DD2363/2023 – Lecture 11
Optimization (ch.15)

Johan Hoffman

Finite element method

The finite element method (FEM) is a generalization of L^2 projection to a solution method for differential equations. Hence, we need to use a Hilbert space V where not only functions but also a suitable representation of their derivatives belongs to $L^2(I)$, referred to as a *Sobolev space*.

A linear partial differential equation with solution $u \in V$ can be expressed in weak form as

$$a(u, v) = L(v), \quad \forall v \in V, \tag{9.24}$$

where $a(\cdot, \cdot) : V \times V \to R$ is a bilinear form, $L(\cdot) : V \to R$ is a linear form, and the Sobolev space V is chosen such that the bilinear and linear forms are well defined, that is,

$$|a(u, v)| < \infty, \quad |L(v)| < \infty.$$

Finite element method

We also need to prescribe boundary conditions for the endpoints of the interval I, which may either take the form of a *Dirichlet boundary condition* which is incorporated into the Sobolev space V, or a *Neumann boundary condition* that is part of the bilinear and linear forms.

In a finite element method we seek an approximate solution to the differential equation (9.24), of the form

$$U(x) = \sum_{j=1}^{N} U_j \phi_j(x).$$

Here $\{\phi_j(x)\}_{j=1}^N$ is a set of finite element basis functions that span a finite element approximation space $V_N \subset V$, and $\{U_j\}_{j=1}^N$ are the coordinates of $U \in V_N$ in that basis.

Finite element method

The approximation can be determined from the variational formulation

$$a(U, v) = L(v), \quad \forall v \in V_N,$$

which corresponds to a system of linear equations

$$A\alpha = b,$$

where $\alpha = (U_j)$ is the vector of N coordinates which determine the approximation by equation (9.25), $A = (a_{ij})$ is an $N \times N$ matrix, and $b = (b_i)$ an N vector, with

$$a_{ij} = a(\phi_j, \phi_i), \quad b_i = L(\phi_i).$$

The Poisson equation

Example 9.12. Consider the Poisson equation

$$-u''(x) = f(x), \quad x \in (0,1),$$

with the two Dirichlet boundary conditions $u(0) = 0$ and $u(1) = 0$. If we multiply both sides of the equation by a test function v and integrate over the interval $[0,1]$, we get

$$-\int_0^1 u''(x)v(x)\,dx = \int_0^1 f(x)v(x)\,dx.$$

By partial integration of the left hand side of the equation, using the boundary conditions, we are lead to the weak form of the equation: find $u \in V$ such that

$$a(u,v) = L(v), \quad v \in V,$$

with a Sobolev space V for which the boundary conditions are satisfied.

The Poisson equation

By partial integration of the left hand side of the equation, using the boundary conditions, we are lead to the weak form of the equation: find $u \in V$ such that

$$a(u, v) = L(v), \quad v \in V,$$

with a Sobolev space V for which the boundary conditions are satisfied. The bilinear and linear forms are defined by

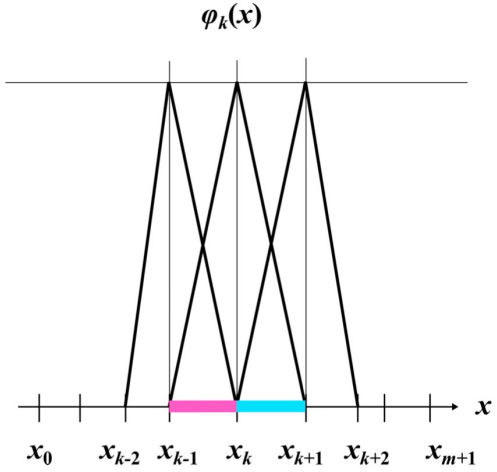
$$a(u, v) = \int_0^1 u'(x)v'(x)\,dx, \quad L(v) = \int_0^1 f(x)v(x)\,dx,$$

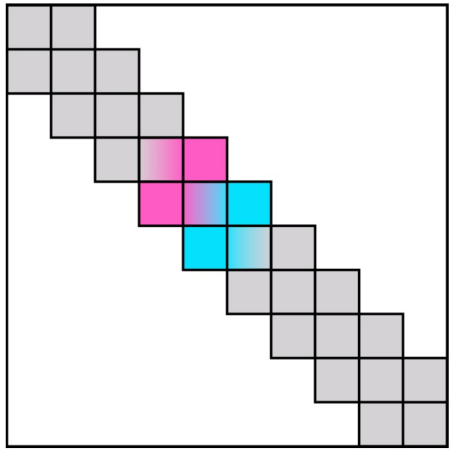
and in the FEM approximation (9.25) the degrees of freedom represent the internal nodes of the mesh, since the endpoint degrees of freedom are determined by the Dirichlet boundary condition.

FEM assembly algorithm

ALGORITHM 9.2. (A, b) = assemble_system(f).
Input: function **f**
Output: assembled matrix **A** and vector **b**.

```
1:  for k=0:no_elements-1 do
2:      q = get_no_local_shape_functions(k)
3:      loc2glob = get_local_to_global_map(k)
4:      for i=0:q do
5:          b[i] = integrate_vector(f, k, i)
6:          for j=0:q do
7:              a[i,j] = integrate_matrix(k, i, j)
8:          end for
9:      end for
10:     add_to_global_vector(b, loc2glob)
11:     add_to_global_matrix(a, loc2glob)
12: end for
13: return A, b
```





Partial differential equation

A *partial differential equation* (PDE) is a differential equation which expresses relations between partial derivatives of a function of several variables. If we move all terms to one side of the equation, we can state any PDE on residual form as

$$R(u(x,t)) = 0,$$

which describes the evolution of a function

$$u : \Omega \times I \to R^M,$$

with $\Omega \subset R^d$ a spatial domain and $I = [0, T]$ a time interval. If $M = 1$, the solution to the PDE is a scalar valued function, otherwise it is vector valued.

Semi-discretization

Here we first discretize the spatial domain Ω, e.g. by a structured grid or an unstructured mesh, to obtain an approximation

$$u(x,t) \approx \sum_{i=1}^{N} U_i(t)\phi_i(x),$$

with $\{\phi\}_{i=1}^{N}$ a set of scalar or vector valued basis functions. The evolution of the degrees of freedom $\{U_i(t)\}_{i=1}^{N}$ can then be described by a system of NM initial values problems, assuming that we use N degrees of freedom for each vector component of the approximation. This system of initial value problems is then solved by a time stepping method.

Finite difference method

Example 14.19 (Finite difference method). Analogous to Example 5.9, consider now the time dependent convection-diffusion partial differential equation

$$\dot{u}(x,t) + u'(x,t) - \epsilon u''(x,t) = f(x), \quad u(x,0) = 0, \quad (x,t) \in \Omega \times I,$$

with a diffusion coefficient $\epsilon > 0$, spatial interval $\Omega = [0,1]$, and suitable conditions on $u(x,t)$ at the boundary points $x = 0$ and $x = 1$. With a finite difference method we can approximate the differential equation by the matrix equation

$$\dot{U}(t) + (D_h^-)U(t) - \epsilon(D_h^+ D_h^-)U(t) = b, \quad U(t) = (U_i(t)), \quad b = (b_i),$$

with $b_i = f(x_i)$, and $U_i(t) = U(x_i,t)$ the values at the nodes x_i of the grid.

Finite element method

Example 14.20 (Finite element method). Now we instead approximate the partial differential equation in Example 14.19 by a finite element method, based on the following variational formulation: for each $t \in I$ find $U(t) \in V_N$, such that

$$(\dot{U}(t), v) + a(U(t), v) = L(v), \quad \forall v \in V_N,$$

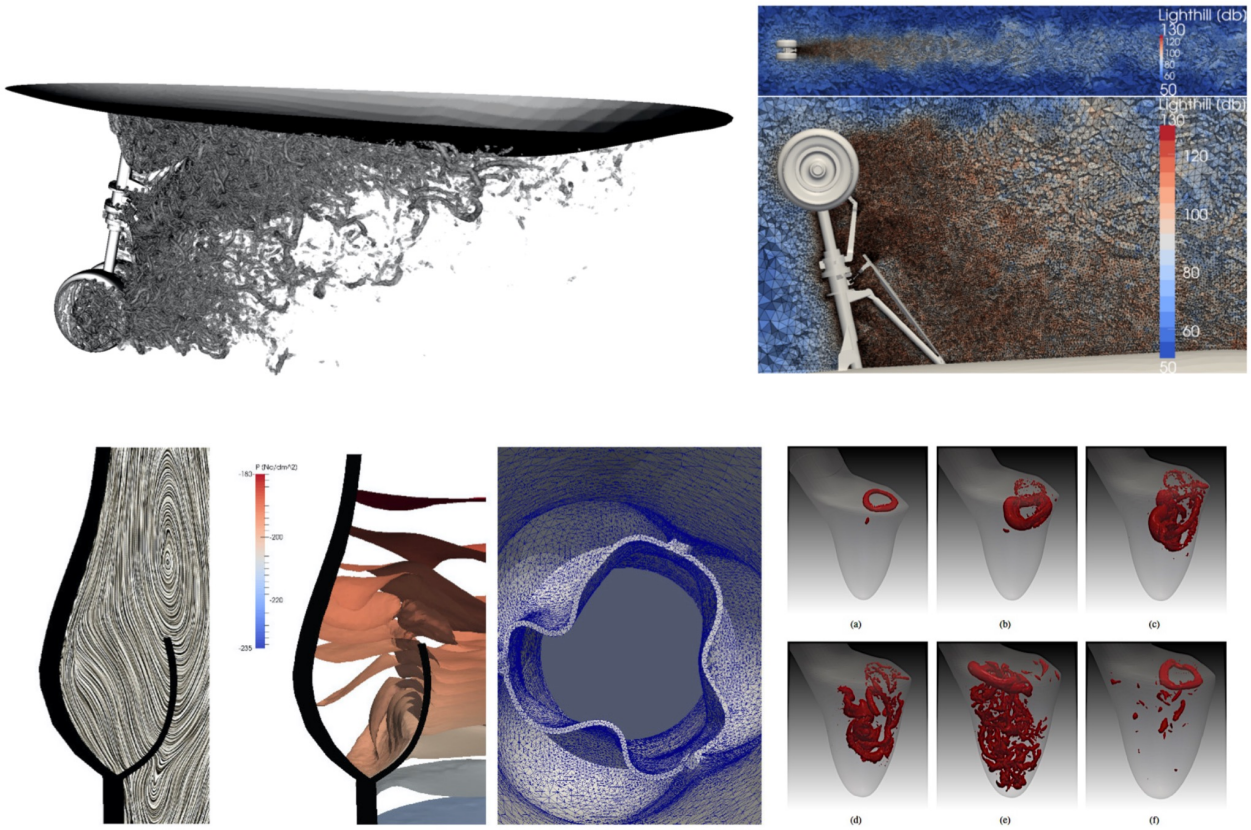
with V_N a piecewise polynomial approximation space defined over an unstructured mesh, and with $a(\cdot, \cdot)$ a bilinear form and $L(\cdot)$ a linear form. This is equivalent to a matrix equation for the vector $U(t) = (U_j(t))$, the degrees of freedom in the piecewise polynomial basis,

$$M\dot{U}(t) + AU(t) = b,$$

with M a mass matrix, A the matrix which corresponds to the bilinear form $a(\cdot, \cdot)$, and b vector generated by the linear form $L(\cdot)$. We can express the matrix equation on the form (14.1), as

$$\dot{U}(t) = f(U(t), t) = M^{-1}(b - AU(t)).$$

Finite element method



Convex minimization

The minimization problem in R^n takes the form: find $x^* \in D$, such that

$$f(x^*) \leq f(x), \quad \forall x \in D, \tag{15.1}$$

with $D \subset R^n$ the *search space*, $x^* \in D$ the *optimal solution*, and $f : D \to R$ the *objective function* (or *cost function*). We say that the minimization problem is *unconstrained* if $D = R^n$. The point $\hat{x} \in D$ is a *local minimum* if there exists an $\epsilon > 0$, such that

$$f(\hat{x}) \leq f(x), \quad \forall x : \|x - \hat{x}\| < \epsilon.$$

Convex minimization

We say that a minimization problem is *convex*, if for all $x, y \in D$ and $t \in (0, 1)$, the search space is convex,

$$(1 - t)x + ty \in D,$$

and the objective function is convex,

$$(1 - t)f(x) + tf(y) \leq f((1 - t)x + ty).$$

If the minimization problem is convex then a local minimum is also a global minimum, and if the problem is strictly convex, with strict inequality, the global minimum is unique.

Convex minimization

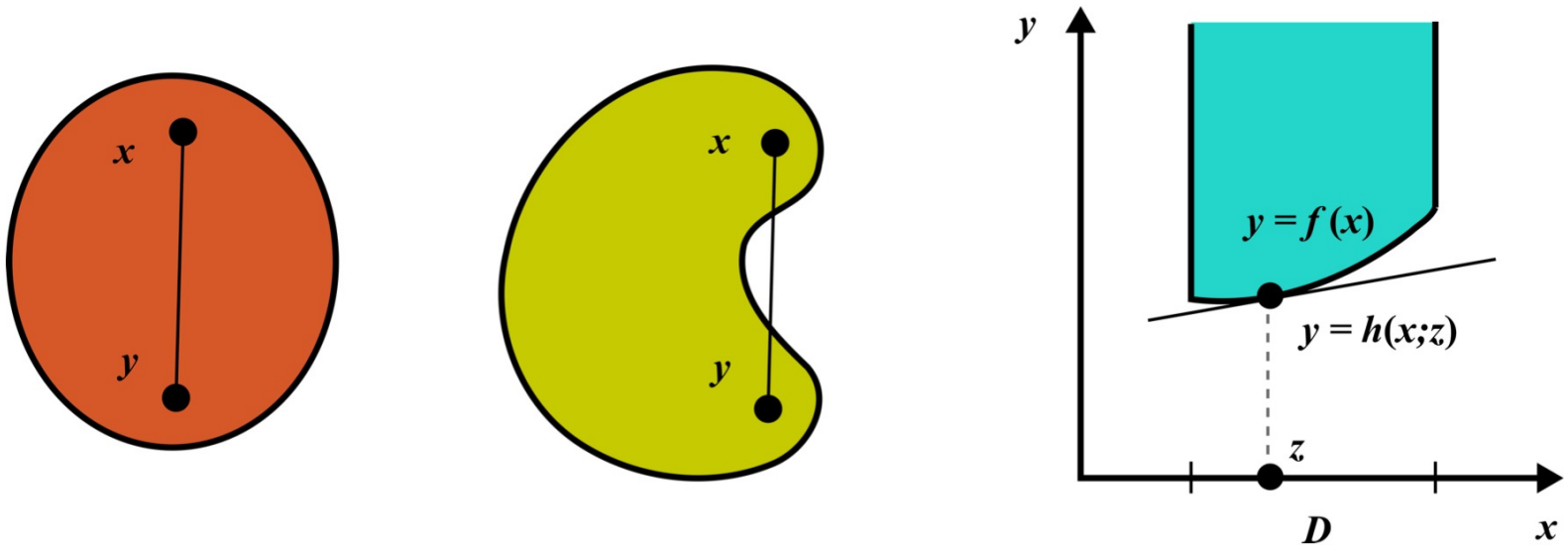


Figure 15.1. *A convex set (left) and a non-convex set (center). For a convex function* $f : D \to R$ *(right), we plot the convex set* $\{y \in R : y \geq f(x), \forall x \in D\}$ *and the tangent hyperplane* $y = h(x; z)$.

Convex minimization

For a convex function $f \in \mathcal{C}^1(D)$ the set $\{y \in R : y \geq f(x), \forall x \in D\}$ is convex. This is equivalent to the condition that the function $f(x)$ must be greater than or equal to the tangent hyperplane $y = h(x; z)$ at any $z \in D$, in other words,

$$f(x) \geq h(x; z) = f(z) + \nabla f(z)^T (x - z), \quad \forall z \in D. \tag{15.2}$$

Example 15.1. The function $f(x) = x^2$ is convex, since

$$0 \leq (x - z)^2 = x^2 - 2xz + z^2 \Leftrightarrow x^2 \geq 2xz - z^2 = z^2 + 2z(x - z) = f(z) + f'(z)(x - z).$$

Convex minimization

To further characterize convexity we use Taylor's formula, which states that for $f \in C^2(D)$,

$$f(x) = f(z) + \nabla f(z)^T(x - z) + \frac{1}{2}(x - z)^T Hf(\eta)(x - z), \quad \forall x, z \in D, \tag{15.3}$$

where $\eta \in D$ lies on the line segment between x and z which by convexity lies inside D. The *Hessian matrix* $Hf(z) = (Hf_{ij}(z))$ is defined by

$$Hf_{ij}(z) = \frac{\partial^2 f}{\partial x_i \partial x_j}(z), \quad i, j = 1, ..., n.$$

Hence, by equation (15.2), for a convex function $f \in C^2(D)$ convexity is equivalent to the Hessian $Hf(\eta)$ being positive semi-definite for all $\eta \in D$. If the function $f(x)$ is convex, we say that the function $-f(x)$ is *concave*.

Example 15.2. The function $f(x) = x^2$ is convex, since $f''(x) = 2 \geq 0$. So are all linear and constant functions $f(x)$, since $f''(x) = 0$, and the exponential function $f(x) = \exp(x)$ because $f''(x) = f(x) = \exp(x) \geq 0$ for all $x \in R$.

Gradient descent minimization

The *gradient descent method*, or *steepest descent method*, is an iterative search method that finds a critical point of an objective function $f(x)$, by searching for the next iterate in the direction opposite the gradient of the objective function. The motivation follows from Taylor's formula,

$$f(x + \Delta x) - f(x) = \nabla f(x)^T \Delta x + \mathcal{O}(\|\Delta x\|^2),$$

which shows that for small increments $\Delta x \in R^n$, the sharpest decrease in the objective function results from aligning Δx with $-\nabla f(x)$, to maximize the inner product $\nabla f(x)^T \Delta x$.

Gradient descent minimization

The *step length* $\alpha^{(k)}$ can be chosen by an iterative *line search* method, to minimize the objective function in the direction of the negative gradient, an optimization problem in itself. Or we can seek to satisfy the condition

$$f(x^{(k)} - \alpha^{(k)} \nabla f(x^{(k)})) \leq \beta f(x^{(k)}),$$

with $\beta < 1$ a parameter of the method. Alternatively, a step length is chosen based on how close to a local minimum $x^{(k)}$ is, characterized by the size of the gradient $\nabla f(x^{(k)})$. As a stopping criterion for the gradient descent method we can use the condition that $\|\nabla f(x^{(k)})\| < \text{TOL}$.

Gradient descent minimization

ALGORITHM 15.1. x = gradient_descent_method(f, x0).
Input: objective function **f**, and initial guess **x0**.
Output: solution vector **x**.

```
1: x[:] = x0[:]
2: Df = compute_gradient(f, x)
3: while norm(Df) > TOL do
4:     Df = compute_gradient(f, x)
5:     alpha = get_step_length(f, Df, x)
6:     x[:] = x[:] - alpha*Df
7: end while
8: return x
```

Gradient descent minimization

Example 15.3. An important case is when the objective function is the quadratic form

$$f(x) = \frac{1}{2}x^T Ax - x^T b + c, \tag{15.4}$$

where A is a symmetric positive definite $n \times n$ matrix, b an n vector, and c a real number. Since the matrix A is symmetric positive definite, the quadratic form is strictly convex. Therefore, the quadratic form has a unique finite global minimum, the critical point of $f(x)$,

$$0 = \nabla f(x) = \nabla(\frac{1}{2}x^T Ax - x^T b + c) = \frac{1}{2}Ax + \frac{1}{2}A^T x - b = Ax - b.$$

Gradient descent minimization

That is, the solution to the minimization problem is the solution to the system of linear equations $Ax = b$. To prove the converse, that $x = A^{-1}b$ is the solution to the minimization problem, add a perturbation $y \in R^n$ to x, such that

$$
\begin{aligned}
f(x+y) &= \frac{1}{2}(x+y)^T A(x+y) - (x+y)^T b + c \\
&= \frac{1}{2}x^T Ax + y^T Ax + \frac{1}{2}y^T Ay - x^T b - y^T b + c \\
&= (\frac{1}{2}x^T Ax - x^T b + c) + \frac{1}{2}y^T Ay + y^T(Ax - b) = f(x) + \frac{1}{2}y^T Ay.
\end{aligned}
$$

Gradient descent minimization

Since A is symmetric positive definite $y^T Ay > 0$, and the global minimum is the solution to the system of linear equations $x = A^{-1}b$. It follows that any system of linear equations with a symmetric positive definite matrix can be reformulated as the equivalent minimization problem of the associated quadratic form. To solve this minimization problem a gradient descent method can be formulated, where the gradient descent direction is equal to the residual,

$$-\nabla f(x^{(k)}) = b - Ax^{(k)} = r^{(k)}.$$

Hence, the gradient descent method is equivalent to a Richardson iteration,

$$x^{(k+1)} = x^{(k)} - \alpha \nabla^{(k)} f(x^{(k)}) = x^{(k)} + \alpha^{(k)} r^{(k)}.$$

To choose a step length we perform a line search to determine the $\alpha^{(k)}$ that minimizes $f(x^{(k+1)})$,

$$\frac{d}{d\alpha^{(k)}} f(x^{(k+1)}) = \nabla f(x^{(k+1)})^T \frac{d}{d\alpha^{(k)}} x^{(k+1)} = \nabla f(x^{(k+1)})^T r^{(k)} = -(r^{(k+1)})^T r^{(k)}.$$

Gradient descent minimization

It follows that

$$\begin{aligned}(b - Ax^{(k+1)})^T r^{(k)} &= 0,\\ (b - A(x^{(k)} + \alpha^{(k)} r^{(k)}))^T r^{(k)} &= 0,\\ (b - A(x^{(k)}))^T r^{(k)} - \alpha^{(k)} (Ar^{(k)})^T r^{(k)} &= 0,\end{aligned}$$

from which we are led to Algorithm 15.2, since

$$\alpha^{(k)} = \frac{(r^{(k)})^T r^{(k)}}{(Ar^{(k)})^T r^{(k)}} = \frac{(r^{(k)})^T r^{(k)}}{(r^{(k)})^T Ar^{(k)}} = \frac{\|r^{(k)}\|^2}{\|r^{(k)}\|_A^2}.$$

Gradient descent minimization

ALGORITHM 15.2. x = gradient_descent_method_Axb(A, b, x0).
Input: matrix **A**, vector **b**, initial guess **x0**.
Output: solution vector **x**.

```
1: x[:] = x0[:]
2: while norm(r)/norm(b) > TOL do
3:     r[:] = b[:] - matrix_vector_product(A, x)
4:     Ar = matrix_vector_product(A, r)
5:     alpha = scalar_product(r, r)/scalar_product(Ar, r)
6:     x[:] = x[:] + alpha*r[:]
7: end while
8: return x
```

Conjugate gradient method

Example 15.4 (Conjugate gradient method). Algorithm 7.4 describes the conjugate gradient method, which solves the minimization problem (15.4) by the search method

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} p^{(k)}. \tag{15.5}$$

The search directions $\{p^{(k)}\}_{k=0}^{n-1}$ are A-orthogonal (conjugated) by construction, through the choice of the parameter

$$\beta^{(k)} = \frac{\|r^{(k+1)}\|^2}{\|r^{(k)}\|^2}$$

in the update formula

$$p^{(k+1)} = r^{(k+1)} + \beta^{(k)} p^{(k)}. \tag{15.6}$$

Analogous to the gradient descent method, the step length $\alpha^{(k)}$ is determined by the condition that the residual $r^{(k+1)}$ should be orthogonal to $r^{(k)}$, and by equation (15.5),

$$0 = (r^{(k)})^T r^{(k+1)} = (r^{(k)})^T A e^{(k+1)} = (r^{(k)})^T A (e^{(k)} - \alpha^{(k)} p^{(k)}),$$

so that by equation (15.6) and the fact that $\{p^{(k)}\}_{k=0}^{n-1}$ are A-orthogonal,

$$\alpha = \frac{(r^{(k)})^T A e^{(k)}}{(r^{(k)})^T A p^{(k)}} = \frac{(r^{(k)})^T r^{(k)}}{(p^{(k)})^T A p^{(k)}} = \frac{\|r^{(k)}\|^2}{\|p^{(k)}\|_A^2}.$$

Least squares method

Example 15.5 (Least squares method). An important minimization problem that we have met in previous chapters is the least squares problem,

$$\min_{x \in R^n} f(x), \quad f(x) = \frac{1}{2}\|Ax - b\|^2,$$

where $A \in R^{m \times n}$ and $b \in R^m$, with $m > n$. The gradient is computed as

$$\begin{aligned}\nabla f(x) &= \frac{1}{2}\nabla(\|Ax - b\|^2) = \frac{1}{2}\nabla((Ax)^T Ax - (Ax)^T b - b^T Ax + b^T b) \\ &= \frac{1}{2}\nabla(x^T A^T Ax - 2x^T A^T b + b^T b) = \frac{1}{2}(A^T Ax + x^T A^T A) - A^T b = A^T(Ax - b),\end{aligned}$$

which gives the gradient descent method as a Richardson iteration for the normal equations. To solve this problem we can use Algorithm 15.2 for the matrix $A^T A$ and the vector $A^T b$,

$$x^{(k+1)} = x^{(k)} - \alpha^{(k)} A^T(Ax^{(k)} - b).$$

Level set

$$L_c(f) = \{x \in D : f(x) = c\},$$

where $L_c(f)$ represents a level curve in R^2, a level surface in R^3, and a hypersurface in R^n.

Theorem 15.6. *If $f \in \mathcal{C}^1(D)$, then $\nabla f(x)$ is orthogonal to $L_c(f)$ at any point $\bar{x} \in L_c(f)$.*

Proof. For $\bar{x} \in L_c(f)$, let $x(t)$ be a parameterized curve in the level set $L_c(f)$ such that $x(\bar{t}) = \bar{x}$. Since $x(t) \in L_c(f)$ for all t, we can define a function $g(t) = f(x(t)) = c$, for which

$$\frac{dg}{dt} = \frac{\partial f}{\partial x_1}\frac{dx_1}{dt} + ... + \frac{\partial f}{\partial x_n}\frac{dx_n}{dt} = \nabla f^T \frac{dx}{dt} = 0,$$

where dx/dt is the tangent of the curve $x(t)$. Hence, the gradient is orthogonal to any parameterized curve $x(t)$ along the level set, specifically to all curves through the point $x(\bar{t}) = \bar{x} \in L_c(f)$.

Level set

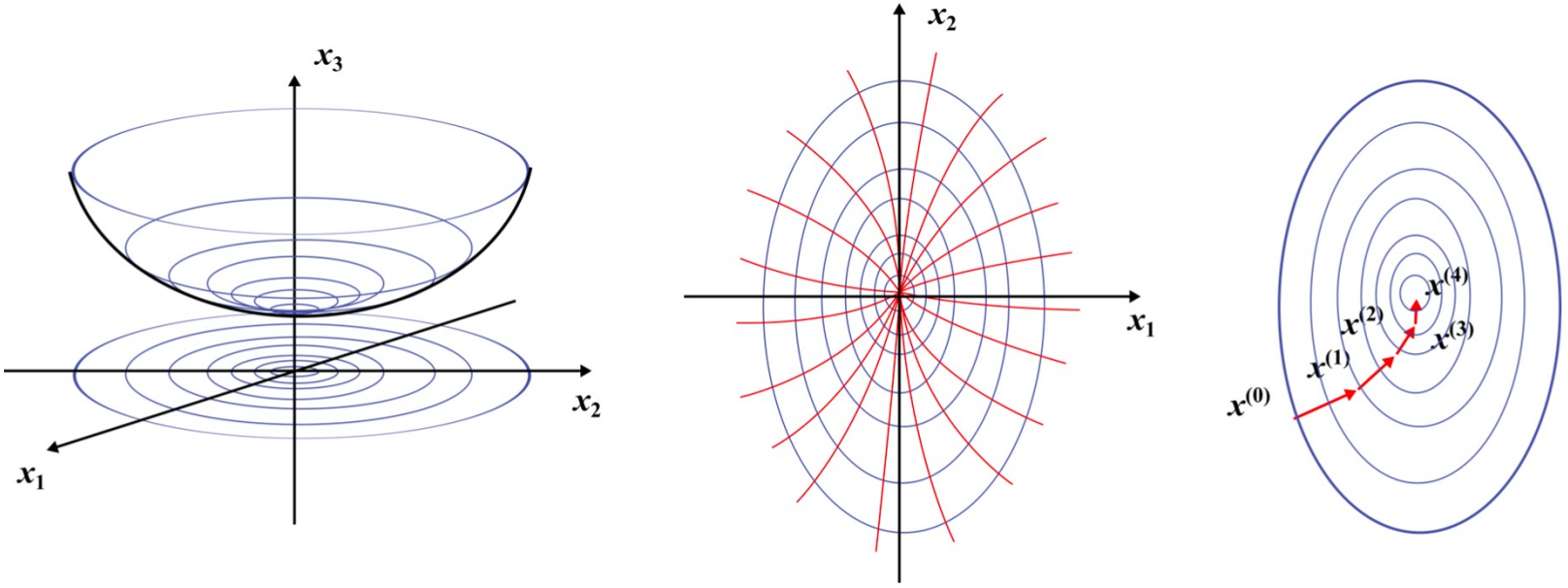
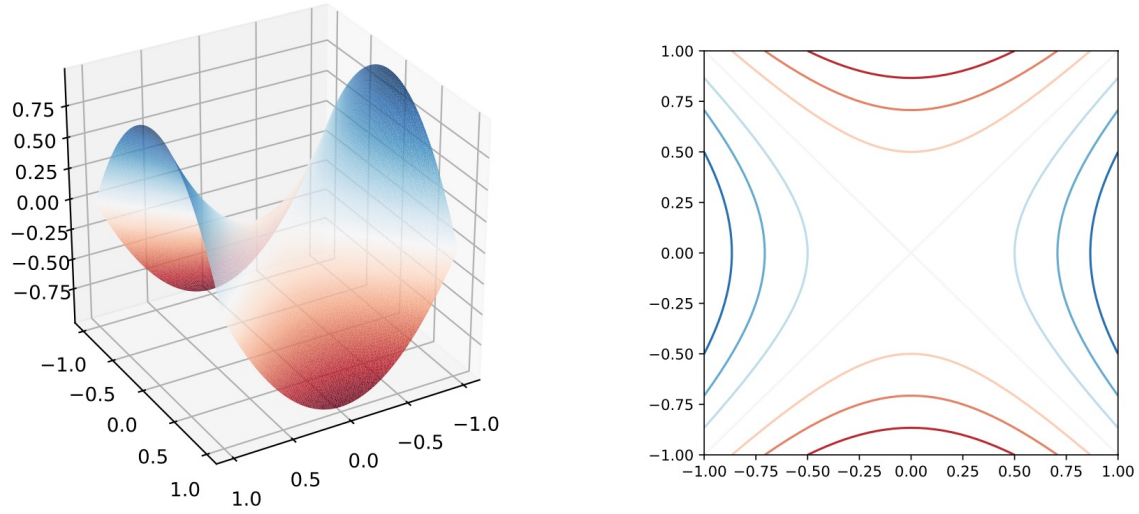


Figure 15.2. *Level curves of a convex function (left), blue level curves and red lines in the direction of the gradient near a minimum (center), and a number of gradient descent steps (right).*

Saddle points

If the Hessian has eigenvalues of both signs the critical point is a *saddle point*. To use a gradient based method to find a saddle point, in parts of the search space D we need to seek the saddle point in the direction of the negative gradient, whereas in other parts of D the saddle point is found in the direction of the positive gradient, see Figure 15.3.



Newton's method

Newton's method to find a local minimum is based on equation (15.7) with $x = x^{(k+1)}$ and $y = x^{(k)}$. We seek the critical point with respect to the increment $\Delta x = x^{(k+1)} - x^{(k)}$, where we let $\nabla_{\Delta x}$ denote the gradient with respect to Δx,

$$\begin{aligned}\nabla_{\Delta x} f(x^{(k+1)}) &= \nabla_{\Delta x} \left(f(x^{(k)}) + \nabla f(x^{(k)})^T \Delta x + \frac{1}{2} \Delta x^T H f(x^{(k)}) \Delta x \right) \\ &= \nabla f(x^{(k)}) + H f(x^{(k)}) \Delta x = 0.\end{aligned}$$

Hence, we are lead to Newton's method

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

Newton's method

ALGORITHM 15.3. x = newton_minimization(f, x0).
Input: function **f**, initial guess **x0**.
Output: solution vector **x**.

```
1: x[:] = x0[:]
2: Df = compute_gradient(f, x)
3: while norm(Df) > TOL do
4:     Df = compute_gradient(f, x)
5:     Hf = compute_hessian(f, x)
6:     dx = solve_linear_system(Hf, -Df)
7:     x[:] = x[:] + dx
8: end while
9: return x
```

Stochastic gradient method

Now consider a minimization problem for which the objective function is a sum,

$$\min_{x \in D} f(x), \quad f(x) = \sum_{i=1}^{n} f_i(x).$$

ALGORITHM 15.4. x = stochastic_gradient_descent_method(f_array, alpha, x0, no_iter).
Input: array of functions **f_array**, initial guess **x0**, step size **alpha**, number of iterations **no_iter**.
Output: solution vector **x**.

```
1: x[:] = x0[:]
2: for i=0:no_iter -1 do
3:     f = random_select(f_array)
4:     Df = compute_gradient(f, x)
5:     x[:] = x[:] - alpha*Df
6: end for
7: return x
```

Regularization

Minimization often represents an *ill-posed problem*, for which no unique solution exists. To make the problem *well-posed*, with a unique solution, one can favour or exclude certain solutions by modifying the objective function using *Tikhonov regularization*,

$$\min_{x \in D} f(x) + \|\Gamma x\|^2,$$

where $\Gamma \in R^{n \times n}$ is the *Tikhonov matrix*.

Example 15.7. *L^2-regularization* corresponds to $\Gamma = \epsilon I$, with $\epsilon \in R$ and I the $n \times n$ identity matrix, which favours solutions to the minimization problem which are small in the L^2 norm.

Multi-objective minimization

$$f : R^n \to R^m,$$

we are lead to a *multi-objective minimization problem*,

$$\min_{x \in D} f(x), \quad f(x) = (f_1(x), ..., f_m(x))^T.$$

A point $y \in D$ is said to be *Pareto dominated* by another point $x \in D$, if

$$f_i(x) \leq f_i(y), \quad \forall i = 1, ..., m,$$

and $f_j(x) < f_j(y)$ for at least one index j. A *Pareto optimal solution* is a point which is not Pareto dominated by any other point, and we refer to the set of Pareto optimal solutions as the *Pareto front*. To solve a multi-objective minimization problem we can seek to approximate the

Gradient-free minimization

Example 15.8 (Pattern search). The *Hooke-Jeeves algorithm*, also referred to as *pattern search*, is a search method, where in each step a local descent direction is constructed by sampling the objective function in a small neighbourhood of radius $\alpha > 0$ around the present point $x^{(k)}$ in the search space. If no descent direction can be detected, the search radius α is reduced until $\alpha <$ TOL. To reduce the risk to get trapped in a loop, randomness can be added to the local search direction.

Gradient-free minimization

Example 15.9 (Particle swarm). Minimization by a particle swarm method is based on the idea of a collection of M evolving particles that represent points in the search space $\{x_i\}_{i=1}^{M} \subset D$. The update of the position of each particle is given by the time stepping algorithm

$$x_i^{(n+1)} = x_i^{(n)} + k_n v_i^{(n+1)}$$

with k_n a time step and $v_i^{(n+1)}$ the velocity of particle p_i at time step n. The velocity is selected as a linear combination of the velocity at the previous time step (inertia) $v_i^{(n)}$, the best position of the particle so far $p_i^{(n)}$, and the best global position of the whole swarm $p_g^{(n)}$, which is information that all particles share with each other. Hence, the total velocity for step i is given by

$$v_i^{(n+1)} = c_0 v_i^{(n)} + c_1 r_1 (p_i^{(n)} - x_i^{(n)})/k_n + c_2 r_2 (p_g^{(n)} - x_i^{(n)})/k_n,$$

where c_0, c_1, c_2 are parameters of the method, and r_1, r_2 random numbers in the interval $[0, 1]$.

Gradient-free minimization

Example 15.10 (Evolutionary algorithms). Evolutionary algorithms are inspired by biological evolution. The basic ingredients are a *population* of individuals, a *selection* criterion for *fitness*, and a mechanism for generation of new individuals that inherit some characteristics of the most fit individuals, to replace the least fit individuals in the next generation. The individuals of a population are the points in the search space $\{x_i\}_{i=1}^{M} \subset D$, for which the objective function $f(x)$ is used to rank the fitness of each individual. New individuals of the population are generated by *crossover*, a combination of the highest ranked individuals from the population, and *mutation*, random modifications of individuals. Evolutionary algorithms are popular for multi-objective minimization, since the population can be used to approximate the Pareto front.

Gradient-free minimization

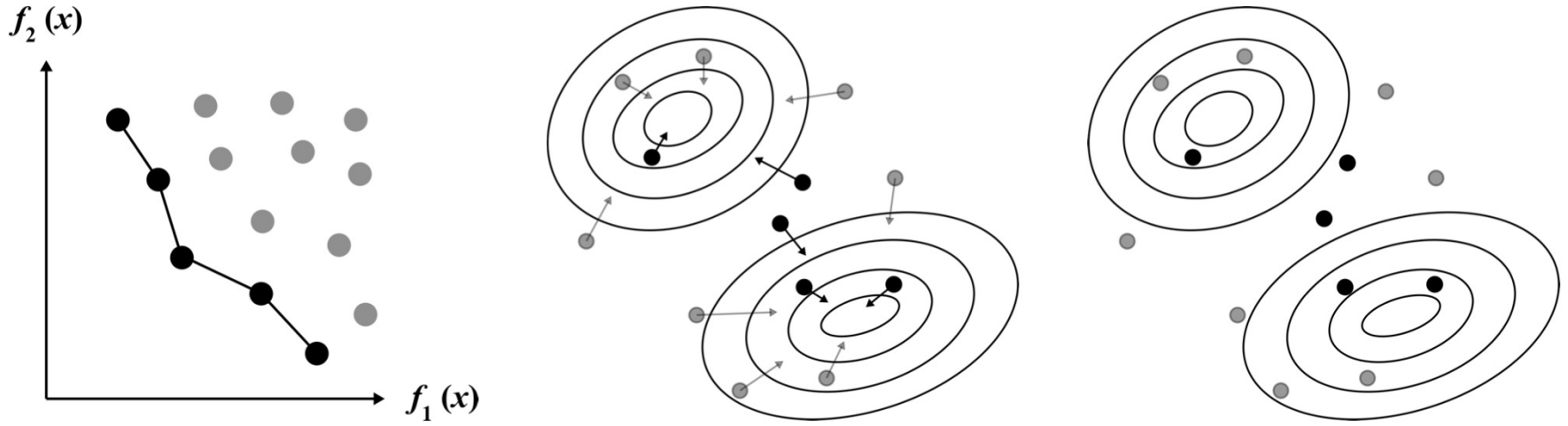


Figure 15.4. *Minimization of an objective function $f(x)$ with two components, $f_1(x)$ and $f_2(x)$, that represent two separate global minima, here visualized as a diagram and as level curves. We illustrate a particle swarm method (center) and an evolutionary algorithm (right), with the particles and individuals marked by points. Grey points are Pareto dominated, whereas black points approximate the Pareto front, connected by a line in the diagram (left). In the particle swarm the velocity of each particle is indicated by an arrow, and in the evolution algorithm the highest ranked individuals correspond to the Pareto front.*

Constrained minimization

Often the search space is defined by a set of equality and inequality constraints, for which the *constrained minimization problem* in R^n takes the following form,

$$\begin{aligned} &\min_{x \in R^n} f(x), \\ &g(x) = c, \\ &h(x) \leq d, \end{aligned} \tag{15.8}$$

with the objective function $f : R^n \rightarrow R$, and the constraints defined by the two functions $g : R^n \rightarrow R^m$ and $h : R^n \rightarrow R^l$, and the vectors $c \in R^m$ and $d \in R^l$. If the objective function is linear, then any optimal solution x^* must be located on the boundary of the search space. But if the objective function is nonlinear, minima can exist both as boundary points and as interior points. Hence, for a linear objective function we seek minima on the boundary of the search space, whereas for a nonlinear objective function we must also search for minima inside the search space.

Constrained minimization

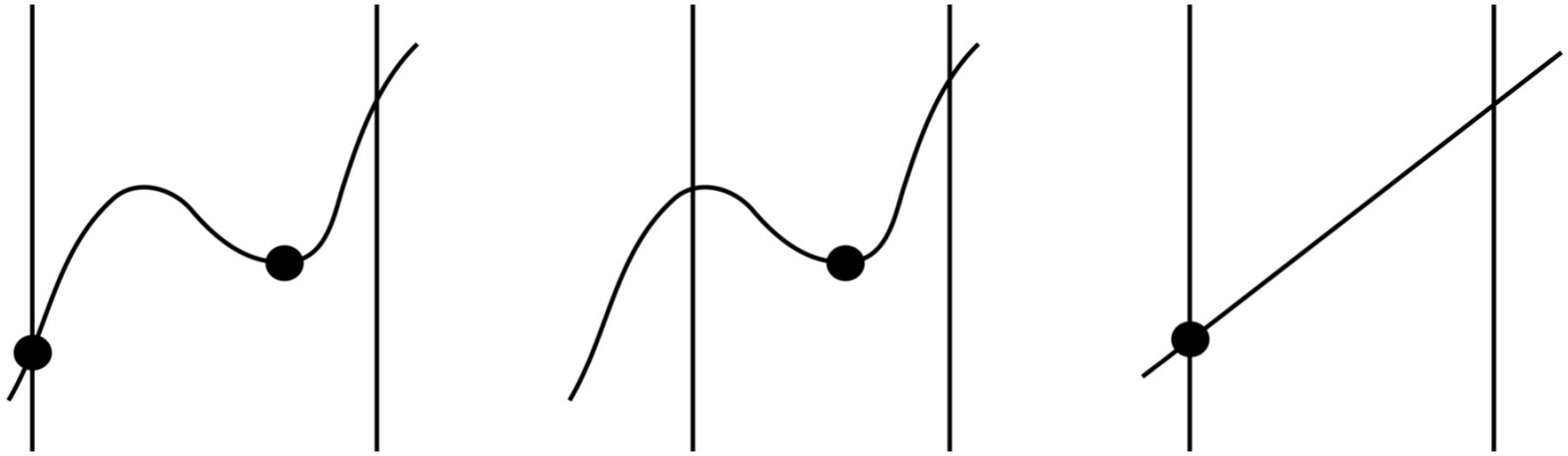


Figure 15.5. *Illustration of three types of constrained minimization problems, in terms of their objective functions, and two inequality constraints represented by vertical lines. The first problem has an interior local minimum in the form of a critical point, but the global minimum is on the boundary (left), the global minimum of the second problem is an interior critical point (center), and the third problem has a linear objective function and therefore the global minimum must be on the boundary (right).*

Projected search methods

To ensure that the steps of a gradient descent method remains inside the search domain we can use a projection of the gradient descent direction. This leads to a *projected gradient descent method*, based on search directions

$$-P_D \nabla f(x^{(k)}),$$

with P_D a projection operator onto D. If a local minimum is located on the boundary ∂D, the projection of the gradient descent direction is zero, so that we can use as the stopping criterion

$$\|\nabla P_D f(x^{(k)})\| < \text{TOL}.$$

Projected Newton methods are based on the same idea, in each step of the algorithm the search direction is projected onto the search space D.

Penalty methods

In a *penalty method* we reformulate the constrained minimization problem as an unconstrained minimization problem. A penalty function is added to the objective function, which consists of a penalty parameter that multiplies a measure of the violation of the constraints. One common choice is a quadratic loss function, based on the l^2 norm, another is a loss function based on the l^1 norm. In the case of an inequality constraint, we use the maximum function to switch the penalty on and off based on activation of the constraint. If we apply a penalty method with quadratic loss functions to the constrained minimization problem (15.8), the new objective function is

$$\bar{f}(x) = f(x) + \gamma_1 \|g(x) - c)\|^2 + \gamma_2 \| \max(0, h(x) - d)\|^2,$$

where $\gamma_1, \gamma_2 > 0$ are penalty parameters.

When minimizing the objective function by an search method we can choose to start inside or outside the search space, that is, with the constraints satisfied or not. If we start from the outside we have an *exterior penalty method*, and from the inside an *interior penalty method*.

Linear and quadratic programming

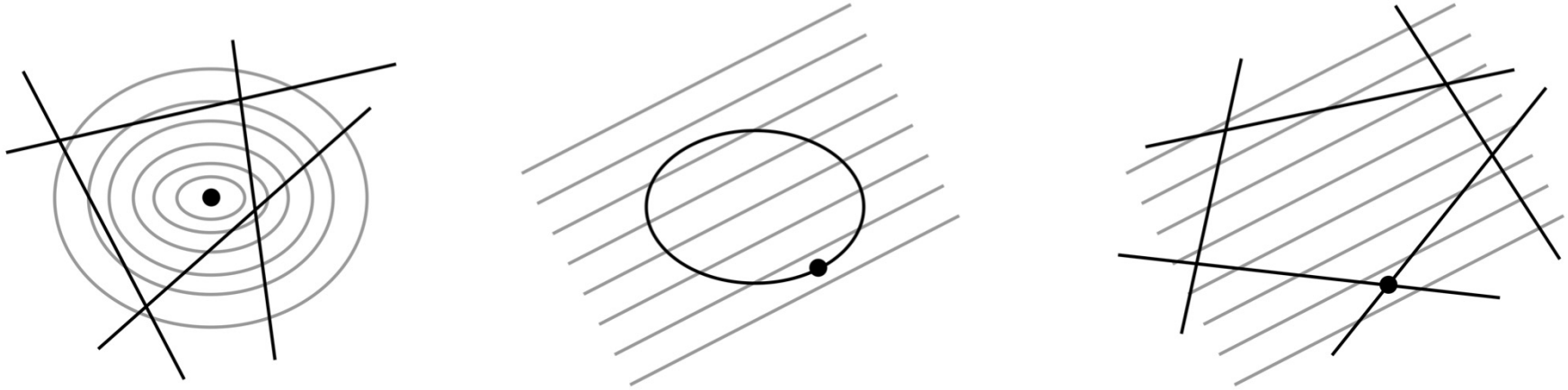


Figure 15.6. *Three constrained minimization problems with their objective functions visualized by level curves and their minima marked by points; quadratic programming with a quadratic objective function and linear constraints (left), a linear objective function and a quadratic constraint (center), and linear programming where both the objective function and the constraints are linear.*

Lagrange multipliers

In the case we have only equality constraints in the constrained minimization problem (15.8), we can define a *Lagrangian* $L : R^{n+m} \to R$, which takes the form

$$L(x, \lambda) = f(x) - \lambda^T (g(x) - c),$$

for the *primal variable* $x \in R^n$ and the *dual variable* $\lambda \in R^m$. λ is also referred to as the *adjoint variable*, or the *Lagrangian multipliers*. By seeking the critical point of the Lagrangian, we obtain the necessary optimality conditions

$$\nabla_x L(x, \lambda) = \nabla f(x) - \lambda^T \nabla g(x) = 0, \tag{15.10}$$

$$\nabla_\lambda L(x, \lambda) = g(x) - c = 0, \tag{15.11}$$

$n + m$ equations from which we can determine the solution

$$(x, \lambda) \in R^{n+m}.$$

Lagrange multipliers

A geometric interpretation may be intuitive, where the gradient $\nabla g_i(x)$ represents a normal vector to the hypersurface that is defined by the equation $g_i(x) = c_i$. In other words, the gradient $\nabla g_i(x)$ is orthogonal to the tangent hyperplane at $x \in R^n$. The gradients $\nabla g_i(x)$ span a vector space

$$S_x = \text{span}(\{\nabla g_i(x)\}_{i=1}^m),$$

which is orthogonal to the local space of constrained search directions at x. Assuming the set of gradients $\{\nabla g_i(x)\}_{i=1}^m$ is linearly independent, it constitutes a basis for S_x. Hence, a necessary condition for a constrained local minimum at $x \in R^n$ is that the gradient of the objective function is not aligned with any feasible search direction in the constrained search space. That is, that the gradient $\nabla f(x) \in S_x$,

$$\nabla f(x) = \lambda^T \nabla g(x) = \sum_{i=1}^{m} \lambda_i \nabla g_i(x),$$

where the Lagrange multipliers λ_i are the coordinates of $\nabla f(x)$ in the basis $\{\nabla g_i(x)\}_{i=1}^m$. With only one constraint in R^2, we can visualize the geometric interpretation as the gradients $\nabla f(x)$ and $\nabla g(x)$ being parallel at a constrained local minimum $x \in R^2$.

Lagrange multipliers

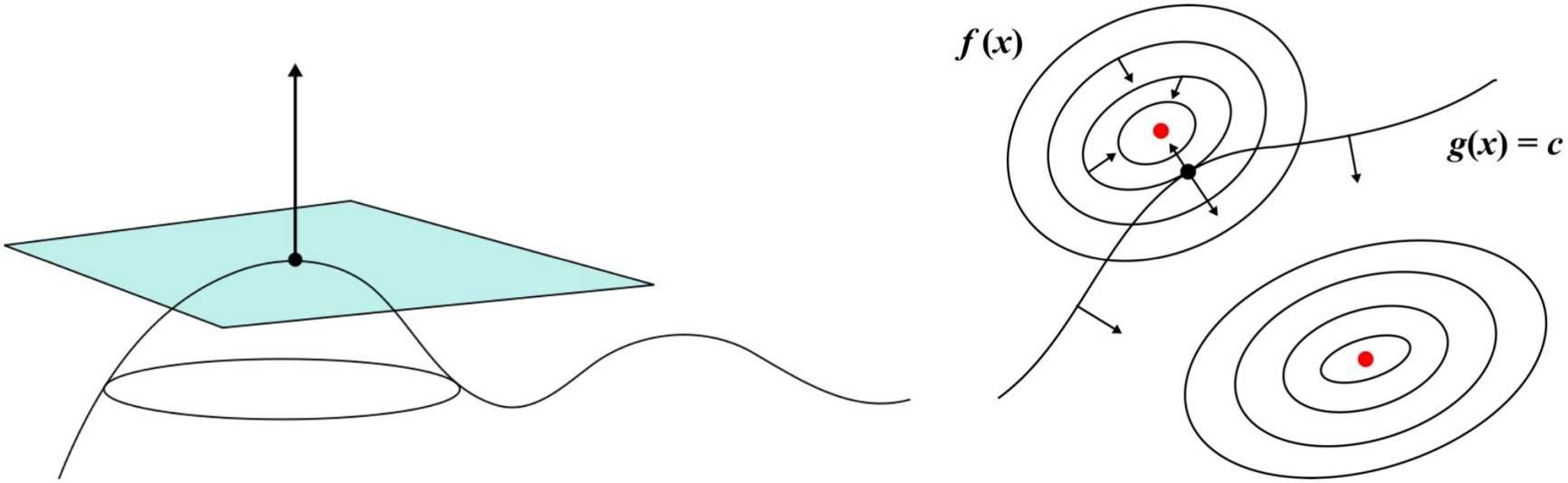


Figure 15.7. *A normal vector orthogonal to the tangent plane of a surface in* R^3 *(left). The Lagrange multiplier method (right) with level curves of the objective function* $f(x)$*, and one constraint* $g(x) = c$ *in* R^2*, illustrating that the gradients* $\nabla f(x)$ *and* $\nabla g(x)$ *are parallel at the constrained minimum.*

Lagrange multipliers

Example 15.13. Consider the constrained minimization problem

$$\min_{x \in R^2} x_1^2 + x_2^2,$$
$$x_2 - 2x_1 = 1,$$

for which the Lagrangian takes the form

$$L(x, \lambda) = x_1^2 + x_2^2 - \lambda(x_2 - 2x_1 - 1).$$

The optimality conditions give that

$$(2x_1, 2x_2)^T - \lambda(-2, 1)^T = 0,$$

from which we conclude that the constrained minimum is $x = (-2/5, 1/5)$, and that $\lambda = 2/5$.

Lagrange multipliers

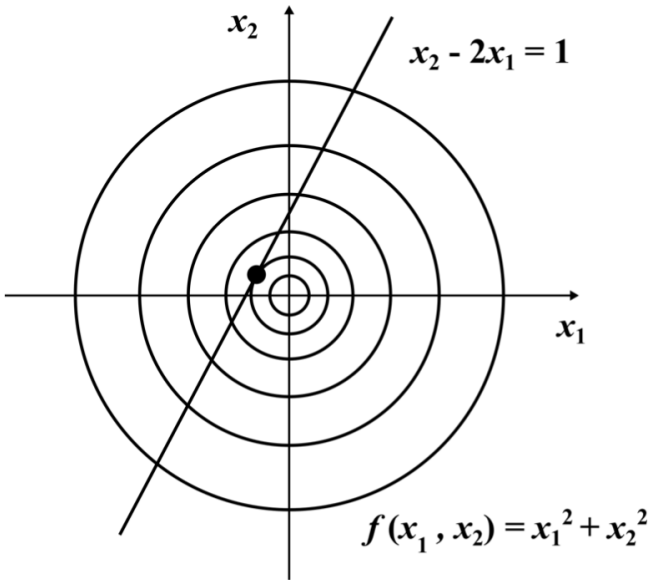


Figure 15.8. *Example of a constrained minimization problem with objective function* $f(x_1, x_2) = x_1^2 + x_2^2$ *and constraint* $x_2 - 2x_1 = 1$, *where the constrained minimum is marked by a dot.*

Lagrange multipliers

Example 15.14 (Adjoint based error analysis). Consider a constrained minimization problem with a linear objective function and linear equality constraints,

$$\begin{aligned} &\min_{x \in R^n} c^T x, \\ &Ax = b, \end{aligned}$$

where $x, b, c \in R^n$ and $A \in R^{n \times n}$. The Lagrangian $L : R^n \times R^n \to R$ takes the form

$$L(x, \lambda) = c^T x - \lambda^T (Ax - b),$$

with the adjoint variable $\lambda \in R^n$, from which we obtain the optimality conditions,

$$\begin{aligned} \nabla_x L(x, \lambda) &= c - A^T \lambda = 0, \\ \nabla_\lambda L(x, \lambda) &= -(Ax - b) = 0, \end{aligned}$$

corresponding to equations for the primal and adjoint variables,

$$\begin{aligned} Ax &= b, \\ A^T \lambda &= c. \end{aligned}$$

Lagrange multipliers

If the matrix A is nonsingular the solution to the primal problem defines the search space as the unique point $x = A^{-1}b$, so that the minimization problem is trivial. But the problem is still of interest. First, assuming that the matrix A is given but the vector b may vary, we can use the adjoint equation to directly evaluate the objective function for different vectors b without solving the primal equation for each b, but instead just multiply by the adjoint variable,

$$c^T x = (A^T \lambda)^T x = \lambda^T A x = \lambda^T b.$$

Second, for an approximation of the primal problem $\bar{x} \approx x$ we can estimate the error with respect to the objective function, in terms of the adjoint variable and the residual $R(\bar{x}) = b - A\bar{x}$,

$$c^T x - c^T \bar{x} = c^T (x - \bar{x}) = (A^T \lambda)^T (x - \bar{x}) = \lambda^T (Ax - A\bar{x}) = \lambda^T R(\bar{x}).$$

For the nonlinear problem (15.10)-(15.11) the corresponding analysis gives

$$\begin{aligned} f(x) - f(\bar{x}) &\approx \nabla f(\bar{x})(x - \bar{x}) = \lambda^T \nabla g(\bar{x})(x - \bar{x}) \\ &\approx \lambda^T (g(x) - g(\bar{x})) = \lambda^T (c - g(\bar{x})) = \lambda^T R(\bar{x}). \end{aligned}$$

Augmented Lagrangian method

Instead of computing the adjoint variables from the optimality conditions, we can design a search method where we in each step seek to minimize the Lagrangian only with respect to the primal variable x, while keeping the adjoint variable λ constant. For this purpose we define the *augmented Lagrangian*

$$L_\gamma(x, \lambda) = f(x) - \lambda^T(g(x) - c) + \frac{\gamma}{2}\|g(x) - c\|^2,$$

which we can interpret as a Lagrangian regularized by a quadratic loss function, or alternatively as a penalty method with penalty parameter γ. In a search algorithm such as a gradient descent method or a Newton method, we update the adjoint variables and the penalty parameter by

$$\lambda_i^{(k+1)} = \lambda_i^{(k)} - \gamma^{(k)}(g(x^{(k)}) - c), \quad \gamma^{(k+1)} = \beta\gamma^{(k)}, \quad \beta > 1.$$

We note that with an approximation of the primal and adjoint variables close to the critical point of the Lagrangian, the penalty term is small.

Optimal control

If a dynamical system is governed by a system of initial value problems in which there are free parameters, we can seek to formulate an optimal control strategy to modify these parameters over time to achieve a certain goal. Consider a dynamical system of the form

$$\dot{u}(t) = f(m(t), u(t), t), \quad u(0) = u_0, \quad t \in [0, T], \tag{15.13}$$

where $u(t) \in R^n$ is the state variable and $m(t) \in R^d$ is the control variable. For L a running cost and g a final cost, the optimal control problem is then to minimize a cost functional

$$J(m, u, T) = g(u(T)) + \int_0^T L(m(t), u(t), t)\, dt. \tag{15.14}$$

Optimal control

Pontryagin's principle states that an optimal control $m_{opt}(t)$ only exists provided that a certain primal-dual system is satisfied, which represents necessary optimality conditions in the context of dynamical systems. The construction of the optimality conditions is a generalization of Hamiltonian mechanics, by formulating a Hamiltonian

$$H(m, u, \lambda, t) = \lambda^T f(m, u, t) + L(m, u, t), \tag{15.15}$$

$$\begin{aligned} \dot{u} &= \frac{\partial H}{\partial \lambda} = f, \\ -\dot{\lambda}^T &= \frac{\partial H}{\partial u} = \lambda^T (\nabla_u f) + (\nabla_u L), \end{aligned}$$

with a final condition for the adjoint variable $\lambda(T) = \nabla_u g(u(T))$. These equations offer a method to compute an optimal solution, by minimization of the Hamiltonian with respect to the control variable $m(t)$.

Optimal control

Example 15.19. Now consider the system of nonlinear initial value problems

$$\dot{u}(t) = f(u(t)), \quad u(0) = u_0, \quad t \in [0, T],$$

where $u : [0, T] \to\in R^n$ and $f : R^n \to\in R^n$. If we have computed an approximate solution $U(t) \approx u(t)$ for which $U(0) = u_0$, the primal-dual system takes the form

$$\begin{aligned} \dot{u} &= f(u), \\ -\dot{\lambda} &= \bar{f}'(w)^T \lambda + \psi, \end{aligned}$$

with initial condition $u(0) = u_0$ and final condition $\lambda(T) = 0$, and with $\bar{f}'(w)$ defined by

$$f(u) - f(U) = \bar{f}'(w)(u - U).$$

Optimal control

$$\int_0^T \psi^T u - \psi^T U \, dt = \int_0^T \psi^T (u - U) \, dt = \int_0^T (-\dot{\lambda}^T + \lambda^T \bar{f}'(w))(u - U) \, dt$$
$$= \int_0^T \lambda^T (\dot{u} + f(u) - \dot{U} - f(U)) \, dt = \int_0^T \lambda^T R(U) \, dt,$$

with the residual $R(U(t)) = -\dot{U}(t) - f(U(t))$.