

These notes are based on ???. No originality is claimed.

Contents

1	The General Optimization Problem	4
2	Basic MATLAB	4
2.1	Starting and quitting MATLAB	5
2.2	Matrices	5
2.3	Graphics	7
2.4	Scripts and functions	7
2.5	Files	8
2.6	More about functions	9
2.7	Homework	9
3	Basic properties of solutions and algorithms	9
3.1	Necessary conditions for a local optimum	9
3.2	Convex (and concave) functions	11
3.3	Global convergence of decent algorithms	11
3.4	Homework	12
4	Basic descent methods	13
4.1	Fibonacci and Golden Section Search	13
4.2	Newton's method	13
4.3	Applying line-search methods	14
4.4	Homework	15
4.5	Quadratic interpolation	15
4.6	Cubic fit	16
4.7	Homework	16
5	The method of steepest decent	17
5.1	The quadratic case	17
5.2	Applying the method in Matlab	18
5.3	Homework	20
6	Newton and quasi-Newton methods	20
6.1	Newton's method	20
6.2	Extensions	20
6.3	The Davidon-Fletcher-Powell (DFP) method	21
6.4	The Broyden-Fletcher-Goldfarb-Shanno (BFGS) method	22
6.5	Homework	22

7	Constrained Minimization Conditions	22
7.1	Necessary conditions (equality constraints)	23
7.2	Necessary conditions (inequality constraints)	24
7.3	Sufficient conditions	25
7.4	Sensitivity	26
7.5	Homework	26
8	Lagrange methods	27
8.1	Quadratic programming	27
8.1.1	Equality constraints	28
8.1.2	Inequality constraints	28
8.2	Homework	29
9	Sequential Quadratic Programming	29
9.1	Newton's Method	29
9.2	Structured Methods	29
9.3	Merit function	30
9.4	Enlargement of the feasible region	30
9.5	The Han–Powell method	31
9.6	Constrained minimization in Matlab	31
9.7	Homework	33
10	Penalty and Barrier Methods	33
10.1	Penalty method	33
10.2	Barrier method	34
10.3	A final project	35
10.4	An exercise in Matlab	35
11	Stochastic Approximation	37

1 The General Optimization Problem

The general optimization problem has the form:

$$\min_{\mathbf{x} \in \mathcal{R}^n} f(\mathbf{x})$$

subject to:

$$\begin{aligned} g_i(\mathbf{x}) &= 0 & i = 1, \dots, m_e \\ g_i(\mathbf{x}) &\leq 0 & i = m_e + 1, \dots, m \\ \mathbf{x}_l &\leq \mathbf{x} \leq \mathbf{x}_u \end{aligned}$$

In particular, if $m = 0$, the problem is called an unconstrained optimization problem. In this course we intend to introduce and investigate algorithms for solving this problem. We will concentrate, in general, in algorithms which are used by the *Optimization* toolbox of MATLAB.

We intend to cover the following chapters:

1. Basic MATLAB.
2. Basic properties of solutions and algorithms.
3. Basic descent methods.
4. Quasi-Newton methods.
5. Least squares optimization.
6. Sequential Quadratic Programming.
7. The REDUCE algorithm.
8. Stochastic approximation.


2 Basic MATLAB

The name MATLAB stands for *matrix laboratory*. It is an interactive system for technical computing whose basic data element is an array that does not require dimensioning.

2.1 Starting and quitting MATLAB

Starting: double-click on the **MATLAB** icon.

Quitting: File/Exit MATLAB or write `quit` in the command line.

Help: Help/Help Window, click on the  icon or type `help`.

2.2 Matrices

MATLAB is case sensitive. Memory is allocated automatically.

```
>> A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
A =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
>> sum(A)
ns =
    34    34    34    34
>> sum(A')
ns =
    34    34    34    34
>> sum(diag(A))
ns =
    34
>> A(1,4) + A(2,3) + A(3,2) + A(4,1);
>> sum(A(1:4,4));
>> sum(A(:,end));
>> A(~isprime(A)) = 0
A =
     0     3     2    13
     5     0    11     0
     0     0     7     0
     0     0     0     0
>> sum(1:16)/4;
>> pi:-pi/4:0
ns =
    3.1416    2.3562    1.5708    0.7854    0
>> B = [fix(10*rand(1,5));randn(1,5)]
```

```

B =
    4.0000    9.0000    9.0000    4.0000    8.0000
    0.1139    1.0668    0.0593   -0.0956   -0.8323
>> B(2:2:10)=[]
B =
     0     3     8     0     1
>> s = 1 -1/2 + 1/3 - 1/4 + 1/5 - 1/6 + 1/7 ...
-1/8 + 1/9 -1/10
s =
    0.6456
>> A'*A
ns =
    378    212    206    360
    212    370    368    206
    206    368    370    212
    360    206    212    378
>> det(A)
ns =
     0
>> eig(A)
ns =
    34.0000
     8.0000
     0.0000
    -8.0000
>> (A/34)^5
ns =
    0.2507    0.2495    0.2494    0.2504
    0.2497    0.2501    0.2502    0.2500
    0.2500    0.2498    0.2499    0.2503
    0.2496    0.2506    0.2505    0.2493
>> A' .* A
ns =
    256    15    18    52
    15   100    66   120
    18    66    49   168
    52   120   168     1
>> n = (0:3)';
>> pows = [n n.^2 2.^n]
pows =

```

0	0	1
1	1	2
2	4	4
3	9	8

2.3 Graphics

```
>> t = 0:pi/100:2*pi;
>> y = sin(t);
>> plot(t,y)
>> y2 = sin(t-0.25);
>> y3 = sin(t-0.5);
>> plot(t,y,t,y2,t,y3)
>> [x,y,z]=peaks;
>> contour(x,y,z,20,'k')
>> hold on
>> pcolor(x,y,z)
>> shading interp
>> hold off
>> [x,y] = meshgrid(-8:.5:8);
>> R = sqrt(x.^2 + y.^2) + eps;
>> Z = sin(R)./R;
>> mesh(x,y,Z)
```

2.4 Scripts and functions

M-files are text files containing MATLAB code. A text editor can be accessed via File/New/M File. M-files end with .m prefix. Functions are M-files that can accept input argument and return output arguments. Variables, in general, are local. MATLAB provides many functions. You can find a function name with the function `lookfor`.

```
>> lookfor inverse
INVHILB Inverse Hilbert matrix.
...
INV    Matrix inverse.
...
INVHESS Inverse of an upper Hessenberg matrix.
```

Or you can write a function in an M-file:

```
function h = falling(t)
global GRAVITY
h = 1/2*GRAVITY*t.^2;
```

save it and run it from MATLAB:

```
>> global GRAVITY
>> GRAVITY = 32;
>> y = falling((0:.1:5)');
>> falling(0:5)
ans =
     0     16     64    144    256    400
```

2.5 Files

The MATLAB environment includes a set of variables built up during the session — the Workplace — and disk files containing programs and data that persist between sessions. Variables can be saved in MAT-files.

```
>> save B A
>> A = 0
A =
     0
>> load B
>> A
A =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
>> save mydata A -ascii
>> dir
.          libut.dll          msvcirt.dll
..         license.dat       msvcopts.bat
b.mat     lmgr325a.dll         msvcrt.dll
bccengmatopts.bat  matfopts.bat       mwoles05.dll
bccopts.bat  matlab.exe         mydata
cmex.bat    medit.exe          patchmat.exe
....
```


2.6 More about functions

To obtain the most speed it's important to vectorize the computations.

Write the M-file *logtab1.m*:

```
function t = logtab1(n) x=0.01; for k=1:n
    y(k) = log10(x);
    x = x+0.01;
end t=y;
```

and the M-file *logtab2.m*:

```
function t = logtab2(n) x = 0.01:0.01:(n*0.01); t = log10(x);
```

Then run in Matlab:

```
>> logtab1(1000);
>> logtab2(1000);
```

2.7 Homework

1. Let $f(x) = ax^2 - 2bx + c$. Under which conditions does f has a minimum? What is the minimizing x ?
2. Let $f(\mathbf{x}) = \mathbf{x}'\mathbf{A}\mathbf{x} - 2\mathbf{b}'\mathbf{x} + \mathbf{c}$, with \mathbf{A} an $n \times n$ matrix, \mathbf{b} and \mathbf{c} n -vectors. Under which conditions does f has a minimum? a unique minimum? What is the minimizing \mathbf{x} ?
3. Write a MATLAB function that finds the location and value of the minimum of a quadratic function.
4. Plot, using MATLAB, a contour plot of the function f with $\mathbf{A} = \begin{bmatrix} 1 & 3 \\ -1 & 2 \end{bmatrix}$, $\mathbf{b} = [5 \ 2]'$ and $\mathbf{c} = [1 \ 3]'$. Mark, on the plot, the location of the minimum.

3 Basic properties of solutions and algorithms

3.1 Necessary conditions for a local optimum

Assume that the function f is defined over $\Omega \subset \mathcal{R}$.

Definition: A point $\mathbf{x}^* \in \Omega$ is said to be a *relative minimum point* or a *local minimum point* of f if there is an $\epsilon > 0$ such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all \mathbf{x} such that $\|\mathbf{x} - \mathbf{x}^*\| < \epsilon$. If the inequality is strict for all $\mathbf{x} \neq \mathbf{x}^*$ then \mathbf{x}^* is said to be a *strict relative minimum point*.

Definition: A point $\mathbf{x}^* \in \Omega$ is said to be a *global minimum point* of f if $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \in \Omega$. If the inequality is strict for all $\mathbf{x} \neq \mathbf{x}^*$ then \mathbf{x}^* is said to be a *strict global minimum point*.

In practice, the algorithms we will consider in the better part of this course converge to a local minimum. We may indicate in the future how the global minimum can be attained.

Definition: Given $\mathbf{x} \in \Omega$, a vector \mathbf{d} is a *feasible direction* at \mathbf{x} if there exists an $\bar{\alpha} > 0$ such that $\mathbf{x} + \alpha \mathbf{d} \in \Omega$ for all $0 \leq \alpha \leq \bar{\alpha}$.

Theorem 1 (First-order necessary conditions.) Let $f \in C^1$. If \mathbf{x}^* is a relative minimum, then for any \mathbf{d} which is feasible at \mathbf{x}^* , we have $\dot{f}(\mathbf{x}^*)' \mathbf{d} \geq 0$.

Corollary 1 If \mathbf{x}^* is a relative minimum and if $\mathbf{x}^* \in \Omega^0$ then $\dot{f}(\mathbf{x}^*) = \mathbf{0}$.

Theorem 2 (Second-order necessary conditions.) Let $f \in C^2$. Let \mathbf{x}^* be a relative minimum. For any \mathbf{d} which is feasible at \mathbf{x}^* , if $\dot{f}(\mathbf{x}^*)' \mathbf{d} = 0$ then $\mathbf{d}' \ddot{f}(\mathbf{x}^*) \mathbf{d} \geq 0$.

Corollary 2 If \mathbf{x}^* is a relative minimum and if $\mathbf{x}^* \in \Omega^0$ then $\dot{f}(\mathbf{x}^*)' \mathbf{d} = 0$ and $\mathbf{d}' \ddot{f}(\mathbf{x}^*) \mathbf{d} \geq 0$ for all \mathbf{d} .

Theorem 3 (Second-order sufficient conditions.) Let $f \in C^2$. Assume $\mathbf{x}^* \in \Omega^0$. If $\dot{f}(\mathbf{x}^*) = \mathbf{0}$ and $\ddot{f}(\mathbf{x}^*)$ is positive definite then \mathbf{x}^* is a strict relative minimum.

3.2 Convex (and concave) functions

Definition: A function f on a convex set Ω is said to be *convex* if

$$f(\alpha \mathbf{x}_1 + (1 - \alpha) \mathbf{x}_2) \leq \alpha f(\mathbf{x}_1) + (1 - \alpha) f(\mathbf{x}_2),$$

for all $\mathbf{x}_1, \mathbf{x}_2 \in \Omega$, $0 \leq \alpha \leq 1$.

Theorem 4 *If f_1 and f_2 are convex and $a > 0$ then $f_1 + f_2$ and af_1 are also convex.*

Theorem 5 *Let f be convex. Then $\Gamma_c = \{\mathbf{x} \in \Omega : f(\mathbf{x}) \leq c\}$ is a convex set.*

Corollary 3 *If f_1, \dots, f_m are convex then the set of points that simultaneously satisfy*

$$f_1(\mathbf{x}) \leq c_1, \dots, f_n(\mathbf{x}) \leq c_n$$

is convex.

3.3 Global convergence of decent algorithms

The algorithms we consider are iterative descent algorithms.

Definition: An algorithm A is a mapping that assigns, to each point, a subset of the space.

Iterative algorithm: The specific sequence is constructed by choosing a point in the subset and iterating the process. The algorithm generates a series of points. Each point being calculated on the basis of the points preceding it.

Descent algorithm: As each new point is generated, the corresponding value of some function decreases in value. Hence, there is a continuous function Z such that if A is the algorithm and Γ is the solution set then

1. If $\mathbf{x} \notin \Gamma$ and $\mathbf{y} \in A(\mathbf{x})$, then $Z(\mathbf{y}) < Z(\mathbf{x})$.
2. If $\mathbf{x} \in \Gamma$ and $\mathbf{y} \in A(\mathbf{x})$, then $Z(\mathbf{y}) \leq Z(\mathbf{x})$.

Definition: An algorithm is said to be *globally convergent* if, for any starting point, it generates a sequence that converges to a solution.

Definition: A point-to-set map A is said to be closed at \mathbf{x} if

1. $\mathbf{x}_k \rightarrow \mathbf{x}$ and
2. $\mathbf{y}_k \rightarrow \mathbf{y}, \mathbf{y}_k \in A(\mathbf{x}_k)$, imply
3. $\mathbf{y} \in A(\mathbf{x})$.

The map A is closed if it is closed at each point of the space.

Theorem 6 *If A is a decent iterative algorithm which is closed outside of the solution set Γ and if the sequence of points is contained in a compact set then any converging subsequence converges to a solution.*

3.4 Homework

1. To approximate the function g over the interval $[0, 1]$ by a polynomial p of degree n (or less), we use the criterion

$$f(\mathbf{a}) = \int_0^1 [g(x) - p(x)]^2 dx,$$

where $\mathbf{a} \in \mathcal{R}^{n+1}$ are the coefficients of p . Find the equations satisfied by the optimal solution.

- 2(a) Using first-order necessary conditions, find the minimum of the function

$$f(x, y, z) = 2x^2 + xy + y^2 + yz + z^2 - 6x - 7y - 8z + 9.$$

- (b) Verify the point is a relative minimum by checking the second-order conditions.

3. Let $\{f_i : i \in I\}$ be a collection of convex functions defined over a convex set Ω . Show that the function $g = \sup_{i \in I} f$ is convex on the region where it is finite.

4. Define the point-to-set mapping on \mathcal{R}^n by

$$A(\mathbf{x}) = \{\mathbf{y} : \mathbf{y}'\mathbf{x} \leq b\},$$

where b is a fixed constant. Is A closed?

4 Basic descent methods

We consider now algorithms for locating a local minimum in the optimization problem with no constraints. All methods have in common the basic structure: in each iteration a direction \mathbf{d}_n is chosen from the current location \mathbf{x}_n . The next location, \mathbf{x}_{n+1} , is the minimum of the function along the line that passes through \mathbf{x}_n in the direction \mathbf{d}_n . Before discussing the different approaches for choosing directions, we will deal with the problem of finding the minimum of a function of one variable — “line search”.

4.1 Fibonacci and Golden Section Search

These approaches assume only that the function is unimodal. Hence, if the interval is divided by the points $x_0 < x_1 < \dots < x_N < x_{N+1}$ and we find that, among these points, x_k minimizes the function then the over-all minimum is in the interval (x_{k-1}, x_{k+1}) .

The Fibonacci sequence ($F_n = F_{n-1} + F_{n-2}$, $F_0 = F_1 = 1$) is the basis for choosing altogether N points sequentially such that the $x_{k+1} - x_{k-1}$ is minimized. The length of the final interval is $(x_{N+1} - x_0)/F_N$.

The solution of the Fibonacci equation is $F_N = A\tau_1^N + B\tau_2^N$, where

$$\tau_1 = \frac{1 + \sqrt{5}}{2} = 1/0.618, \quad \tau_2 = \frac{1 - \sqrt{5}}{2}.$$

It follows that $F_{N-1}/F_N \sim 0.618$ and the rate of convergence of this line search approach is linear.

4.2 Newton’s method

The best known method of line search is Newton’s method. Assume not only that the function is continuous but also that it is smooth. Given the first and second derivatives of the function at x_n , one can write the Taylor expansion:

$$f(x) \approx q(x) = f(x_n) + f'(x_n)(x - x_n) + f''(x_n)(x - x_n)^2/2.$$

The minimum of $q(x)$ is attained at

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}.$$

(Note that this approach can be generalized to the problem of finding the zeros of the function $g(x) = q'(x)$.)

Theorem 7 *Let the function g have a continuous second derivative and let x^* be such that $g(x^*) = 0$ and $g'(x^*) \neq 0$. Then the Newton method converges with an order of convergence of at least two, provided that x_0 is sufficiently close to x^* .*

4.3 Applying line-search methods

In order for Matlab to be able to read/write files in disk D you should use the command

```
>> cd d;
```

Now you can write the function:

```
function y = humps(x)
y = 1./((x-0.3).^2 + 0.01)+ 1./((x - 0.9).^2 + 0.04) -6;
```

in the M-file `humps.m` in directory D.

```
>> fplot('humps', [-5 5])
>> grid on
>> fplot('humps', [-5 5 -10 25])
>> grid on
>> fplot(' [2*sin(x+3), humps(x)]', [-5 5])
>> fmin('humps',0.3,1)
```

```
ans =
```

```
0.6370
```

```
>> fmin('humps',0.3,1,1)
```

Func evals	x	f(x)	Procedure
1	0.567376	12.9098	initial
2	0.732624	13.7746	golden
3	0.465248	25.1714	golden
4	0.644416	11.2693	parabolic
5	0.6413	11.2583	parabolic
6	0.637618	11.2529	parabolic
7	0.636985	11.2528	parabolic
8	0.637019	11.2528	parabolic
9	0.637052	11.2528	parabolic

```
ans =
```

```
0.6370
```

4.4 Homework

1. Consider the iterative process

$$x_{n+1} = \frac{1}{2} \left(x_n + \frac{a}{x_n} \right),$$

where $a > 0$. Assuming the process converges, to what does it converge? What is the order of convergence.

2. Find the minimum of the function -humps. Use different ranges.
- 3(a) Given $f(x_n)$, $f'(x_n)$ and $f'(x_{n-1})$, show that

$$q(x) = f(x) + f'(x_n)(x - x_n) + \frac{f'(x_{n-1}) - f'(x_n)}{x_{n-1} - x_n} \cdot \frac{x - x_n)^2}{2},$$

has the same derivatives as f at x_n and x_{n-1} and is equal to f at x_n .

- (b) Construct a line search algorithm based on this quadratic fit.

4.5 Quadratic interpolation

Assume we are given $x_1 < x_2 < x_3$ and the values of $f(x_i)$, $i = 1, 2, 3$, which satisfy

$$f(x_2) < f(x_1) \quad \text{and} \quad f(x_2) < f(x_3).$$

The quadratic passing through these points is given by

$$q(x) = \sum_{i=1}^3 f(x_i) \frac{\prod_{j \neq i} (x - x_j)}{\prod_{j \neq i} (x_i - x_j)}.$$

The minimum of this function is attained at the point

$$x_4 = \frac{1}{2} \frac{\beta_{23}f(x_1) + \beta_{31}f(x_2) + \beta_{12}f(x_3)}{\gamma_{23}f(x_1) + \gamma_{31}f(x_2) + \gamma_{12}f(x_3)},$$

with $\beta_{ij} = x_i^2 - x_j^2$ and $\gamma_{ij} = x_i - x_j$. An algorithm $A : \mathcal{R}^3 \rightarrow \mathcal{R}^3$ can be defined by such a pattern. If we start from an initial 3-points pattern $\mathbf{x} = (x_1, x_2, x_3)$ the algorithm A can be constructed in such a way that $A(\mathbf{x})$ has the same pattern. The algorithm is continuous, hence closed. It descends with respect to the function $Z(\mathbf{x}) = f(x_1) + f(x_2) + f(x_3)$. It follows that the algorithm converges to the solution set $\Gamma = \{\mathbf{x}^* : f'(x_i^*) = 0, i = 1, 2, 3\}$. It can be shown that the order of convergence to the solution is (approximately) 1.3.

4.6 Cubic fit

Given x_1 and x_2 , together with $f'(x_1)$, $f''(x_1)$, $f'(x_2)$ and $f''(x_2)$, one can consider a cubic polynom of the form

$$q(x) = a_0 + a_1x + a_2x^2 + a_3x^3.$$

The local minimum is determined by the solution of the equation

$$q'(x) = a_1 + 2a_2x + 3a_3x^2 = 0,$$

which satisfies

$$q''(x) = 2a_2x + 6a_3x > 0.$$

It follows that the appropriate interpolation is given by

$$x_3 = x_2 - (x_2 - x_1) \frac{f'(x_2) + \beta_2 - \beta_1}{f'(x_2) - f'(x_1) + 2\beta_2},$$

where

$$\begin{aligned}\beta_1 &= f'(x_1) + f'(x_2) - 3 \frac{f(x_1) - f(x_2)}{x_1 - x_2} \\ \beta_2 &= (\beta_1^2 - f'(x_1)f'(x_2))^{1/2}.\end{aligned}$$

The order of convergence of this algorithm is 2.

4.7 Homework

1. What conditions on the values and derivatives at two points guarantee that a cubic fit will have a minimum between the two points? Use the answer to develop a search scheme that is globally convergent for unimodal functions.
2. Suppose the continuous real-valued function f satisfies

$$\min_{0 \leq x} f(x) < f(0).$$

Starting at any $x > 0$ show that, through a series of halving and doubling of x and evaluation of the corresponding $f(x)$'s, a three-point pattern can be determined.

3. Consider the function

$$f(x, y) = e^x(4x^2 + 2y^2 + 4xy + 2y + 1).$$

Use the function `fmin` to plot the function

$$g(y) = \min_x f(x, y).$$

5 The method of steepest decent

The method of steepest decent is a method of searching for the minimum of a function of many variables f . In each iteration of this algorithm a line search is performed in the direction of the steepest decent of the function at the current location. In other words,

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha_n \dot{f}(\mathbf{x}_n),$$

where α_n is the nonnegative scalar that minimizes $f(\mathbf{x}_n - \alpha \dot{f}(\mathbf{x}_n))$. It can be shown that relative to the solution set $\{\mathbf{x}^* : \dot{f}(\mathbf{x}^*) = \mathbf{0}\}$, the algorithm is decending and closed, thus converging.

5.1 The quadratic case

Assume

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}' Q \mathbf{x} - b \mathbf{x}' \mathbf{b} = \frac{1}{2} (\mathbf{x} - \mathbf{x}^*)' Q (\mathbf{x} - \mathbf{x}^*) - \frac{1}{2} \mathbf{x}^{*'} Q \mathbf{x}^*,$$

where Q a positive definite and symmetric matrix and $\mathbf{x}^* = Q^{-1}b$ is the minimizer of f . Note that in this case $\dot{f}(\mathbf{x}) = Q\mathbf{x} - \mathbf{b}$. and

$$f(\mathbf{x}_n - \alpha \dot{f}(\mathbf{x}_n)) = \frac{1}{2} (\mathbf{x}_n - \alpha \dot{f}(\mathbf{x}_n))' Q (\mathbf{x}_n - \alpha \dot{f}(\mathbf{x}_n)) - (\mathbf{x}_n - \alpha \dot{f}(\mathbf{x}_n))' \mathbf{b},$$

which is minimized at

$$\alpha_n = \frac{\dot{f}(\mathbf{x}_n)' \dot{f}(\mathbf{x}_n)}{\dot{f}(\mathbf{x}_n)' Q \dot{f}(\mathbf{x}_n)}.$$

It follows that

$$\begin{aligned} \frac{1}{2} (\mathbf{x}_{n+1} - \mathbf{x}^*)' Q (\mathbf{x}_{n+1} - \mathbf{x}^*) = \\ \left\{ 1 - \frac{(\dot{f}(\mathbf{x}_n)' \dot{f}(\mathbf{x}_n))^2}{\dot{f}(\mathbf{x}_n)' Q \dot{f}(\mathbf{x}_n) \dot{f}(\mathbf{x}_n)' Q^{-1} \dot{f}(\mathbf{x}_n)} \right\} \times \frac{1}{2} (\mathbf{x}_n - \mathbf{x}^*)' Q (\mathbf{x}_n - \mathbf{x}^*). \end{aligned}$$

Theorem 8 (Kantorovich inequality) Let Q be a positive definite and symmetric matrix and let $0 < a = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n = A$ be the eigenvalues. Then

$$\frac{(\mathbf{x}'\mathbf{x})^2}{(\mathbf{x}'Q\mathbf{x})(\mathbf{x}'Q^{-1}\mathbf{x})} \geq \frac{4aA}{(a+A)^2}.$$

Theorem 9 For the quadratic case

$$\frac{1}{2}(\mathbf{x}_{n+1} - \mathbf{x}^*)'Q(\mathbf{x}_{n+1} - \mathbf{x}^*) \leq \left(\frac{A-a}{A+a}\right)^2 \frac{1}{2}(\mathbf{x}_n - \mathbf{x}^*)'Q(\mathbf{x}_n - \mathbf{x}^*).$$

5.2 Applying the method in Matlab

Write the M-file `fun.m`:

```
function f=fun(x)
f = exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1);
```

In the Matlab session:

```
>> x=[-1,1];
>> x=fminu('fun',x)
x =
    0.5000   -1.0000
>> fun(x)
ans =
    1.3029e-010
>> x=[-1,1];
>> options(6)=2;
>> x = fminu('fun',x,options)
x =
    0.5000   -1.0000
```

Write the M-file `fun1.m`:

```
function f = fun1(x)
f = 100*(x(2)-x(1)^2)^2 + (1 - x(1))^2;
```

and in the Matlab session:

```

>> x=[-1,1];
>> options(1)=1;
>> options(6)=0;
>> x = fminu('fun1',x,options)
f-COUNT  FUNCTION  STEP-SIZE  GRAD/SD
      4      4      0.500001    -16
      9 3.56611e-009  0.500001    0.0208
     14 7.36496e-013  0.000915682  -3.1e-006
     21 1.93583e-013  9.12584e-005  -1.13e-006
     24 1.55454e-013  4.56292e-005  -7.16e-007
Optimization Terminated Successfully
Search direction less than 2*options(2)
Gradient in the search direction less than 2*options(3)
NUMBER OF FUNCTION EVALUATIONS=24
x =
    1.0000    1.0000
>> x=[-1,1];
>> options(6)=2;
>> x = fminu('fun1',x,options)
f-COUNT  FUNCTION  STEP-SIZE  GRAD/SD
      4      4      0.500001    -16
      9 3.56611e-009  0.500001    0.0208
     15 1.11008e-012  0.000519178  -4.82e-006

Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND = 1.931503e-017.
> In c:\matlab\toolbox\optim\cubici2.m at line 10
  In c:\matlab\toolbox\optim\searchq.m at line 54
  In c:\matlab\toolbox\optim\fminu.m at line 257

....

     192 4.56701e-013 -4.52912e-006  -1.02e-006
     195 4.5539e-013  2.26456e-006  -4.03e-007
     198 4.55537e-013 -1.13228e-006  -1.02e-006
     201 4.55336e-013  5.66141e-007  -4.03e-007
Maximum number of function evaluations exceeded;
increase options(14).
x =
    1.0000    1.0000

```

5.3 Homework

1. Investigate the function

$$f(x, y) = 100(y - x^2)^2 + (1 - x)^2.$$

Why doesn't the steepest decent algorithm converge?

6 Newton and quasi-Newton methods

6.1 Newton's method

Based on the Taylor expansion

$$f(\mathbf{x}_n) \approx f(\mathbf{x}_n) + \dot{f}(\mathbf{x}_n)'(\mathbf{x} - \mathbf{x}_n) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_n)'\ddot{f}(\mathbf{x}_n)(\mathbf{x} - \mathbf{x}_n)$$

one can derive, just as for the line-search problem, Newton's method:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - (\ddot{f}(\mathbf{x}_n))^{-1}\dot{f}(\mathbf{x}_n).$$

Under the regularity conditions it can be shown that local rate of convergence of this method is 2. A modification of this approach is to set

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha_n(\ddot{f}(\mathbf{x}_n))^{-1}\dot{f}(\mathbf{x}_n),$$

where α_n minimizes the function $f(\mathbf{x}_n - \alpha(\ddot{f}(\mathbf{x}_n))^{-1}\dot{f}(\mathbf{x}_n))$.

6.2 Extensions

Consider the approach of choosing

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha_n S_n \dot{f}(\mathbf{x}_n),$$

where S_n is some symmetric and positive-definite matrix and α_n is the non-negative scalar that minimizes $f(\mathbf{x}_n - \alpha S_n \dot{f}(\mathbf{x}_n))$. It can be shown that since S_n is positive-definite the algorithm is descending.

Assume

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}'Q\mathbf{x} - b\mathbf{x}'\mathbf{b} = \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)'Q(\mathbf{x} - \mathbf{x}^*) - \frac{1}{2}\mathbf{x}^{*'}Q\mathbf{x}^*,$$

were Q a positive definite and symmetric matrix and $\mathbf{x}^* = Q^{-1}\mathbf{b}$ is the minimizer of f . Note that in this case $\dot{f}(\mathbf{x}) = Q\mathbf{x} - \mathbf{b}$. and

$$f(\mathbf{x}_n - \alpha S_n \dot{f}(\mathbf{x}_n)) = \frac{1}{2}(\mathbf{x}_n - \alpha S_n \dot{f}(\mathbf{x}_n))' Q (\mathbf{x}_n - \alpha S_n \dot{f}(\mathbf{x}_n)) - (\mathbf{x}_n - \alpha S_n \dot{f}(\mathbf{x}_n))' \mathbf{b},$$

which is minimized at

$$\alpha_n = \frac{\dot{f}(\mathbf{x}_n)' S_n \dot{f}(\mathbf{x}_n)}{\dot{f}(\mathbf{x}_n)' S_n Q S_n \dot{f}(\mathbf{x}_n)}.$$

It follows that

$$\begin{aligned} \frac{1}{2}(\mathbf{x}_{n+1} - \mathbf{x}^*)' Q (\mathbf{x}_{n+1} - \mathbf{x}^*) = \\ \left\{ 1 - \frac{(\dot{f}(\mathbf{x}_n)' S_n \dot{f}(\mathbf{x}_n))^2}{\dot{f}(\mathbf{x}_n)' S_n Q S_n \dot{f}(\mathbf{x}_n) \dot{f}(\mathbf{x}_n)' Q^{-1} \dot{f}(\mathbf{x}_n)} \right\} \times \frac{1}{2}(\mathbf{x}_n - \mathbf{x}^*)' Q (\mathbf{x}_n - \mathbf{x}^*). \end{aligned}$$

Theorem 10 *For the quadratic case*

$$\frac{1}{2}(\mathbf{x}_{n+1} - \mathbf{x}^*)' Q (\mathbf{x}_{n+1} - \mathbf{x}^*) \leq \left(\frac{B_n - b_n}{B_n + b_n} \right)^2 \frac{1}{2}(\mathbf{x}_n - \mathbf{x}^*)' Q (\mathbf{x}_n - \mathbf{x}^*),$$

where B_n and b_n are the largest and smallest eigenvalues of SQ .

6.3 The Davidon-Fletcher-Powell (DFP) method

This is a rank-two correction procedure. The algorithm starts with some positive-definite algorithm S_0 , initial point \mathbf{x}_0 :

1. Minimizes $f(\mathbf{x}_n) - \alpha S_n \dot{f}(\mathbf{x}_n)$ to obtain \mathbf{x}_{n+1} , $\Delta_k \mathbf{x} = \mathbf{x}_{n+1} - \mathbf{x}_n = -\alpha_n S_n \dot{f}(\mathbf{x}_n)$, $\dot{f}(\mathbf{x}_{n+1})$ and $\Delta_n \dot{f} = \dot{f}(\mathbf{x}_{n+1}) - \dot{f}(\mathbf{x}_n)$.
2. Set

$$S_{n+1} = S_n + \frac{\Delta_k \mathbf{x}' \Delta_k \mathbf{x}}{\Delta_k \mathbf{x}' \Delta_k \dot{f}} - \frac{S_n \Delta_k \dot{f} \Delta_k \dot{f}' S_n}{\Delta_k \dot{f}' S_n \Delta_k \dot{f}}.$$

3. Go to 1.

It follows, since $\Delta_k \mathbf{x}' \dot{f}(\mathbf{x}_{n+1}) = 0$, that if S_n is positive definite then so is S_{n+1} .

6.4 The Broyden-Fletcher-Goldfarb-Shanno (BFGS) method

In this method the Hessian is approximated. This is a rank-two correction procedure as well. The algorithm starts with some positive-definite algorithm H_0 , initial point \mathbf{x}_0 :

1. Minimizes $f(\mathbf{x}_n) - \alpha H_n^{-1} \dot{f}(\mathbf{x}_n)$ to obtain \mathbf{x}_{n+1} , $\Delta_k \mathbf{x} = \mathbf{x}_{n+1} - \mathbf{x}_n = -\alpha H_n^{-1} \dot{f}(\mathbf{x}_n)$, $\dot{f}(\mathbf{x}_{n+1})$ and $\Delta_n \dot{f} = \dot{f}(\mathbf{x}_{n+1}) - \dot{f}(\mathbf{x}_n)$.
2. Set

$$H_{n+1} = H_n + \frac{\Delta_k \dot{f}' \Delta_k \dot{f}}{\Delta_k \dot{f}' \Delta_k \mathbf{x}} - \frac{H_n \Delta_k \mathbf{x} \Delta_k \mathbf{x}' H_n}{\Delta_k \mathbf{x}' H_n \Delta_k \mathbf{x}}.$$

3. Go to 1.

6.5 Homework

1. Use the function `fminu` with `options(6)=0` (BFGS), `options(6)=1` (DFP) and `options(6)=2` (steepest descent) to compare the performance of the algorithms. Apply the function to minimize $f(\mathbf{x}) = \mathbf{x}' Q \mathbf{x}$, here Q is a diagonal matrix. Use different ratios between the smallest and the largest eigenvalues different dimensions.
2. Investigate the rate of convergence of the algorithm

$$\mathbf{x}_{n+1} = \mathbf{x}_n - [\delta I + (\ddot{f}(\mathbf{x}_n))^{-1}] \dot{f}(\mathbf{x}_n).$$

What is the rate if δ is larger than the smallest eigenvalue of $(\ddot{f}(\mathbf{x}^*))^{-1}$?

3. Use the formula

$$[A + \mathbf{b}\mathbf{a}']^{-1} = A^{-1} - \frac{A^{-1} \mathbf{a}\mathbf{b}' A^{-1}}{1 + \mathbf{b}' A^{-1} \mathbf{a}},$$

in order to get a direct updating formula for the inverse of H_n in the BFGS method.

7 Constrained Minimization Conditions

The general (constrained) optimization problem has the form:

$$\min_{\mathbf{x} \in \mathcal{R}^d} f(\mathbf{x})$$

subject to:

$$\begin{aligned} g_i(\mathbf{x}) &= 0 & i = 1, \dots, m_e \\ g_i(\mathbf{x}) &\leq 0 & i = m_e + 1, \dots, m \\ \mathbf{x}_l &\leq \mathbf{x} \leq \mathbf{x}_u \end{aligned}$$

The first m_e constraints are called *equality constraints* and the last $m - m_e$ constraints are the *inequality constraints*.

7.1 Necessary conditions (equality constraints)

We assume first that $m_e = m$ — all constraints are equality constraints. Let \mathbf{x}^* be a solution of the optimization problem. Let $\mathbf{g} = (g_1, \dots, g_m)$. Note that \mathbf{g} is a (non-linear) transformation from \mathcal{R}^d into \mathcal{R}^m . The set $\{\mathbf{x} \in \mathcal{R}^n : \mathbf{g}(\mathbf{x}) = \mathbf{0}\}$ is a surface in \mathcal{R}^n . This surface is approximated near \mathbf{x}^* by $\mathbf{x}^* + M$, where

$$M = \{\mathbf{y} : \dot{\mathbf{g}}(\mathbf{x}^*)' \mathbf{y} = \mathbf{0}\}.$$

In order for this approximation to hold, \mathbf{x}^* should be a *regular point* of the constraint, i.e. $(\dot{g}_1(\mathbf{x}^*), \dots, \dot{g}_m(\mathbf{x}^*))$ should be linearly independent.

Theorem 11 (Lagrange multipliers) *Let \mathbf{x}^* be a local extremum point of f subject to the constraint $\mathbf{g} = \mathbf{0}$. Assume that \mathbf{x}^* is a regular point of these constraints. Then there is a $\lambda \in \mathcal{R}^m$ such that*

$$\dot{f}(\mathbf{x}^*) + \dot{\mathbf{g}}(\mathbf{x}^*) \lambda = \dot{f}(\mathbf{x}^*) + \sum_{j=1}^m \lambda_j \dot{\mathbf{g}}_j(\mathbf{x}^*) = \mathbf{0}.$$

Given λ , one can consider the *Lagrangian*:

$$l(\mathbf{x}, \lambda) = f(\mathbf{x}) + \mathbf{g}(\mathbf{x}) \lambda.$$

The necessary conditions can be formulated as $\dot{l} = \mathbf{0}$. The matrix of partial second derivatives of l (with respect to \mathbf{x}) at \mathbf{x}^* is

$$\ddot{l}_{\mathbf{x}}(\mathbf{x}^*) = \ddot{f}(\mathbf{x}^*) + \ddot{\mathbf{g}}(\mathbf{x}^*) \lambda = \ddot{f}(\mathbf{x}^*) + \sum_{j=1}^m \ddot{\mathbf{g}}_j(\mathbf{x}^*) \lambda_j$$

We say that this matrix is positive semidefinite over M if $\mathbf{x}'\ddot{l}_{\mathbf{x}}(\mathbf{x}^*)\mathbf{x} \geq 0$, for all $\mathbf{x} \in M$.

Theorem 12 (Second-order condition) *Let \mathbf{x}^* be a local extremum point of f subject to the constraint $\mathbf{g} = \mathbf{0}$. Assume that \mathbf{x}^* is a regular point of these constraints, and let $\lambda \in \mathcal{R}^m$ be such that*

$$\dot{f}(\mathbf{x}^*) + \dot{\mathbf{g}}(\mathbf{x}^*)\lambda = \dot{f}(\mathbf{x}^*) + \sum_{j=1}^m \lambda_j \dot{\mathbf{g}}_j(\mathbf{x}^*) = \mathbf{0}.$$

Then the matrix $\ddot{l}_{\mathbf{x}}(\mathbf{x}^)$ is positive semidefinite over M .*

We now consider the case where $m_e < m$. Let \mathbf{x}^* be a solution of the constrained optimization problem. A constraint g_j is *active* at \mathbf{x}^* if $g_j(\mathbf{x}^*) = 0$ and it is *inactive* if $g_j(\mathbf{x}^*) < 0$. Note that all equality constraints are active. Denote by J the set of all active constraints.

7.2 Necessary conditions (inequality constraints)

For the consideration of necessary conditions when inequality constraints are present the definition of a regular point should be extended. We say now that \mathbf{x}^* is regular if $\{\dot{g}_j(\mathbf{x}^*) : j \in J\}$ are linearly independent.

Theorem 13 (Kuhn-Tucker Conditions) *Let \mathbf{x}^* be a local extremum point of f subject to the constraint $g_j(\mathbf{x}) = 0$, $1 \leq j \leq m_e$ and $g_j(\mathbf{x}) \leq 0$, $m_e + 1 \leq j \leq m$. Assume that \mathbf{x}^* is a regular point of these constraints. Then there is a $\lambda \in \mathcal{R}^m$ such that $\lambda_j \geq 0$, for all $j > m_e$, and*

$$\begin{aligned} \dot{f}(\mathbf{x}^*) + \sum_{j=1}^m \lambda_j \dot{\mathbf{g}}_j(\mathbf{x}^*) &= \mathbf{0} \\ \sum_{j=m_e+1}^m \lambda_j \dot{\mathbf{g}}_j(\mathbf{x}^*) &= \mathbf{0}. \end{aligned}$$

Let

$$\ddot{l}_{\mathbf{x}}(\mathbf{x}^*) = \ddot{f}(\mathbf{x}^*) + \sum_{j=1}^m \lambda_j \ddot{\mathbf{g}}_j(\mathbf{x}^*).$$

Theorem 14 (Second-order condition) *Let \mathbf{x}^* be a local extremum point of f subject to the constraint $g_j(\mathbf{x}) = 0$, $1 \leq j \leq m_e$ and $g_j(\mathbf{x}) \leq 0$, $m_e + 1 \leq j \leq m$. Assume that \mathbf{x}^* is a regular point of these constraints, and let $\lambda \in \mathcal{R}^m$ be such that $\lambda_j \geq 0$, for all $j > m_e$, and*

$$\dot{f}(\mathbf{x}^*) + \sum_{j=1}^m \lambda_j \dot{\mathbf{g}}_j(\mathbf{x}^*) = \mathbf{0}.$$

Then the matrix $\ddot{l}_{\mathbf{x}}(\mathbf{x}^)$ is positive semidefinite on the tangent subspace of the active constraints.*

7.3 Sufficient conditions

Sufficient conditions are based on second-order conditions:

Theorem 15 (Equality constraints) *Suppose there is a point \mathbf{x}^* satisfying $\mathbf{g}(\mathbf{x}^*) = \mathbf{0}$, and a $\lambda \in \mathcal{R}^m$ such that*

$$\dot{f}(\mathbf{x}^*) + \sum_{j=1}^m \lambda_j \dot{\mathbf{g}}_j(\mathbf{x}^*) = \mathbf{0}.$$

Suppose also that the matrix $\ddot{l}_{\mathbf{x}}(\mathbf{x}^)$ is positive definite on M . Then \mathbf{x}^* is a strict local minimum for the constrained optimization problem.*

Theorem 16 (Inequality constraints) *Suppose there is a point \mathbf{x}^* that satisfies the constraints. A sufficient condition for \mathbf{x}^* to be a strict local minimum for the constrained optimization problem is the existence of a $\lambda \in \mathcal{R}^m$ such that $\lambda_j \geq 0$, for $m_e < j \leq m$, and*

$$\dot{f}(\mathbf{x}^*) + \sum_{j=1}^m \lambda_j \dot{\mathbf{g}}_j(\mathbf{x}^*) = \mathbf{0} \tag{1}$$

$$\sum_{j=m_e+1}^m \lambda_j \mathbf{g}_j(\mathbf{x}^*) = \mathbf{0}, \tag{2}$$

and the Hessian matrix $\ddot{l}_{\mathbf{x}}(\mathbf{x}^)$ is positive on the subspace*

$$M' = \{\mathbf{y} : \dot{\mathbf{g}}_j(\mathbf{x}^*)' \mathbf{y} = 0, j \in J\}$$

where $J = \{j : \mathbf{g}_j(\mathbf{x}^) = 0, \lambda_j > 0\}$*

7.4 Sensitivity

The Lagrange multipliers can be interpreted as the price of incremental change in the constraints. Consider the class of problems:

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{g}(\mathbf{x}) = \mathbf{c}. \end{array}$$

For each \mathbf{c} , assume the existence of a solution point $\mathbf{x}^*(\mathbf{c})$. Under appropriate regularity conditions the function $\mathbf{x}^*(\mathbf{c})$ is well behaved with $\mathbf{x}^*(\mathbf{0}) = \mathbf{x}^*$.

Theorem 17 (Sensitivity Theorem) *Let $f, \mathbf{g} \in C^2$ and consider the family of problem defined above. Suppose that for $\mathbf{c} = \mathbf{0}$ there is a local solution \mathbf{x}^* that is a regular point and that, together with its associated Lagrange multiplier vector λ , satisfies the second-order sufficient conditions for a strict local minimum. Then for every \mathbf{c} in a neighborhood of $\mathbf{0}$ there is $\mathbf{x}^*(\mathbf{c})$, continuous in \mathbf{c} , such that $\mathbf{x}^*(\mathbf{0}) = \mathbf{x}^*$, $\mathbf{x}^*(\mathbf{c})$ is a local minimum of the constrained problem indexed by \mathbf{c} , and*

$$\dot{f}(\mathbf{x}^*(\mathbf{c}))|_{\mathbf{c}=\mathbf{0}} = -\lambda.$$

7.5 Homework

1. Read the help file on the function `fminu`. Investigate the effect of supplying the gradients with the parameter `grad` on the performance of the procedure. Compare, in particular the functions `bilinear` and `fun1`.
2. Consider the constraints $x_1 \geq 0$, $x_2 \geq 0$ and $(x_2 - x_1 - 1)^2 \leq 0$. Show that $(1, 0)$ is feasible but not regular.
3. Find the rectangle of given perimeter that has greatest area by solving the first-order necessary conditions. Verify that the second-order sufficient conditions are satisfied.
4. Three types of items are to be stored. Item A costs one dollar, item B costs two dollars and item C costs 4 dollars. The demand for the three items are independent and uniformly distributed in the range $[0, 3000]$. How many of each type should be stored if the total budget is 4,000 dollars?

5. Let A be an $n \times m$ matrix of rank m and let L be an $n \times n$ matrix that is symmetric and positive-definite on the subspace $M = \{\mathbf{y} : A\mathbf{y} = \mathbf{0}\}$. Show that the $(n + m) \times (n + m)$ matrix

$$\begin{bmatrix} L & A' \\ A & \mathbf{0} \end{bmatrix}$$

is non-singular.

6. Consider the quadratic program

$$\begin{aligned} & \text{minimize} && \mathbf{x}'Q\mathbf{x} - 2\mathbf{b}'\mathbf{x} \\ & \text{subject to} && A\mathbf{x} = \mathbf{c}. \end{aligned}$$

Prove that \mathbf{x}^* is a local minimum point if and only if it is a global minimum point.

7. Maximize $14x - x^2 + 6y - y^2 + 7$ subject to $x + y \leq 2$, $x + 2y \leq 3$.

8 Lagrange methods

The Lagrange methods for dealing with constrained optimization problem are based on solving the Lagrange first-order necessary conditions. In particular, for solving the problem with equality constraints only:

$$\begin{aligned} & \text{minimize} && f(\mathbf{x}) \\ & \text{subject to} && \mathbf{g}(\mathbf{x}) = \mathbf{0}, \end{aligned}$$

the algorithms look for solutions of the problem:

$$\begin{aligned} f'(\mathbf{x}) + \sum_{j=1}^m \lambda_j g'_j(\mathbf{x}) &= \mathbf{0} \\ \mathbf{g}(\mathbf{x}) &= \mathbf{0}, \end{aligned}$$

8.1 Quadratic programming

An important special case is when the target function f is quadratic and the constraints are linear:

$$\begin{aligned} & \text{minimize} && (1/2)\mathbf{x}'Q\mathbf{x} + \mathbf{x}'\mathbf{c} \\ & \text{subject to} && \mathbf{a}'_i\mathbf{x} = b_i, \quad 1 \leq i \leq m_e \\ & && \mathbf{a}'_i\mathbf{x} \leq b_i, \quad m_e + 1 \leq i \leq m \end{aligned}$$

with Q a symmetric matrix.

8.1.1 Equality constraints

In the particular case where $m_e = m$ the above becomes

$$\begin{aligned} & \text{minimize} && (1/2)\mathbf{x}'Q\mathbf{x} + \mathbf{x}'\mathbf{c} \\ & \text{subject to} && A\mathbf{x} = \mathbf{b}. \end{aligned}$$

and the Lagrange necessary conditions become

$$\begin{aligned} Q\mathbf{x} + A'\lambda + \mathbf{c} &= \mathbf{0} \\ A\mathbf{x} - \mathbf{b} &= \mathbf{0}. \end{aligned}$$

This system is nonsingular if Q is positive definite on the subspace $M = \{\mathbf{x} : A\mathbf{x} = \mathbf{0}\}$. and the solution becomes:

$$\begin{aligned} \mathbf{x} &= Q^{-1}A'(AQ^{-1}A')^{-1}[AQ^{-1}\mathbf{c} + \mathbf{b}] - Q^{-1}\mathbf{c} \\ \lambda &= -(AQ^{-1}A')^{-1}[AQ^{-1}\mathbf{c} + \mathbf{b}]. \end{aligned}$$

8.1.2 Inequality constraints

In the general quadratic programming problem the method of *active set* is used. A working set of constraints W_n is updated in each iteration. The set W_n contains all constraints that are suspected to satisfy an equality relation at the solution point. In particular, it contains the equality constraints. An algorithm for solving the general quadratic problem is:

1. Start with a feasible point \mathbf{x}_0 and a working set W_0 . Set $n = 0$
2. Solve the quadratic problem

$$\begin{aligned} & \text{minimize} && (1/2)\mathbf{d}'Q\mathbf{d} + (\mathbf{c} + Q\mathbf{x}_n)'\mathbf{d} \\ & \text{subject to} && \mathbf{a}'_i\mathbf{d} = 0, \quad i \in W_n. \end{aligned}$$

If $\mathbf{d}_n^* = \mathbf{0}$ go to 4.

3. Set $x_{n+1} = \alpha_n \mathbf{d}_n^*$, where

$$\alpha_n = \min_{\mathbf{a}'_i\mathbf{d}_n^* > 0} \left\{ 1, \frac{b_i - \mathbf{a}'_i\mathbf{x}_n}{\mathbf{a}'_i\mathbf{d}_n^*} \right\}.$$

If $\alpha_n < 1$, adjoin the minimizing index above to W_n to form W_{n+1} . Set $n = n + 1$ and return to step 2.

4. Compute the Lagrange multiplier in step 3 and let $\lambda_n = \min\{\lambda_i : i \in W_n, i > m_e\}$. If $\lambda_n \geq 0$, stop; \mathbf{x}_n is a solution. Otherwise, drop λ_n from W_n to form W_{n+1} and return to step 2.

8.2 Homework

1. Read about the function `constr`.
2. Investigate the properties of the function for quadratic programming.

9 Sequential Quadratic Programming

Let us go back to the general problem

$$\begin{aligned} & \text{minimize} && f(\mathbf{x}) \\ & \text{subject to} && \mathbf{g}_i(\mathbf{x}) = 0, \quad i = 1, \dots, m_e \\ & && \mathbf{g}_i(\mathbf{x}) \leq 0, \quad i = m_e + 1, \dots, m. \end{aligned}$$

The SQP method solves this problem by solving a sequence of QP problems where the Lagrangian function l is approximated by a quadratic function and the constraints are approximated by a linear hyper-space.

9.1 Newton's Method

Consider the case of equality constraints only. At each iteration the problem

$$\begin{aligned} & \text{minimize} && (1/2)\mathbf{d}'\ddot{l}_{\mathbf{x}}(\mathbf{x}_n, \lambda_n)\mathbf{d} + \dot{l}(\mathbf{x}_n, \lambda_n)'\mathbf{d} \\ & \text{subject to} && \dot{g}_i(\mathbf{x}_n)'\mathbf{d} + g_i(\mathbf{x}_n) = 0, \quad i = 1, \dots, m \end{aligned}$$

is solved. It can be shown that the rate of convergence of this algorithm is 2 (at least) if the starting point $(\mathbf{x}_0, \lambda_0)$ is close enough to the solution $(\mathbf{x}^*, \lambda^*)$. A disadvantage of this approach is the need to compute Hessian.

9.2 Structured Methods

These methods are modifications of the basic Newton method, with approximations replacing Hessian. One can rewrite the solution to the Newton step in the form

$$\begin{bmatrix} \mathbf{x}_{n+1} \\ \lambda_{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_n \\ \lambda_n \end{bmatrix} - \begin{bmatrix} \ddot{l}_n & \dot{\mathbf{g}}_n' \\ \dot{\mathbf{g}}_n & \mathbf{0} \end{bmatrix}^{-1} \begin{bmatrix} \dot{l}_n \\ \mathbf{g}_n \end{bmatrix}.$$

Instead, one can use the formula

$$\begin{bmatrix} \mathbf{x}_{n+1} \\ \lambda_{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_n \\ \lambda_n \end{bmatrix} - \alpha_n \begin{bmatrix} H_n & \dot{\mathbf{g}}_n' \\ \dot{\mathbf{g}}_n & \mathbf{0} \end{bmatrix}^{-1} \begin{bmatrix} \dot{l}_n \\ \mathbf{g}_n \end{bmatrix},$$

with α_n and H_n properly chosen.

9.3 Merit function

In order to choose the α_n and to assure that the algorithm will converge a *merit function* is associated with the problem such that a solution of the constrained problem is a (local) minimum of the merit function. The algorithm should be descending with respect to the merit function.

Consider, for example, the problem with inequality constraints only:

$$\begin{aligned} & \text{minimize} && f(\mathbf{x}) \\ & \text{subject to} && \mathbf{g}_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, m. \end{aligned}$$

The absolute-value merit function is given by

$$Z(\mathbf{x}) = f(\mathbf{x}) + c \sum_{i=1}^m \mathbf{g}_i(\mathbf{x})^+.$$

The parameter α is chosen by minimizing the merit function in the direction chosen by the algorithm.

Theorem 18 *If H is positive-definite and if $c > \max_{1 \leq i \leq m} \lambda_i$ then the algorithm is descending with respect to the absolute-value merit function.*

9.4 Enlargement of the feasible region

Consider, again, the problem with inequality constraints only:

$$\begin{aligned} & \text{minimize} && f(\mathbf{x}) \\ & \text{subject to} && \mathbf{g}_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, m. \end{aligned}$$

and its solution with a structural SQP algorithm.

Assume that at the current iteration $\mathbf{x}_n = \mathbf{x}$ and $H_n = H$. Then one wants to consider the QP problem:

$$\begin{aligned} & \text{minimize} && (1/2)\mathbf{d}'H\mathbf{d} + \dot{f}(\mathbf{x}) \\ & \text{subject to} && \dot{g}_i(\mathbf{x})'\mathbf{d} + g(\mathbf{x}) \leq 0, \quad i = 1, \dots, m. \end{aligned}$$

However, it is possible that this problem is infeasible at the point \mathbf{x} . Hence, the the original method breaks down. However, one can consider instead the problem

$$\begin{aligned} & \text{minimize} && (1/2)\mathbf{d}'H\mathbf{d} + \dot{f}(\mathbf{x}) + c \sum_{i=1}^m \xi_i \\ & \text{subject to} && \dot{g}_i(\mathbf{x})'\mathbf{d} + g(\mathbf{x}) \leq \xi_i, \quad i = 1, \dots, m, \\ & && -\xi_i \leq 0, \quad i = 1, \dots, m, \end{aligned}$$

which is always feasible.

Theorem 19 *If H is positive-definite and if $c > \max_{1 \leq i \leq m} \lambda_i$ then the algorithm is descending with respect to the absolute-value merit function.*

9.5 The Han–Powell method

The Han–Powell method is what is used by Matlab for SQP. It is a Quasi-Newton method, where H_n is updated using the BFGS approach:

$$H_{n+1} = H_n + \frac{(\Delta l)(\Delta l)'}{(\Delta \mathbf{x})'(\Delta l)} - \frac{H_n(\Delta \mathbf{x})(\Delta \mathbf{x})'H_n}{(\Delta \mathbf{x})'H_n(\Delta \mathbf{x})},$$

where

$$\Delta \mathbf{x} = \mathbf{x}_{n+1} - \mathbf{x}_n, \quad \Delta l = l(\mathbf{x}_{n+1}, \lambda_{n+1}) - l(\mathbf{x}_n, \lambda_n).$$

It can be shown that H_{n+1} is positive-definite if H_n is and if $(\Delta \mathbf{x})'(\Delta l) > 0$.

9.6 Constrained minimization in Matlab

Write the M-file `fun2.m`:

```
function [f,g]=fun2(x)
f = exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1);
g(1,1) = 1.5 + x(1)*x(2) - x(1) - x(2);
g(2,1) = -x(1)*x(2) - 10;
```

and run the Matlab session:

```
>> x0 = [-1,1];
>> x = constr('fun2', x0)
x =
    0.1956    1.0910
>> x = constr('fun2', x0)
x =
   -9.5474    1.0474
>> [f,g] = fun2(x)
f =
    0.0236
g =
  1.0e-015 *
   -0.8882
         0
>> options = [];
>> vlb = [0,0];
>> vlu = [];
>> x = constr('fun2', x0, options, vlb, vlu)
x =
         0    1.5000
>> [f,g] = fun2(x)
f =
    8.5000
g =
         0
   -10
```

Write the M-file `grudf2.m`:

```
function [df,dg]=grudf2(x)
f = exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1);
df = [f + exp(x(1))*(8*x(1) + 4*x(2)),
      exp(x(1))*(4*x(1) + 4*x(2) + 2)];
dg = [x(2) - 1, -x(2);
      x(1) - 1, -x(1)];
```

and run the session:

```
>> vlb = [];
>> x = constr('fun2', x0, options, vlb, vlu, 'grudf2')
```


$$\mathbf{x} = \begin{array}{cc} -9.5474 & 1.0474 \end{array}$$

9.7 Homework

1. Let H be a positive-definite matrix and assume that throughout some compact set the quadratic programming has a unique solution, such that the Lagrange multipliers are not larger than c . Let $\{\mathbf{x}_n : n \geq 0\}$ is a sequence generated by the recursion $\mathbf{x}_{n+1} = \mathbf{x}_n + \alpha_n \mathbf{d}_n$, where \mathbf{d} is the direction found by solving the QP centered at \mathbf{x}_n and with H fixed and α_n is determined by minimization of the function Z . Show that any limit point of $\{\mathbf{x}_n\}$ satisfies the first order necessary conditions for the constrained minimization problem.
2. Extend the result in 1 for the case where $H = H_n$ changes but yet $\epsilon \|\mathbf{x}\|^2 \leq \mathbf{x}' H_n \mathbf{x} \leq c \|\mathbf{x}\|^2$ for some $0 < \epsilon < c < \infty$ and for all \mathbf{x} and n .

10 Penalty and Barrier Methods

The basic approach in these methods is to solve a sequence of unconstrained problems. The solutions of these problems converge to the solution of the original problem.

10.1 Penalty method

Consider the problem

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) \\ \text{subject to} & g_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, m. \end{array}$$

Choose a continuous penalty function which is zero inside the feasible set and positive outside of it. For example,

$$P(\mathbf{x}) = (1/2) \sum_{i=1}^m \max\{0, g_i(\mathbf{x})\}^2.$$

Minimize, for each c , the problem

$$q(c, \mathbf{x}) = f(\mathbf{x}) + cP(\mathbf{x}).$$

When c is increased it is expected that the solution $\mathbf{x}^*(c)$ converges to \mathbf{x}^* .

Lemma 1 *Let $c_{n+1} > c_n$, then*

$$q(c_n, \mathbf{x}_n^*) \leq q(c_{n+1}, \mathbf{x}_{n+1}^*) \quad (3)$$

$$P(\mathbf{x}_n^*) \geq P(\mathbf{x}_{n+1}^*) \quad (4)$$

$$f(\mathbf{x}_n^*) \leq f(\mathbf{x}_{n+1}^*). \quad (5)$$

Lemma 2 *Let \mathbf{x}^* be the solution of the original problem. Then for each n*

$$f(\mathbf{x}^*) \geq q(c_n, \mathbf{x}_n^*) \geq f(\mathbf{x}_n^*).$$

Theorem 20 *Let $\{x_n\}$ be a sequence generated by the penalty method. Then, any limit point of the sequence is a solution to the original problem.*

10.2 Barrier method

Consider again the problem

$$\begin{aligned} & \text{minimize} && f(\mathbf{x}) \\ & \text{subject to} && g_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, m, \end{aligned}$$

and assume that the feasible set is the closure of its interior. A barrier function is a continuous and positive function over the feasible set which goes to ∞ as \mathbf{x} approaches the boundary. For example,

$$B(\mathbf{x}) = \sum_{i=1}^m \frac{1}{g_i(\mathbf{x})}.$$

Minimize, for each c , the problem

$$\begin{aligned} & \text{minimize} && r(c, \mathbf{x}) = f(\mathbf{x}) + \frac{1}{c}B(\mathbf{x}) \\ & \text{subject to} && g_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, m. \end{aligned}$$

Note that a method of unconstrained minimization can be used because the solution is in the interior of the feasible set. When c is increased it is expected that the solution $\mathbf{x}^*(c)$ converges to \mathbf{x}^* .

Theorem 21 *Let $\{x_n\}$ be a sequence generated by the barrier method. Then, any limit point of the sequence is a solution to the original problem.*

10.3 A final project

Choose an algorithm for minimization of a non-linear programming problem. Program the steps of the algorithms using Matlab. Apply your program to a specific example and compare the results to those of the appropriate built-in function. Submit the program + plots and outputs that demonstrate the steps of the algorithm.

10.4 An exercise in Matlab

Use the M-files `fun1.m`, `const1.m` and `Z.m`:

```
function f = fun1(x)
f = 100*(x(2)-x(1)^2)^2 + (1 - x(1))^2;
```

```
function g = cons1(x)
g = x(1)^2 + x(2)^2 - 1.5;
```

```
function a = Z(t,x,d)
y = x + t*d;
if cons1(y) >= 0
    a = fun1(y) + 2*cons1(y);
else
    a = fun1(y);
end
```

and write the M-file `myconstr.m`:

```
function x = myconstr(x0,H,n)
x = x0;
for i = 1:n
    c = gradfun1(x);
    A=gradcons1(x);
    b=-cons1(x);
    d = qp(H,c,A,b);
    a = fmin('Z',0,2,[],x,d);
    x = x + a*d;
end
```

```
function f = fun1(x)
```

```

f = 100*(x(2)-x(1)^2)^2 + (1 - x(1))^2;

function g = cons1(x)
g = x(1)^2 + x(2)^2 - 1.5;

function df = grudfun1(x)
df1 = -400*(x(2) - x(1)^2)*x(1) + 2*x(1) - 2;
df2 = 200*(x(2) - x(1)^2);
df = [df1;df2];

function dg=grudcons1(x);
dg = [2*x(1), 2*x(2)];

Run the Matlab session:

>> x0 = [0;0];
>> constr('f = fun1(x); g = cons1(x);',x0)
ans =
    0.9072
    0.8227
>> H=[1,0;0,1];
>> myconstr(x0,H,1000)
ans =
    0.9131
    0.8329
>> x0 = [0.9;0.8];
>> myconstr(x0,H,10)
ans =
    0.9072
    0.8228
>> x =constr('f = fun1(x); g = cons1(x);',x0);
>> (-400*(x(2) - x(1)^2)*x(1) + 2*x(1) - 2)/(2*x(1))
ans =
   -0.0387
>> 200*(x(2) - x(1)^2)/(2*x(2))
ans =
   -0.0386
>> 100*(grudcons1(x+[0.01;0])-grudcons1(x))
ans =
    2.0000    0
>> 100*(grudcons1(x+[0;0.01])-grudcons1(x))

```

```

ans =
      0      2.0000
>> 100*(fun1grad(x+[0.01;0])-fun1grad(x))
ans =
>>
      671.5124 -364.8935
>> 100*(fun1grad(x+[0;0.01])-fun1grad(x))
ans =
      -362.8935  200.0000
>> H = [672,-364;-364,200] + 0.386*[2,0;0,2];
>> x0 = [0.7;0.5];
>> xx = x0;
>> for n=1:12
x0 = myconstr(x0,H,1);
xx=[xx,x0];
end
>>[x1,x2]=meshgrid(-1.5:0.1:1.5);
>> t = 0:pi/20:2*pi;
>> contour(x1,x2,fun12(x1,x2),30)
>> hold on
>> plot(sqrt(1.5)*exp(i*t))
>> plot(xx(1,:),xx(2,:),'*-')

```

11 Stochastic Approximation

Let h be a monotone function from the interval $[a, b]$ to \mathcal{R} . Let θ be the (unique) solution of the equation $h(x) = 0$. Assume that $h'(\theta) > 0$. Unfortunately, for a given point x , one cannot evaluate $h(x)$ but we observe $Y = h(x) + \varepsilon$, where ε is a random variable with $E\varepsilon = 0$ and $E\varepsilon^2 = \sigma^2 < \infty$, independent of how x is selected. The Robbins-Monroe scheme involves an infinite sequence of positive constants $\{c_n : n = 0, 1, 2, \dots\}$ which satisfy the conditions

$$\sum_{n=0}^{\infty} c_n = \infty, \quad \sum_{n=0}^{\infty} c_n^2 < \infty. \quad (6)$$

The recursion is given by

$$x_n = x_{n-1} - c_{n-1}Y_{n-1}, \quad a < x_0 < b. \quad (7)$$

Theorem 22 *For the recursion defined in (7), with constants c_n satisfying (6), it can be shown that $\mathbb{E}(x_n - \theta)^2 \xrightarrow{n \rightarrow \infty} 0$.*

Theorem 23 *For the recursion defined in (7), with constants c_n satisfying (6), it can be shown that $n^{1/2}(x_n - \theta)$ converges to a zero-mean normal random variable.*