4. Lectures 15, 16: Optimization

Plan:

- ◇ Optimization. Basic problems.
- ◇ Unconstrained optimization. Gradient descent.
- ◇ Constrained optimization. Linear programming.
- ◇ Genetic algorithm.
- ◇ Lagrange multipliers.

4.1. **Basic problems of optimization. . .** A certain *objective function* $f(x)$ is required to take its maximum or minimum value given some constraints on its dependent variables $x = (x_1, x_2, ...)$. Solutions, not necessarily optimal, satisfying the constraints are called *feasible.*

4.2. **Unconstrained optimization.** We want to minimize function $f(x)$ over all $x \in \mathbb{R}^n$. This is *unconstrained* minimization. The function could be a scalar or vector. For simplicity, consider scalar f. The minimum or maximum is reached at the point x_0 where $\nabla f = 0$. For minimum we have to require the second derivative to be positive at that point. For a function of many variables positivity of the second derivative means positive definiteness of the Hessian matrix:

$$H_{ij}(x_0) = \frac{\partial^2 f}{\partial x_i \partial x_j}(x_0) \quad \text{is positive definite.}$$

We denote $x_0 = argmin(f)$ to mean the argument of $f(x)$ that minimizes the value of f.

In Matlab, one can use **fminbnd** function to find the minimum of a function of a single variable. For example, $x = \textbf{fminbnd}('x\hat{}2 * \cos(x)', 3, 4)$ will find the minimum of $f = x^2 \cos x$ in the search interval $[3, 4]$. The algorithm behind **fminbnd** is not based on derivatives, but rather on successively dividing the interval until the minimum is found, similar to the bisection method.

4.3. **Gradient descent. . . . .** Now we look at minimization methods that use the gradient information. If $f(x)$ is a function with a minimum, then $-\nabla f$ is a vector that points "downhill", and this may be used to approach the minimum step by step, by iterations.

The idea is that if we start at x^0, then x^1 is found looking downhill:

$$x^1 = x^0 - \tau \nabla f\left(x^0\right).$$

The main thing is now to choose τ, i.e. figure out how much one should go down at each step. If too much, the point x^1 can actually miss the minimum in the direction down the gradient. Then the right τ should be such that f reaches its minimum as a function of τ:

$$\frac{d}{d\tau} f\left(x^0 - \tau \nabla f\left(x^0\right)\right) = 0.$$

Therefore

$$\nabla f\left(x^1\right) \cdot \nabla f\left(x^0\right) = 0,$$

which means that at x^1, the next downhill direction should be chosen orthogonal to the direction at x^0. This determines τ.

Example 33. Find the minimum of $f = x^2 + 3y^2$ using the gradient descent.

The gradient is

$$\nabla f = (2x, 6y).$$

Then the descent algorithm is given by

$$\begin{bmatrix} x^{n+1} \\ y^{n+1} \end{bmatrix} = \begin{bmatrix} x^n \\ y^n \end{bmatrix} - \tau \begin{bmatrix} 2x^n \\ 6y^n \end{bmatrix} = \begin{bmatrix} (1-2\tau)\, x^n \\ (1-6\tau)\, y^n \end{bmatrix} = \begin{bmatrix} 1-2\tau & 0 \\ 0 & 1-6\tau \end{bmatrix} \begin{bmatrix} x^n \\ y^n \end{bmatrix},$$

where τ is found from the orthogonality

$$\begin{aligned}\nabla f\left(x^{n+1}\right)\cdot\nabla f\left(x^{n}\right)&=0\\ \left(2x^{n+1},6y^{n+1}\right)\cdot\left(2x^{n},6y^{n}\right)&=0\\ \left(x^{n}-2\tau x^{n}\right)x^{n}+9\left(y^{n}-6\tau y^{n}\right)y^{n}&=0,\end{aligned}$$

that is

$$\tau=\frac{\left(x^{n}\right)^{2}+9\left(y^{n}\right)^{2}}{2\left(x^{n}\right)^{2}+54\left(y^{n}\right)^{2}}.$$

We implement this in Matlab code.

Listing 1. Gradient descent example

```
function gradient_descent
% Based on Kutz.
% Find the minimum of x^2 + 3y^2 using gradient descent

clear all %, clf

v = -3:0.2:3;
[X,Y] = meshgrid(v);
Z = X.^2 + 3*Y.^2;
subplot(2,1,1), surfc(X,Y,Z)

[px,py] = gradient(-Z,0.1,0.1);
subplot(2,1,2), contour(v,v,Z), grid on, hold on
quiver(v,v,px,py);                hold off

x(1) = 2; y(1) = 0.1; % initial guess
f(1) = x(1)^2 + 3*y(1)^2; % initial function value

err(1) = 1;
tol = 10^(-8);

for j=1:100

    tau=(x(j)^2 + 9*y(j)^2)/(2*x(j)^2 + 54*y(j)^2);
    x(j+1)=(1-2*tau)*x(j);  % update values
    y(j+1)=(1-6*tau)*y(j);
    f(j+1)=x(j+1)^2 + 3*y(j+1)^2;
    err(j+1) = abs(f(j+1)-f(j));

    if err(j+1)<tol  % check convergence
        break
    end

end
disp(['iter=',num2str(j+1),', x = ',num2str(x(end)),...
    ', y=',num2str(y(end)),', f=',num2str(f(end))])
hold on
```

```
comet(x,y)
```

4.4. **Convergence of the gradient descent.** First, consider an example $f(x,y) = \frac{1}{2}\left(x^2 + by^2\right)$ with $0 < b \leq 1$. The level sets of f are ellipses, very narrow along y direction when b is small. This creates big problems with convergence of the gradient descent. Next, we explain how.

Note that $\nabla f = (x, by)$, the minimum is $(0,0)$. If we start the iterations with $\left(x^0, y^0\right) = (b, 1)$, then we find that

$$x^k = b\left(\frac{b-1}{b+1}\right)^k, \quad y^k = \left(\frac{1-b}{1+b}\right)^k, \quad f^k = \left(\frac{1-b}{1+b}\right)^{2k} f_0. \tag{4.1}$$

Note that if $b = 1$, the convergence in 1 step. What if b is close to 0? When $b \ll 1$, then $r = \frac{1-b}{1+b} \sim 1 - 2b$ and $x^k \sim b(-1)^k(1-2kb)$, $y^k \sim 1 - 2kb$, $f^k \sim f_0(1-4kb)$. That is, it takes on the order of $1/2b$ iterations to get close to the minimum. The iterations zig-zag in the narrow valley along the y-axis. The convergence is linear as with every iteration, the distance to the minimum is reduced by the same factor r.

The exact solution above is found as follows. The step of the gradient descent solves the orthogonality condition

$$(x^n - \tau x^n)\, x^n + b^2\,(y^n - \tau b y^n)\, y^n = 0,$$

from which

$$\tau = \frac{(x^n)^2 + b^2 (y^n)^2}{(x^n)^2 + b^3 (y^n)^2}.$$

Then, after some algebra, we get

$$x^{n+1} = -\frac{b^2(1-b)(y^n)^2}{(x^n)^2 + b^3(y^n)^2} x^n, \quad y^{n+1} = \frac{(1-b)(x^n)^2}{(x^n)^2 + b^3(y^n)^2} y^n. \tag{4.2}$$

From here,

$$\frac{x^{n+1}}{y^{n+1}} = -b^2 \frac{y^n}{x^n}.$$

Let $z^n = y^n/x^n$, then

$$z^{n+1} = -\frac{1}{b^2}\frac{1}{z^n}.$$

For this equation, we find that $z^{2k+1} = -\frac{1}{b^2}\frac{1}{z^0}$ and $z^{2k} = z^0$. Therefore,

$$\frac{y^{2k}}{x^{2k}} = \frac{y^0}{x^0} = \frac{1}{b}, \quad \frac{y^{2k+1}}{x^{2k+1}} = -\frac{1}{b^2}\frac{x^0}{y^0} = -\frac{1}{b}.$$

That is

$$\frac{y^k}{x^k} = \frac{(-1)^k}{b}.$$

Substituting this into (4.2), we find

$$x^{n+1} = \frac{b-1}{b+1} x^n,$$

from which the results in (4.1) follow.

To understand the convergence of the gradient descent for a general $f(x)$, we assume that the Hessian H is strictly convex with eigenvalues λ in the interval $0 < m \leq \lambda \leq M$ for all x.

With the descent step $x^{k+1} = x^k - \tau\nabla f^k$, we can estimate that

$$f\left(x^{k+1}\right) \le f\left(x^k\right) + \left(\nabla f^k\right)^T\left(x^{k+1} - x^k\right) + \frac{1}{2}M\|x^{k+1} - x^k\|^2 =$$
$$= f\left(x^k\right) - \tau\|\nabla f^k\|^2 + \frac{1}{2}M\tau^2\|\nabla f^k\|^2.$$

The best τ minimizes the left side. The right side is minimized by $\tau = 1/M$. Then

$$f\left(x^{k+1}\right) \le f\left(x^k\right) - \frac{1}{2M}\|\nabla f^k\|^2.$$

Using the smallest bound for λ, we find for the minimum value $f\left(x^*\right)$

$$f\left(x^*\right) \ge f\left(x^k\right) - \frac{1}{2m}\|\nabla f^k\|^2.$$

From these two inequalities,

$$2m\left(f\left(x^k\right) - f\left(x^*\right)\right) \le \|\nabla f^k\|^2 \le 2M\left(f\left(x^k\right) - f\left(x^{k+1}\right)\right),$$

that is

$$f\left(x^{k+1}\right) - f\left(x^*\right) \le \left(1 - \frac{m}{M}\right)\left(f\left(x^k\right) - f\left(x^*\right)\right).$$

Which means that the error with every iterations drops linearly by a factor of $1 - m/M$, which is nearly 1 when the ratio m/M is small.

The convergence of the gradient descent is controlled by the condition number of the Hessian. Numerous approaches exist that try to overcome this difficulty.

4.5. **Newton's method .** If $f(x)$ has a minimum at x^*, then $\nabla f\left(x^*\right) = 0$. Start with a guess x^k and iterate to approach x^*. What is a step to take? Near x^k, Taylor expand the gradient

$$\nabla f\left(x^{k+1}\right) = \nabla f\left(x^k\right) + H\left(x^k\right)\left(x^{k+1} - x^k\right) + ...,$$

where H is the Hessian. Since we aim at $\nabla f\left(x^{k+1}\right) = 0$, choose x^{k+1} such that $\Delta x^k = x^{k+1} - x^k$ solves the linear system

$$H\left(x^k\right)\Delta x^k = -\nabla f\left(x^k\right).$$

Note that $x^{k+1} = x^k + \Delta x^k$ is a minimizer of the quadratic function

$$f\left(x^k\right) + \nabla f\left(x^k\right)^T\left(x - x^k\right) + \frac{1}{2}\left(x - x^k\right)^T H\left(x^k\right)\left(x - x^k\right).$$

Newton's method is second order

$$\|x^{k+1} - x^*\| \le C\|x^k - x^*\|^2.$$

For example, consider minimizing $f = \frac{1}{3}x^3 - 4x$. Then $\nabla f = x^2 - 4$ and $H = 2x$.

$$H\left(x^k\right)\Delta x^k = -\nabla f\left(x^k\right)$$
$$2x^k\left(x^{k+1} - x^k\right) = -\left(x^k\right)^2 + 4,$$

from which

$$x^{k+1} = \frac{1}{2}\left(x^k + \frac{4}{x^k}\right).$$

With $x^0 = 2.5$,we get $x^1 = 2.05$, $x^2 = 2.0006$, $x^3 = 2.000000009$, with every iteration the number of zeros after the decimal point increases rapidly. The error is

$$x^{k+1} - 2 = \frac{1}{2x^k}\left(x^k - 2\right)^2,$$

and as claimed, it decreases quadratically with iterations, provided we are close to the root.

Example 34. The same minimum can be found using **fminsearch** function, which uses the Nelder-Meade algorithm

Listing 2. Fminsearch example 2

```
function fminsearch_example
% uses Nelder-Meade method to find the minimum of f(x,y)
% e.g. for f=(x-1)^2 + a*y^2, we get c = [1 0]

clear all, clf

v = -2:0.2:2;
[X,Y] = meshgrid(v);
Z = (X-1).^2 + 3*Y.^2;
surfc(X,Y,Z)

% call the solver. funq is the function f(x,y) to be minimized,
% [1 2] are the initial guesses for the unknown x and y
% On return, c contains the solution, and
% fval the value of the objective function

[c fval] = fminsearch(@funq,[1 5])

    function f = funq(c)

        f = (c(1)-1).^2 + 3*c(2).^2;

    end
end
```

Example 35. The **fminsearch** can be also used to find the best fit with given data by minimizing the error function as in the following example

Listing 3. Fminsearch example

```
function fminsearch_example2
% uses Nelder-Meade method to find the minimum
% c = [14.6, 0.21, 63.0]

x = 1:24;   % raw data
y = [75 77 76 73 69 68 63 59 57 55 54 52 ...
   50 50 49 49 49 50 54 56 59 63 67 72];

plot(x,y,'ko')
grid on
```

```
% call the solver.
% datafit is the function to be minimized,
% [10 0 60] are the initial guesses for the unknown coefficients c = [A B C].
% On return, c contains the solution, and fval the value of the objective
% function
[c fval] = fminsearch(@datafit,[10 1 60])

xx=1:0.01:24;
yfit = c(1)*cos(c(2)*xx) + c(3); % this is the fitting function with the best coefficient

txtoptions = {'Interpreter','latex','FontSize',18};
plot(x,y,'ko',xx,yfit,'k-'); grid on
xlabel ('x',txtoptions{:})
ylabel ('y',txtoptions{:})
legend({'data','$$y = c_1 \cos(c_2 x) + c_3$$'},txtoptions{:},'Location','NorthEast');
title({'Data and their fit using fminsearch, c = ',num2str(c)},txtoptions{:})

    function e2 = datafit(c)

        e2 = sqrt(sum((c(1)*cos(c(2)*x)+c(3) - y).^2)/24);

    end
end
```

4.6. **Constrained optimization. Linear programming.** .... Often, one needs to find a minimum of $f(x)$ subject to some constraints. When both f and the constraints are linear functions of the unknown variables, the problem is called *linear program.* Its standard form is

$$\text{minimize } f(x) = c^T x$$
$$\text{subject to } Ax = b \text{ and } x \geq 0.$$

Other forms can be converted to this standard form by introducing extra variables (called *slack* variables). For example, if the constraint is $Ax \leq b$ instead of the equality, one introduces variables y such that $Ax + y = b$ and, in addition, $y \geq 0$. Then the constraints become $[A\,I]\begin{bmatrix} x \\ y \end{bmatrix} = b$ and $\begin{bmatrix} x \\ y \end{bmatrix} \geq 0$, which are in the standard form.

Example 36. Solve the following LP problem:

$$\begin{aligned} \text{minimize } & f(x) = -2x_1 - x_2 \\ \text{subject to } & x_1 + \frac{8}{3}x_2 \leq 4 \\ & x_1 + x_2 \leq 2 \\ & 2x_1 \leq 3 \\ & x_1 \geq 0 \\ & x_2 \geq 0. \end{aligned}$$

This can be solved using Matlab's **linprog** function. The **linprog** solves the problem given in the following form:

$$\begin{aligned} \text{minimize } & f(x) = c^T x \\ \text{subject to } & Ax \le b \\ & \bar{A}x = \bar{b} \\ & x_- \le x \le x_+ \end{aligned}$$

Its general format is, with the names matching with the above formulas:

$$[x, fval, exitflag] = linprog(c, A, b, Abar, bbar, xl, xu, x0, options),$$

where $x0$ is the initial guess. When any of these input variables are not required, one can use the empty matrix [] instead of that variable.

For the present example, the result is: $x_1 = 3/2$, $x_2 = 1/2$, and $f_{min} = -7/2$, obtained with $linprog(c, A, b)$, where

$$A = \begin{bmatrix} 1 & 8/3 \\ 1 & 1 \\ 2 & 0 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}, \quad b = \begin{bmatrix} 4 \\ 2 \\ 3 \\ 0 \\ 0 \end{bmatrix}, \quad c = \begin{bmatrix} -2 \\ -1 \end{bmatrix}.$$

Note that here we include the inequalities $x_1 \ge 0$ and $x_2 \ge 0$ into the matrix inequality, rather than as a constraint with xl and xu.

Listing 4. linprog example

```
function linprog_example
% use: [x,fval,exitflag]= linprog(c,A,b,Abar,bbar,xl,xu,x0,options)
clear all

c = [-2 -1]
A = [1 8/3; 1 1; 2 0; -1 0; 0 -1]
b = [4; 2; 3; 0; 0]
[x fval exitflag] = linprog(c,A,b)
```

4.7. **Genetic algorithm.** ....

Example 37. For the same data that were used in the least-squares fitting by **fminsearch**, we now find the best fit by using the genetic algorithm. The data are given by

x = 1:24

y = [75 77 76 73 69 68 63 59 57 55 54 52 50 50 49 49 49 50 54 56 59 63 67 72].

We need to fit these data with the function $y = A\cos(Bx) + C$. The previous fit was done by minimizing the square error

$$E = \sum (A\cos(Bx) + C - y)^2$$

by a minimization routine. It can, of course be also done using the normal equation.

Here, the idea is quite different. We choose a bunch of random samples of parameters A, B, C and calculate E for all of them. Then, we select a few of these samples for which the error was the smallest, and throw away the rest.

If we choose to sample $n = 50$ values for the set (A, B, C), then we get 50 values of E, and select only the best $n_2 = 10$, for which E is the smallest. This is the stage of the selection of the fittest. Say, for A it is A_1 with only 10 components.

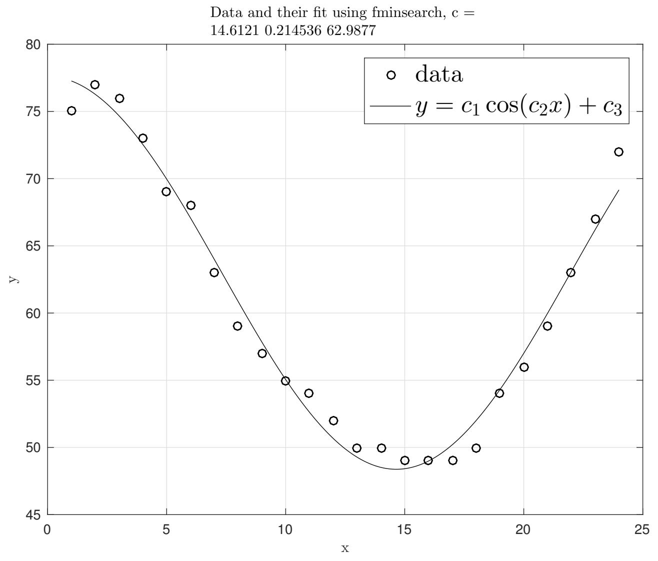


FIGURE 4.1. Fitting the data with fminsearch.

Next, we initiate some mutations. For each of A_1, B_1 and C_1, we create 4 more (vector) values by adding some random noise. For example,

$A_2 = A_1 + randn\,(n_2, 1)$, $A_3 = A_1 + randn\,(n_2, 1)$, ..., $A_5 = A_1 + randn\,(n_2, 1)$

will create 4 different mutants for A_1. Similarly, for B_1 and C_1.

Once these mutations are created, we form the next sample of 50 by combining the original with the 4 mutants:

$$A = \begin{bmatrix} A_1 & A_2 & A_3 & A_4 & A_5 \end{bmatrix}$$
$$B = \begin{bmatrix} B_1 & B_2 & B_3 & B_4 & B_5 \end{bmatrix}$$
$$C = \begin{bmatrix} C_1 & C_2 & C_3 & C_4 & C_5 \end{bmatrix}.$$

Then, the procedure is repeated. Except, that with the next generation of mutants, we reduce the strength of the mutations by adding $randn\,(n, 1)\,/jgen$, where $jgen$ is the generation number. This should help convergence if such is the case.

Note that one should start with a reasonably good guess for the parameters, otherwise, it may take forever to converge. For example, in this particular case, a reasonable guess for C is the average value of y, which is about 60, as seen in Fig. 4.1. For A, it is about 15, as that is the magnitude of the spread in y about the average. And for B, a reasonable guess would be such that $15B = \pi$, as 15 is about half the period of the function.

LISTING 5. Genetic algorithm example

```
function genetic_example
%Based on Kutz
% fminsearch gives c = [14.6, 0.21, 63.0]
clear all

x = 1:24;
y = [75 77 76 73 69 68 63 59 57 55 54 52 50 ...
     50 49 49 49 50 54 56 59 63 67 72];
```

```
subplot(2,1,1), plot(x,y,'ko')
grid on

%pause
xx=1:0.01:24;

m = 400; % number of generations, try also 2000
n = 100; % number of trials,                try 200
n2= n/5; % number of trials to be kept,   this must be n/5, try 40
A = 10+randn(n,1);      %12
B =  1+randn(n,1);      %pi/12
C = 60+randn(n,1);      %60

 for jgen=1:m

     for j=1:n  % evaluate objective function
         E(j)= sum((A(j)*cos(B(j)*x) + C(j) - y).^2);
     end

     [Es,Ej]=sort(E);  % sort from small to large

     Ak1=A(Ej(1:n2)); % best 10 solutions
     Bk1=B(Ej(1:n2));
     Ck1=C(Ej(1:n2));

     scale = jgen; %sqrt(jgen);

     Ak2=Ak1+randn(n2,1)/scale; % 10 new mutations
     Bk2=Bk1+randn(n2,1)/scale;
     Ck2=Ck1+randn(n2,1)/scale;

     Ak3=Ak1+randn(n2,1)/scale; % 10 new mutations
     Bk3=Bk1+randn(n2,1)/scale;
     Ck3=Ck1+randn(n2,1)/scale;

     Ak4=Ak1+randn(n2,1)/scale; % 10 new mutations
     Bk4=Bk1+randn(n2,1)/scale;
     Ck4=Ck1+randn(n2,1)/scale;

     Ak5=Ak1+randn(n2,1)/scale; % 10 new mutations
     Bk5=Bk1+randn(n2,1)/scale;
     Ck5=Ck1+randn(n2,1)/scale;

     A=[Ak1; Ak2; Ak3; Ak4; Ak5]; % group new 50
     B=[Bk1; Bk2; Bk3; Bk4; Bk5];
     C=[Ck1; Ck2; Ck3; Ck4; Ck5];
```

```
        yfit = A(1)*cos(B(1)*xx) + C(1);
        subplot(2,1,1), plot(x,y,'ko',xx,yfit,'k-'); grid on
        subplot(2,1,2), semilogy(abs([A,B,C]),'o');  drawnow
        disp(['A=',num2str(A(1)),', B=',num2str(B(1)),', C=',num2str(C(1))])
        %pause

  end

end
```

Listing 6. Built-in genetic algorithm example

```
function ga_example
clear all

% some data with noise
n = 30;
xx = linspace(0,2*pi,n);
yy = 3*cos(2*xx) + 1 + 0.5*rand(1,n);

[x,fval, flag] = ga(@(x)fit_line(x),3,[],[],[],[],[],[])
%x = ga('fit',n,A,b,Abar,bbar,xl,xu,nonlin,options)

yfit = x(1)*cos(x(2)*xx) + x(3);
plot(xx,yy,'ro',xx,yfit,'b-x'); grid on

function E = fit_line(x)

     E = sum((x(1)*cos(x(2)*xx) + x(3) - yy).^2);

end
end
```

4.8. **Convexity.** .... For a unique minimum to exist, function f must be convex.

Definition 38. A function f is called convex if it satisfies

$$f\left(px+(1-p)\,y\right)\le pf\left(x\right)+(1-p)\,f\left(y\right)$$

for any x and y and $p\in(0,1)$.

That is, a convex function between x and y lies below a straight line connecting points x and y.
Examples of convex functions:

$$f_1=ax+b,\quad f_2=x^2,\quad f_3=\max\left(f_1,f_2\right).$$

The maximum of one or more convex functions is always convex. Indeed, to verify this, let $f=\max\left(f_1,f_2,\ldots\right)$ and $z=px+(1-p)\,y$, then, since each f_i is convex and less than f, we get

$$f_i\left(z\right)\le pf_i\left(x\right)+(1-p)\,f_i\left(y\right)\le pf\left(x\right)+(1-p)\,f\left(y\right).$$

Therefore, the maximum of the left-hand side will also be less than the right-hand side:

$$f\left(z\right)\le pf\left(x\right)+(1-p)\,f\left(y\right),$$

which means exactly the convexity of f.

In particular, if we have a lot of linear functions, then their maximum will be a convex function. A given convex function will *equal* to the maximum of *all* the linear functions below it, i.e., the maximum of all of its tangents.

- ◇ A *convex set* K is a set such that if $x, y \in K$, then $px + (1-p)\,y \in K$.
- ◇ A convex set is the intersection of all half-spaces that contain it.
- ◇ The *intersection* of any family of convex sets is a convex set. Generally, not true about *unions.*

Some useful facts about matrices following from *pos.def. + pos.def. = pos.def.* :

- ◇ The set of positive definite $n \times n$ matrices is convex.
- ◇ The set of positive semi-definite $n \times n$ matrices is also convex.

Some basic examples:

- ◇ $f = c^T x$ is convex, $\nabla f = c$, $H = \partial^2_{ij} f = 0$ - semi-definite.
- ◇ $f = \frac{1}{2} x^T S x$ with s.p.d. S is convex, $\nabla f = Sx$, $H = \partial^2_{ij} f = S$.
- ◇ Norms $f(x) = \|x\|$ are convex functions of x as they must satisfy the triangle inequality

$$\|px + (1-p)\,y\| \leq p\|x\| + (1-p)\,\|y\|,$$

which is exactly the convexity condition. The unit ball $\|x\| \leq 1$ will then be a convex set.

4.9. **Lagrange multipliers . .** We are asked to minimize a function subject to some constraints.

Example 39. Minimize $f = x_1^2 + x_2^2$ on the line $K : a_1 x_1 + a_2 x_2 = b$.

We need the point on K that is nearest to $(0,0)$, the minimum of f. Since $f = const$ are circles and K is a line, then we are looking to find the tangency condition.

The method of Lagrange multipliers considers a new function L that involves both f and K and looks for its minimum.

$$L(x, \lambda) = f(x_1, x_2) + \lambda (a_1 x_1 + a_2 x_2 - b),$$

where λ is the Lagrange multiplier.

Then

$$\frac{\partial L}{\partial x_1} = 2x_1 + \lambda a_1 = 0$$
$$\frac{\partial L}{\partial x_2} = 2x_2 + \lambda a_2 = 0$$
$$\frac{\partial L}{\partial \lambda} = a_1 x_1 + a_2 x_2 - b = 0.$$

The last equation is just the constraint.

From the first equations

$$x_1 = -\frac{a_1}{2}\lambda, \quad x_2 = -\frac{a_2}{2}\lambda.$$

Putting these into the last equation, we find

$$\lambda = \frac{-2b}{a_1^2 + a_2^2}$$

and hence

$$x_1 = \frac{a_1 b}{a_1^2 + a_2^2}, \quad x_1 = \frac{a_2 b}{a_1^2 + a_2^2}.$$

The value of the function at the tangency point is

$$f_{min} = \frac{b^2}{a_1^2 + a_2^2}.$$

And the interesting thing here is that the Lagrange multiplier is the derivative of the minimum cost f with respect to the constraint level b:

$$\lambda = -\frac{\partial f_{min}}{\partial b}.$$

That is, the *Lagrange multiplier is a measure of the sensitivity of the cost to the changes in the level of constraint.*

More generally, suppose we need to minimize

$$f = \frac{1}{2} x^T S x \text{ subject to } A^T x = b$$

with m constraints on n variables x. Then the Lagrange function is

$$L = \frac{1}{2} x^T S x + \lambda^T \left(A^T x - b\right)$$

with m Lagrange variables λ. The minimum equations become

$$Sx + A\lambda = 0 \tag{4.3}$$

$$A^T x = b. \tag{4.4}$$

We can solve this system by first finding $x = -S^{-1} A \lambda$, then from the second equation $\lambda = -\left(A^T S^{-1} A\right)^{-1} b$. Thus the solution is

$$\lambda^* = -\left(A^T S^{-1} A\right)^{-1} b, \quad x^* = S^{-1} A \left(A^T S^{-1} A\right)^{-1} b.$$

The minimum cost is then

$$f^* = \frac{1}{2} \left(x^*\right)^T S x^* = \frac{1}{2} b^T \left(A^T S^{-1} A\right)^{-1} b.$$

And the gradient of the cost with respect of the constraint level is the same Lagrange multiplier (with a minus):

$$\frac{\partial f^*}{\partial b} = \left(A^T S^{-1} A\right)^{-1} b = -\lambda^*.$$

The system (4.3-4.4) can be written as

$$\begin{bmatrix} S & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}.$$

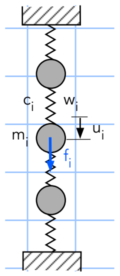
Applying elimination this becomes

$$\begin{bmatrix} S & A \\ 0 & -A^T S^{-1} A \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix},$$

where the bottom right block is called the Schur complement. Now we see that, since S is positive definite and the Schur complement is negative definite, our matrix in the system is indefinite. There is a saddle point in the Lagrangian $L(x, \lambda)$, meaning that it has a minimum in x, but a maximum in λ. Convex in x and concave in λ.

4.10. **Masses and springs ..** We begin with a basic example of equilibrium in a chain of masses and springs.

Example 40. A vertical chain of springs and masses is in equilibrium. The ends of the chain are fixed. The masses are m_i, the spring constants are c_i, and the forces acting on each mass are f_i (say, gravity). The problem is to find the equilibrium state.



Suppose forces in the springs are w_i, and u_i are the displacements of each mass from its equilibrium position when no forces are acting and springs are not compressed or stretched. We also need the spring elongations $e_i = u_i - u_{i-1}$.

Direct approach. For simplicity, consider 3 masses and 4 springs as in the figure. Generalization will be straightforward.

Notice the following connection between the displacement u and the force f.

$$u \xrightarrow{1} e = Au \xrightarrow{2} w = Ce \xrightarrow{3} f = A^T w.$$

We next explain this connection and identify the matrices A and C.

(1) First, e is the elongation vector of the springs:

$$\begin{aligned} e_1 &= u_1 - u_0 = u_1 \\ e_2 &= u_2 - u_1 \\ e_3 &= u_3 - u_2 \\ e_4 &= u_4 - u_3 = -u_3. \end{aligned}$$

Therefore

$$e = Au, \quad A = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix}.$$

Notice, because of the fixed-fixed boundary conditions, $u_0 = u_4 = 0$.

(2) Next comes the connection between the elongation e and the forces in the springs, $w = Ce$. This is Hooke's law: in each spring, the elastic force is proportional to the spring elongation, $w_i = c_i e_i$. Therefore,

$$C = \begin{bmatrix} c_1 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 \\ 0 & 0 & c_3 & 0 \\ 0 & 0 & 0 & c_4 \end{bmatrix}.$$

(3) Finally, the forces on each mass are balanced in equilibrium: $f_i = w_i - w_{i+1}$. This is written as

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}. \quad f = A^T w.$$

The reason that the matrix here is A^T is that, for elongation, A comes from derivative d/dx of the displacement u – the strain is the gradient of displacement. For the force balance, we have the equilibrium equation $\nabla\cdot\sigma = f$, the balance of divergence of stress and body forces. The divergence $\nabla\cdot$ is the transpose of the gradient, ∇, hence A^T in the force balance.

Combining these equations, we obtain

$$f = A^T C A u = K u,$$

which is solved simply as

$$u = K^{-1} f = \left(A^T C A\right)^{-1} f. \tag{4.5}$$

Note that the stiffness matrix K can be shown to be

$$K = \begin{bmatrix} c_1 + c_2 & -c_2 & 0 \\ -c_2 & c_2 + c_3 & -c_3 \\ 0 & -c_3 & c_3 + c_4 \end{bmatrix}.$$

Note also that $A^T C A$ is the discrete brother of $\nabla\cdot c(x)\nabla \equiv \operatorname{div}(c(x)\operatorname{grad})$ operator in differential equations. We will see this again later in the course.

Special cases. Until now we looked at the *fixed-fixed* boundary conditions, the ends of the chain were fixed.

Now consider two other possibilities, fixed-free and free-free.

◇ *Fixed-free.* The top end is still fixed, but now $c_4 = 0$. Then

$$K = \begin{bmatrix} c_1 + c_2 & -c_2 & 0 \\ -c_2 & c_2 + c_3 & -c_3 \\ 0 & -c_3 & c_3 \end{bmatrix}$$

is still a positive definite matrix and the solution is still the same $u = K^{-1} f$.

◇ *Free-free.* Now we cut both the top and bottom ends: $c_1 = 0$, $c_4 = 0$. Then

$$K = \begin{bmatrix} c_2 & -c_2 & 0 \\ -c_2 & c_2 + c_3 & -c_3 \\ 0 & -c_3 & c_3 \end{bmatrix}$$

and this matrix is now singular – add the rows (or columns) and you will get a zero row. The question is: Does the system $Ku = f$ still have solutions? The answer is: it depends on f. For the system to have solutions, f must be in the column space of K, $f \in C(K)$. That requires f to be orthogonal to the left nullspace of K: $f \perp N\left(K^T\right)$. The left nullspace (and $N(K)$ because of symmetry of K) is spanned by $u_0 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$, therefore the solvability condition is that

$$f^T u_0 = 0, \rightarrow f_1 + f_2 + f_3 = 0.$$

In this case, there are infinitely many solutions, since to any solution u we can add ku_0 with any constant k: $K(u + ku_0) = Ku + kKu_0 = f + 0$.

Physically, this means that the mass-spring chain can be in equilibrium provided the sum of external forces is zero. The nullspace part of the solution means that the equilibrium can be retained if we shift all the masses by equal amounts in the same direction.

Energy approach. The same solution can be obtained via energy considerations. The idea is to minimize the energy of the springs subject to the constraints imposed by the forces on the masses.

Let

$$E(w) = \sum_{i=1}^{4} E_i(w_i) = \sum_{i=1}^{4} \frac{c_i e_i^2}{2} = \sum_{i=1}^{4} \frac{w_i^2}{2c_i} = \frac{1}{2} w^T C^{-1} w$$

be the total energy of the springs, where we assume that springs obey the Hooke's law:

$$w_i = c_i e_i.$$

We need to minimize $E(w)$ subject to the force-balance constraint, $f = A^T w$. For this purpose, define the Lagrange function

$$L = E(w) + \lambda \left(A^T w - f\right).$$

Then

$$\frac{\partial L}{\partial w} = \frac{\partial E}{\partial w} + A\lambda = C^{-1} w + A\lambda = 0 \tag{4.6}$$

$$\frac{\partial L}{\partial \lambda} = A^T w - f = 0. \tag{4.7}$$

From here, we could write

$$w = -CA\lambda, \quad -A^T CA\lambda - f = 0,$$

therefore

$$\lambda = -\left(A^T CA\right)^{-1} f \text{ and } w = CA\left(A^T CA\right)^{-1} f.$$

Comparing this with (4.5), we see that the Lagrange multiplier is simply the displacement of the masses (with a minus sign): $\lambda = -u$.

Now we calculate the value of the minimum energy:

$$\begin{aligned} E_{min} &= \frac{1}{2} w^T C^{-1} w = \\ &\frac{1}{2} f^T \left(A^T CA\right)^{-1} A^T C^T C^{-1} CA \left(A^T CA\right)^{-1} f \\ &= \frac{1}{2} f^T \left(A^T CA\right)^{-T} \underbrace{A^T \overbrace{C^T C^{-1}}^{I} CA}_{A^T CA} \left(A^T CA\right)^{-1} f \\ &\text{(using symmetry of } K = A^T CA \text{ and that } C^T = C) \\ &= \frac{1}{2} f^T \left(A^T CA\right)^{-1} f = \frac{1}{2} f^T K^{-1} f. \end{aligned}$$

That us

$$E_{min} = \frac{1}{2} f^T K^{-1} f.$$

Therefore,

$$\frac{\partial E_{min}}{\partial f} = K^{-1} f = -\lambda = u \implies$$

the displacement u is the sensitivity of the minimum energy with respect to the applied forces:

$$\delta E_{min} = u \cdot \delta f,$$

small change in the applied forces leads to a small change in the minimum energy, and the displacement is a measure of how sensitive the energy is to such changes. For example, if u is very small, to change the minimum energy one must apply relatively large forces to the masses.

Remark. Note that the same solution can be obtained by minimizing, without any constraints, the total potential energy

$$P = \frac{1}{2}u^T K u - u^T f$$

that includes both the spring energy and the energy in the field of gravity. Then $dP/du = 0$ gives immediately $u = K^{-1}f$ and $P_{min} = -\frac{1}{2}f^T K f = -\frac{1}{2}f^T u$.

Saddle point nature of the problem. If we do not eliminate λ and then solve for w, but instead decide to solve for both by rewriting the system (4.6-4.7) as a block system

$$\begin{bmatrix} C^{-1} & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} w \\ \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ f \end{bmatrix}, \tag{4.8}$$

then we obtain, as above, a saddle-point problem with a matrix

$$S = \begin{bmatrix} C^{-1} & A \\ A^T & 0 \end{bmatrix}$$

that is not positive definite, but rather indefinite. Using elimination this matrix converts to

$$S' = \begin{bmatrix} C^{-1} & A \\ 0 & -A^T C A \end{bmatrix}$$

with positive definite C^{-1} and negative definite $-A^T CA$. Solving the linear system (4.8) is equivalent to finding a saddle point of a quadratic form $q = \frac{1}{2}z^T S' z - b^T z$, where $b = \begin{bmatrix} 0 \\ f \end{bmatrix}$. Such a quadratic form would have both positive and negative terms, such as: $a_1 z_1^2 + a_2 z_2^2 - d_1 z_3^2$, so that it has a minimum over z_1, z_2 when z_3 is fixed, and a maximum over z_3 when z_1, z_2 are fixed.

For example, for two springs and one mass with $m = 1$ and $c_1 = c_2 = 1$, we find that

$$L = \frac{1}{2}w_1^2 + \frac{1}{2}w_2^2 - uw_1 + uw_2 + uf$$

and

$$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 1 \\ -1 & 1 & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ u \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ f \end{bmatrix}.$$

This matrix S can be factored as $S = LDL^T$ with

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 1 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{bmatrix}$$

and the quadratic form becomes

$$q = \frac{1}{2}\left[(w_1 - u)^2 + (w_2 + u)^2 - 2u^2\right].$$

Stokes problem. In fluid mechanics, the Stokes flow corresponds to slow motion of a viscous fluid in which viscous forces and pressure balance each other and dominate over inertial effects. The equations governing Stokes flow are one of incompressibility div$v = 0$ and momentum balance $-\nabla p + \Delta v + f = 0$, where v is fluid velocity, p is pressure, f is the body force and we have taken the density and viscosity as unity. These two equations can be written as a system

$$\begin{bmatrix} -\Delta & \nabla \\ \nabla^T & 0 \end{bmatrix} \begin{bmatrix} v \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix},$$

where the incompressibility condition is written as $\nabla^T v = 0$. The saddle point nature of the problem is evident. Pressure p is a Lagrange multiplier in this case.