independent of price (so whatever is made can be sold at a constant unit profit) and that the costs are linear in the amounts manufactured. The cost of the material shows up as the 'loss' of material (which is presumably the cost of the material per sq cm minus salvage value, if any) and

also in the profit per piece, which would be selling price minus the material price. As you would see, that is a lot of assumptions to make about a simple problem. It just means that a clean optimization formulation is often the first step in solving a 'real' problem.

Extension: A more general way of looking at **Q4b** is this. Given a positive definite matrix Q, define an inner product <x,y> on two vectors x and y as x^TQy (note that for Q=I, this is the usual 'dot' product). Over R^n, an inner product on two vectors satisfies the following: <x,y> = <y,x>, <x,x> = 0 for x=0, <x,x> > 0 for x not equal to zero. <x+y,z> = <x,z> + <y,z> and <αx,y> = α<x,y> for a scalar α. Then if we define ||x|| as sqrt<x,x>, it is true that (a) |<x,y>| <= ||x|| ||y|| (this is called the Cauchy Schwarz inequality) and (b) ||x|| is indeed a norm on vectors. The triangle inequality for this norm can be derived from the C.S. inequality.

Norms defined from inner products are very convenient and define what are called Hilbert spaces and allow a much larger optimization framework over vector spaces (e.g. optimization of functionals, with applications to control theory, design and a host of applications) – see Luenberger, Optimization over Vector Spaces, or other books on Functional Analysis for this. There are very nice geometric ideas based on projections and duality and so on, which would generalize and extend your intuition on R^n, and allow you to model and formulate least squares and other problems very nicely.

The main point of Q4b was however, this. Some of you may have wondered, if you have read any text-book, why Quasi Newton methods are also called Variable Metric methods. The reason for this is the following. Each QN approximation B for the second derivative matrix is a positive definite matrix. The norm $||x||_B = \text{sqrt}(x^TBx)$ then can be used to define a local direction finding step as follows. For the quadratic function $f(x) = ½\ x^TBx + d^Tx$, the steepest descent direction with respect to the B-norm is the solution to $\text{Min}_h\ ||h||_B + f'(x,h)$, where f' is the directional derivative of f at x in the direction h. Now verify that at any x, with B = I (a positive definite matrix for sure), this gives $h = -\nabla f(x)$, i.e. the usual steepest descent direction and for B = Hessian matrix (if positive definite), $h = -\nabla^2 f(x)^{-1}\nabla f(x)$, i.e. the Newton direction. For other positive definite matrices B, different directions can be derived as solutions to this minimization problem (using a different metric or norm each time). This solution of optimization problems using norms derived from a different positive definite matrix at each stage gives rise to the term variable metric method.

ME 609 : Quiz 3 : Closed book exam : 28 March 2006 : 8.00-8.55 a.m.

Q1 The solution to the Equality Constrained QP : Min $3 x_1^2 + 2 x_1 x_2 + x_1 x_3 + 2.5 x_2^2 + 2 x_2 x_3 + 2 x_3^2 – 8 x_1 – 3 x_2 – 3 x_3$ s.t. $x_1 + x_3 = 3$, $x_2 + x_3 = 0$ is $x^* = [2, -1, 1]$. Verify that this satisfies the KKT conditions. Is this solution optimal to the inequality constrained problem with $x_1 + x_3 <= 3$, $x_2 + x_3 = 0$ also? **[10 points]**

Brief solution:

Equality constrained QP:

KKT Conditions
$\nabla f(x^*) + \mu_1 \nabla h_1(x^*) + \mu_2 \nabla h_2(x^*) = 0$
$h_1(x^*) <= 0$; $\mu_1 h_1(x^*) = 0$, $h_2(x^*) <= 0$; $\mu_2 h_2(x^*) = 0$
For $x^*=[2,-1,1]$, $\nabla f(x^*) = [3,-2,1]$, $\nabla h_1(x^*)=[1,0,1]$, $\nabla h_2(x^*)=[0,1,1]$,
Sub. in KKT condition,
$3+\mu_1 =0$, $-2+ \mu_2 =0$, $1+\mu_1 +\mu_2=0$.
Also, for $x^*=[2,-1,1]$, $h_1(x^*) = 0$; $h_2(x^*)= 0$;
Hence, KKT conditions are satisfied

Inequality constrained QP:
KKT Conditions
$\nabla f(x^*) + \lambda_1 \nabla g_1(x^*) + \lambda_2 \nabla g_2(x^*) = 0$
$g_1(x^*) <= 0$; $\lambda_1 g_1(x^*) = 0$; $\lambda_1 >= 0$ and $g_2(x^*) <= 0$; $\lambda_2 g_2(x^*) = 0$; $\lambda_2 >= 0$
It is clear $\lambda_1 = -3 <0$,
Hence KKT Condition is not satisfied and solution is not optimal.
The optimal solution for this is actually $x^*=[1.3793, -0.27586, 0.27586]$.

Q2 There are m jobs which are to be done by n people (m > n). Each person can do a maximum of k jobs. There is a cost c_{ij} if person i is allotted job j. Formulate an LP to solve the problem of deciding how the jobs are to be done (i.e. which person is to do which jobs). For each person i, there are some jobs (contained in set S_i) which person i is **not** able to do. Remember that the LP will have continuous decision variables and the required decision is a discrete one (does person i do job j or not?), so you have to justify your approach. **[10 points]**

Brief solution:
One way is the following. Define $c_{ij} = M$ (a large number) if j belongs to the forbidden set for person i. Then set up the network flow LP:
Min $\Sigma_i \Sigma_j c_{ij} x_{ij}$
s.t. $\Sigma_i x_{ij} = 1$, all j
$\Sigma_j x_{ij} <= k$, all i
$0 <= x_{ij} <= 1$

Here, x_{ij} takes on value 1 if person I does job j and 0 otherwise. By the integrality property of network flow LPs, the basic feasible solutions of this LP take on 0, 1 values and provide the required assignment.

Q3 Give an example of a constrained optimization problem involving differentiable objective function and constraint functions where the solution does **not** satisfy the KKT conditions. **[10 points]**

Examples already provided in notes

Quiz 4 : 25 April, 2006

You can use the following random numbers in the interval [0,1] if you like.

0.3146, 0.3673, 0.9008, 0.7848, 0.1271, 0.5554, 0.4539, 0.1610, 0.3102, 0.8798, 0.7392

Q1 For the 7-course 3-slot timetabling problem, a network representation may be convenient. Each node is a course and the arc between nodes represents the fact that there is common registration (k_{ij} – the label on arc i-j is the number of students registered in common for courses i and j).

Consider two different solutions a_1 = [1 2 3 1 2 3 1] and a_2 = [2 1 3 2 2 3 3] where the i-th represents the slot assigned to course i. Illustrate single point crossover and mutation operators on these two solutions and conclude whether it has led to any improvement. Note that selection has already been done, and you have to just implement the evolution.

Solution sketch: A crossover point has to be selected at random, to form two new solutions. A mutation position has to be selected at random and the new slot at that position also has to be selected at random. The fitness is the number of students who have clashing slots as a result of the slot assignment.

Q2 Explain how you will implement the simulated annealing algorithm on the TSP. Illustrate this with **two** iterations on the problem with data for the following eight node problem (where entries in the symmetric C matrix below represent costs from i to j.

Start with the solution a = [1 2 3 4 5 6 7 8] where a_i represents the i-th city visited in the tour.

Solution sketch: For implementing simulated annealing, the first thing to do is to describe a neighbourhood structure. For the TSP, a possible one is the following. Tours a and b are neighbours if one is obtainable from the other by one swap of positions on the tour. With this definition, a neighbour has to be picked at random (by selecting two positions on the tour for exchange – e.g. if positions 3 and 5 are chosen, the new solution obtained is [1 2 5 4 3 6 7 8]. If this results in lower cost, it is taken, otherwise, it is accepted with a probability determined by the temperature (an initial temperature has to be selected).

ME 609 Optimization methods in engineering – End semester exam –
29-4-06 – 2-5 p.m. – Closed book exam –
No clarifications (make appropriate assumptions)

Q1) Give an example of a discrete decision problem that can be solved efficiently through linear programming. You should quote a specific type of problem that has decision variables that take on either 0,1 or some other finite set of values and justify how LP can be used for solving it. **[3 marks]**

Soln: Examples of this type of problem are the assignment problem or the shortest path problem. A general category of such problems is that of Network flow LPs with linear costs on directed arcs in a network, with capacity constraints on arcs and flow balance constraints on nodes. The constraint matrix in such cases is a matrix of 0's and 1's that is totally unimodular, which has the property that every submatrix has determinant 0, 1 or –

1 and which therefore yields integer solutions if the data is integer. In particular, if the capacity bounds on arcs are 0 and 1, then we get binary values for the flow variables, which can be interpreted as the solution to the discrete problem (e.g. assignment or shortest path).

Note that the knapsack problem and the set covering problem are not in this category, although a single LP relaxation can often give a good solution to the problem and will almost always enable one to get a good approximate solution. Problems like TSP are hard to solve approximately using LP, specifically by using just one LP of 'reasonable' size.

Q2) Someone suggests that Newton's method for nonlinear optimization can be used for solving linear programmes efficiently. If you agree with this statement, how would you go about doing so? If you do not agree, say why not. **[3 marks]**

Soln: This is true. Some variants of interior point methods can be interpreted as Newton's method applied to the system of equalities for the primal dual system, with the non-negativity conditions added on in the algorithm. Barrier methods (a version of interior point methods that penalize violation of constraints) can also be interpreted in the language of Newton like methods.

Q3) Why is the Quasi Newton method also called a Variable Metric method? What are the numerical reasons for using a Quasi Newton method in preference to Newton's method? **[3 marks]**

Soln: The QN method maintains a positive definite matrix Q at each stage which estimates the Hessian matrix of the function at each iteration (and comes close to approximating it at the minimum). The descent direction at every stage is the solution to the optimization problem that minimizes the linear model of the function (the directional derivative) together with the norm of the direction vector with respect to the quadratic norm defined by this estimate (ie. Sqrt($1/2\ x^T Qx$). Since this norm (metric) varies from iteration to iteration, the method is called a variable metric method.

The QN method seeks to achieve three numerical goals vis a vis an implementation of the Newton method (which has desirable theoretical properties). One is to avoid computing the n^2 entries of the Hessian matrix at each step, by using an updating formula that is less intensive. Secondly, by using a low rank approximation for the update, we get an easy update of the inverse at every stage, rather than having to explicitly compute it. The third benefit is to preserve a positive definite approximation to the Hessian matrix, which therefore guarantees a descent direction at each stage.

Q4) Commercial/scientific software for solving linear programming problems is now freely available. What are those algorithms based on? **[2 marks]**

Soln: Revised simplex implementations or interior point methods.

Q5) For the n-course, m-slot timetabling problem, if the vector a = [a_1, …, a_n] represents a possible solution (a_i representing the slot assigned to the i-th course), define a neighbourhood structure on the solution space, and for that, compute the number of neighbours for a given solution. **[2 marks]**

Soln: A neighbour of a slot assignment can be defined as an assignment that differs in the slot assigned to any one course. With n courses and m slots, each assignment will have n(m-1) neighbours.

If the slot timetabling problem has a further constraint, that no more than k courses can be scheduled in any slot, show how you would account for this in a simulated annealing based algorithm and also in a genetic algorithm for this problem. **[2 marks]**

Soln: One way is to check a solution for feasibility, and discard it if not feasible and sample a neighbour again. Another way is to look only for feasible solutions while sampling for neighbours. The second option is difficult to implement in GAs.

Q6) Consider a twice differentiable function f(x). Explain why the Newton method may NOT converge to a local minimum for some starting iterates x^0. Illustrate this with an example (even a graph of f(x), a function of one variable is enough). **[4 marks]**

Soln: The Newton method may not converge to a local minimum if we start far away from it. Basically, if the starting point is far enough that the Taylor approximation is not numerically valid to justify the quadratic approximation, we may not even get descent. It is also possible to terminate at a stationary point that is not a minimum, if we start at such points.

Suggest a modification of this method so that it will work (either show some method for getting a suitable starting iterate x^0 or provide a modification that will work for any x^0). **[2 marks]**

Soln: The simplest modification is to modify the Hessian matrix till we get a positive definite matrix, which will guarantee descent. The continuity of second derivatives will ensure that close to a minimum, the second derivative matrix will anyway be positive definite and so this modification will not be required as we get close to the solution.

Q7) Write down the KKT conditions for the problem

Max ½ $x^T Q x$ s.t. $||x||_2^2 <= 1$.
Apply this to solve the problem where Q is the matrix

$$\begin{matrix} 2 & -1 \\ -1 & 1 \end{matrix}$$

Is this solution a local minimum or a global minimum? **[5 marks]**

Soln: Q is a negative definite matrix, so ½ $x^T Q x$ <= 0 for all x.

The KKT conditions are $Qx + 2\lambda x = 0$, $\lambda >= 0$, $x^T x - 1 <= 0$, $\lambda (x^T x - 1) = 0$. The case $x^T x - 1 < 0$ gives $\lambda = 0$ and so x has to be zero (since Q is nonsingular). This is actually a global maximum of f.

The case $x^T x = 1$ gives $Qx = -2\lambda x$ which can be solved as an eigenvalue eigenvector equation. It gives positive values of λ and corresponding values of x (normalized using $x^T x = 1$), which therefore satisfy the KKT conditions. The second derivative matrix Q is negative definite, so these would be local minima.

Q8) Consider the following two systems for a given matrix A and given vector b;
System 1 : $\{x : Ax = b, x >= 0\}$, System 2 : $\{y : y^T A <= 0, y^T b > 0\}$
Show that exactly one of them has a solution. **[3 marks]**

Soln: Suppose there is an x that is feasible for system 1. Then take any y that satisfies $y^T A <= 0$. For this y, we have $y^T b = y^T Ax <= 0$ because $y^T A <= 0$ and $x >= 0$. So it cannot satisfy $y^T b > 0$.

It is also easy to show that if there is a y satisfying system 2, then we have $y^T A <= 0$. For any $x >= 0$, we then have $y^T Ax <= 0$. Such an x cannot satisfy $Ax = b$, because then it would mean that $y^T b <= 0$, which is not true.

It can also be shown directly that if system 1 does NOT have a solution, then one is guaranteed for system 2.

Where is this result used in constrained optimization? **[2 marks]**

Soln: This is used to derive the KKT conditions, where the columns of A would be the constraint gradients and the vector b the objective function gradient.

Q9) Show that for a convex function f defined over a convex set K, a local minimum x* is also a global minimum over K. **[4 marks]**

Soln: Since x* is a local minimum of f, there is a neighbourhood of x*, say N(x*) so that $f(x^*) <= f(x)$ for any $x \varepsilon N(x^*)$. Suppose there is a point y in K with $f(y) < f(x^*)$. Consider points of the form $\lambda x^* + (1-\lambda)y$. For small enough l, this will lie in N(x*). From convexity, we have $f(\lambda x^* + (1-\lambda)y) <= \lambda f(x^*) + (1-\lambda)f(y) < \lambda f(x^*) + (1-\lambda)f(x^*) = f(x^*)$, which contradicts the fact that x* is a local minimum.

Q10) Consider a pivoting algorithm for quadratic programming, such as the active set method (a pivoting algorithm goes from one solution of a system of linear equations to another). When applied to the problem Min $-x_1^2 - x_2^2$
s.t. $x_1 + x_2 <= 2$, $x_1 - x_2 <= 2$, $x_2 <= 1$, $x_2 >= -1$, $x_2 - x_1 <= 10$, $x_1 + x_2 >= -10$,
show that this algorithm can terminate at a local minimum that is not necessarily a global minimum. If not for this specific example, illustrate graphically for any example how this can happen. **[5 marks]**

Soln: The point [1,0] is a local minimum that is not a global minimum.

Q11) Let f* be the optimum value of the constrained problem Min f(x) s.t. $g_1(x) <= 0$, $g_2(x) <= 0$, …, $g_m(x) <= 0$ for some real valued functions f, g_1, …, g_m. Show that the value of the problem with one constraint dropped is a lower bound on f*. **[2 marks]**

Soln: Let g* be the optimum value of the new problem (with one constraint dropped). Any x that is feasible for the original problem is also feasible for the new problem. Since g* is the best value for the new problem, we have g* <= f(x). In particular, the solution x that gives f* also satisfies this, so g* <= f*.

Show also that the solution to the problem Min $f(x) + \lambda g_1(x)$ s.t. $g_2(x) <= 0$, …, $g_m(x) <= 0$ provides a lower bound on f*, for any fixed $\lambda >= 0$. **[2 marks]**

Soln: Consider the problem Min $f(x) + \lambda g_1(x)$ s.t. $g_1(x) <= 0$, $g_2(x) <= 0$, …, $g_m(x) <= 0$. Let h* be the optimum value for this problem. Then since $\lambda >= 0$ and $g_1(x) <= 0$, we have h* <= f*. Since the problem Min $f(x) + \lambda g_1(x)$ s.t. $g_2(x) <= 0$, …, $g_m(x) <= 0$ is a relaxed version of this, we have a lower bound on h* and therefore on f*.

Q12) For an optimization problem over the constraint set
$K = \{(x_1,x_2)$ s.t. $(x_1^2 + x_2^2) = 1$, $(x_1 + 1)^2 + x_2^2 = 4\}$,
show that the KKT conditions may **NOT** hold at the point $(x_1, x_2) = [1,0]$. **[3 marks]**

Soln: The only point satisfying both the constraints is [1,0]. At this point, the constraint gradients are both of the form [a, 0]. So any objective function gradient which has a non zero value of the second component at [1 0] cannot satisfy the KKT conditions.

Q13) For a given m by n matrix A and RHS vector b, the problem of finding x to satisfy Ax = b is a common one in statistics and curve fitting. Assume A is full rank. Consider the cases m < n and m > n and say how you would go about getting a good solution to this problem in each case. **[4 marks]**

Soln: For the case m > n, the relevant problem could be Min $||Ax - b||_2$ or Min $||Ax - b||_2^2$. This has standard solutions of the form $x^* = (A^TA)^{-1}A^Tb$, where the inverse exists because A is full rank.

For the case m < n, there could be many solutions satisfying Ax = b. One option is to look for a solution that minimizes $||x||_2$ or minimizing $||x||_2^2$subject to Ax = b. This also has a solution of the form $x^* = A^T(AA^T)^{-1}b$.