Chapter 2

INVERSE PROBLEMS AND ERROR MINIMISATION

2.1 A Copernican revolution: direct and inverse problems

In engineering science, direct problems are defined as those where, given the input or the cause of a phenomenon or of a process in a device, the purpose is that of finding the output or the effect.
Inverse problems, conversely, are those where, given the measured or expected output or effect, one wants to determine the input or the cause; moreover, inverse problems are also those where, given the input and the corresponding output, one tries to understand their interconnection.
The two types of problems, when applied to the same phenomenon or process, represent the two logical ways of conceiving it: from input to output or the other way round. The latter way is central for design.
In applied electromagnetics, inverse problems may appear in two forms:

i) given measured data, which may be affected by noise or error, in a field region, to identify or recover the relevant field sources or material properties or boundary conditions of the region (identification or parameter-estimation problems);
ii) given desired fields in a device, or given the device performance based on them, to determine, or design, sources or materials or shape of the device, producing the specified performance (synthesis or optimal design problems).

In particular, optimal design problems, which are very popular in all branches of engineering, belong to a group of inverse problems where the purpose is to design a device which can provide a prescribed behaviour or an optimal performance. If optimal design problems should be solved only by means of a trial-and-error approach, it would not be possible to know something *a priori* about their solution, which would rely just on the designer experience and intuition. On the contrary, the study of inverse problems puts the ground for a systematic approach to the design.

2.2 Insidiousness of inverse problems

Despite the historical background summarized in the previous Section, the greater part of engineering science is dominated by direct problems, *i.e.* problems that can be characterized as those in which exactly enough information is provided to carry out a unique solution. A general description of direct problems may be described as follows: let x, y and A symbolyze the input, the output and the operator modelling the input-to-output transformation, respectively. Then, the direct problem is to find Ax, *i.e.* the value of the given operator at a point in its domain. Assuming that the operator A is invertible, the inverse problem for A is the direct problem for A^{-1} ; if A is not invertible, the solution of the inverse problem does not exist.

On the other hand, if operator A represents a function, then for any given input x in its domain, a unique output y is determined: in other words, the direct problem has a unique solution. There is no guarantee, however, that the inverse problem $A^{-1}y$ has a unique solution: $A^{-1}y_1$ might be equal to $A^{-1}y_2$ even if y_1 and y_2 are different.

Moreover, if the operator A is continuous in some sense, then the solution of the direct problem is stable with respect to small changes in the input, *i.e.* $\frac{dy}{y}$ is small if $\frac{dx}{x}$ is small. Even when the operator has a well-defined inverse A^{-1}, so that the inverse problem is uniquely solvable, there is no guarantee that its solution is stable against small changes dy; the inverse operator may, in fact, be discontinuous.

From the mathematical viewpoint, following the Hadamard's definition, well-posed problems (or properly, correctly posed problems) are those for which:

i) a solution always exists;
ii) there is only one solution;
iii) a small change of data leads to a small change in the solution.

The last property implies that the solution depends continuously upon the data, which often are measured quantities and therefore are affected by noise or error.

Ill-posed problems, instead, are those for which:

i) a solution may not exist;
ii) there may be more than one solution;
iii) a small change of data may lead to a big change in the solution.

2.3 Classification of inverse problems

There are many ways to classify inverse problems. The formulation of inverse problems in electricity and magnetism implies to associate a procedure for field computation (direct problem) and a procedure for the solution of the inverse problem. Therefore, a classification can be based on the approach for field computation (integral or differential, analytical or numerical). Another classification can be made, according to the formulation of the inverse problem and the relevant mathematical method employed for the solution; this viewpoint will be developed later.

When the given data come from measurements and the parameters governing field equations, including material properties, are to be found, one speaks of identification problems.

Otherwise, when the given data are arbitrarily taken and the field source or specifications of the field region (*e.g.* boundary conditions) are required, the problem is called a synthesis problem.

In engineering applications, often, the goal is to design the geometry of a device so that a prescribed performance of the device, depending on the field, is obtained. This kind of problem is commonly defined as optimal shape design problem.

The ultimate goal of the problem is to perform an Automated Optimal Design (AOD), when the solution is obtained automatically in terms of the required or best performance.

About ill-posed problems, the following remark can be put forward.

Identification problems have always a solution at least, while a solution may not exist for optimal design problems; this happens when *e.g.* the prescribed quantity does not fit with data. On the contrary, if multiple solutions exist to a given problem, they may differ by *e.g.* a degree of smoothness or exactness.

2.4 Green's formula and Fredholm's equation

In field theory, using an integral approach (*e.g.* Green's function method, moment method), equations of the type:

$$g(x) = \int_{\Omega_1} K(x,y) f(y) dy \ , \ x \in \Omega_0 \ , \ y \in \Omega_1 \tag{2.1}$$

where Ω_0 is the field domain and Ω_1 is the source domain, are frequently dealt with.
When f is given, g is the unknown and K is the known kernel, the problem of finding g is a direct problem. In a sense, (2.1) is nothing but the Green's formula of magnetostatics in an unbounded domain.
In turn, when g is given, f is the unknown and K is the known kernel, the problem of finding f is an inverse problem. In this case, (2.1) is called Fredholm's equation of the first kind.
Normally, kernel K is assumed to be bounded, *i.e.* a constant $p > 0$ exists such that $|K(x,y)| \le p$, and symmetrical, *i.e.* $K(x,y) = K(y,x)$. It can be shown that f does not depend continuously upon the given function g. Therefore, Fredholm's equation gives origin to an ill-posed problem. An example dealing with magnetic field is now presented; a few relationships between field and potential, which will be developed in Chapter 4, are here anticipated.

2.3.1 Case studies

i) Analysis of a known current distribution producing a magnetic field

Given a system of rectangular coordinates, let a two-dimensional unbounded domain, which is supposed to be homogeneous and free of ferromagnetic material, be considered. The problem is that of determining the induction field $\bar{B}(x,y)$ produced in a subregion Ω_0 , defined by $\alpha_1 \le x \le \alpha_2$ and $\beta_1 \le y \le \beta_2$ (Fig. 2.1) due to a direct current, flowing along a z-directed conductor in air and distributed with density $J(x',y')$ in a subregion Ω_1, defined by $-\frac{a}{2} \le x' \le \frac{a}{2}$ and $-\frac{b}{2} \le y' \le \frac{b}{2}$.

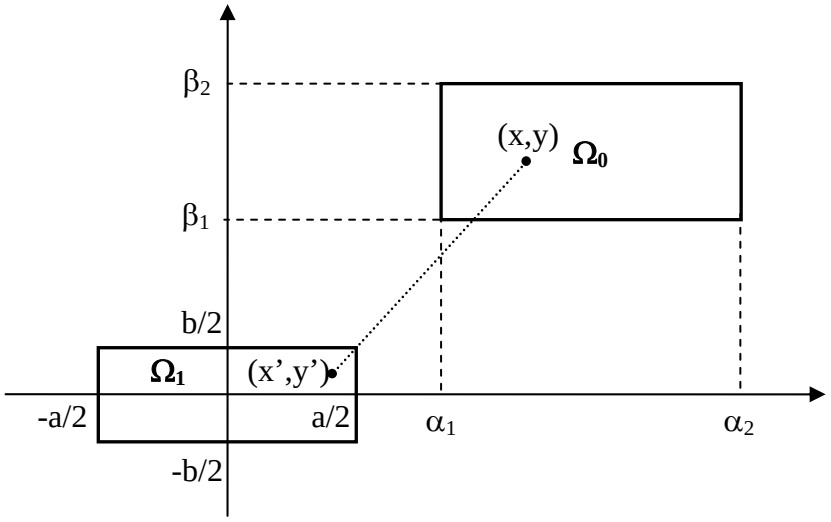

Fig. 2.1 – Current-carrying conductor Ω_1 producing a field in Ω_0 .

The vector potential A [Wbm^{-1}], parallel to J [Am^{-2}], due to the current-carrying conductor of rectangular cross-section, centred at the origin and having width a and height b, has amplitude given by

$$A(x,y)=\frac{\mu_0}{4\pi}\int_{-\frac{a}{2}}^{\frac{a}{2}}\int_{-\frac{b}{2}}^{\frac{b}{2}}\ln\sqrt{(x-x')^2+(y-y')^2}\;J(x',y')\,dy'\,dx' \qquad (2.2)$$

where $(x,y)\in\Omega_0$, $(x',y')\in\Omega_1$ and μ_0 is the material permeability. It can be noted that in (2.2) the kernel $K(x,y)=\frac{1}{2\pi}\ln\sqrt{(x-x')^2+(y-y')^2}$ is equal to the Green's function in an unbounded two-dimensional domain. From the physical point of view, $\frac{\mu_0}{2}K\,dI$ is the magnetic potential of a filamentary conductor carrying current $dI = J\,dx'dy'$. After integrating (2.2), and so recovering the effect of the finite cross-section of the actual conductor, the field is given by $\overline{B}(x,y)=\mu_0^{-1}\left(\frac{\partial A}{\partial y},-\frac{\partial A}{\partial x}\right)$.

ii) Synthesis of a current distribution producing a known magnetic field

Under the same assumptions as in the previous case study, the problem is that of identifying the density $J(x',y')$ with $-\frac{a}{2} \le x' \le \frac{a}{2}$ and $-\frac{b}{2} \le y' \le \frac{b}{2}$ of the direct current flowing along a z-directed conductor in air, such that the magnetic field produced in a sub-region Ω_0 , defined by $\alpha_1 \le x \le \alpha_2$ and $\beta_1 \le y \le \beta_2$ (Fig. 2.1) , is equal to a prescribed value.

Again, the vector potential is given by (2.2); if induction $\overline{B}_0(x,y)$ [Wbm^{-2}] is prescribed in Ω_0, one can get the corresponding potential A by integrating the relationships $B_x = \frac{\partial A}{\partial y}$ and $B_y = -\frac{\partial A}{\partial x}$. Therefore, equation (2.2), the left-hand side of which is assumed to be known, becomes a Fredholm's equation of the first kind in the unknown $J(x',y')$. The solution can be arranged by a numerical technique.

As far as the sensitivity against small changes in the input is concerned, the kernel gradient gives:

$$\frac{\partial K}{\partial x} = \frac{x - x'}{2\pi\sqrt{(x-x')^2 + (y-y')^2}\,\ln\sqrt{(x-x')^2 + (y-y')^2}} = \\ = \frac{x - x'}{4\pi^2 K e^{2\pi K}} \tag{2.4}$$

and

$$\frac{\partial K}{\partial y} = \frac{y - y'}{2\pi\sqrt{(x-x')^2 + (y-y')^2}\,\ln\sqrt{(x-x')^2 + (y-y')^2}} = \\ = \frac{y - y'}{4\pi^2 K e^{2\pi K}} \tag{2.5}$$

respectively, which can be regarded as components of the error amplification.

2.5 Solving inverse problems by minimising a functional

In general, the unknowns x of an inverse problem are called design variables. They are real values, although in some cases they are integer, belonging to a feasible region $\Omega \subseteq R^{n_v}$. In multivariate problems, $n_v > 1$.
The design variables may be geometric coordinates of the field region or values of sources or parameters characterizing the region.
The solution of the inverse problem is generally performed by means of the minimisation of a suitable function f(x) called objective function or cost function or design criterion:

given $x_0 \in \Omega \subseteq R^{n_v}$

$$\text{find } \inf_x f(x) \quad , \quad x \in \Omega \subseteq R^{n_v} \tag{2.6}$$

where x_0 is the initial guess. Properly speaking, (2.6) is a problem of unconstrained optimisation; for (2.6) to be meaningful, it is assumed that f is bounded in Ω.
This function may represent some performance depending on the field, or simply the residual between computed and known field values (error functional). In the second case study of Section 2.3.1, for instance, a suitable functional to be minimised is the norm of the discrepancy between actual and prescribed magnetic induction in region Ω_0, depending on the specific current J in region Ω_1, *i.e.* $\left\| \bar{B}(x, y, J(x', y')) - \bar{B}_0(x, y) \right\|$ with $(x, y) \in \Omega_0$ and $(x', y') \in \Omega_1$.
In general, the objective function f, which depends on the field, is not known analytically. Consequently, the classical conditions of minimality for unconstrained problems (*i.e.* null gradient and positive-definite Hessian matrix) cannot be applied *a priori*, because the objective function is known only numerically as a set of values at sample points. Moreover, f might be neither convex nor differentiable or smooth: therefore, it is not guaranteed to get solutions; in particular, f might exhibit some local minima in addition to the global one. Any way, a solution to (2.6) can be obtained numerically and the procedure may be troublesome and time-consuming.

The numerical solution of inverse problems in electromagnetics require, as a rule, a routine for calculating the field, which is integrated with a routine minimising the objective function.
Usually, the device or system to be optimised is represented by a finite-element model in two or three dimensions. The main flow of the computation is driven by the minimisation routine, which in the simplest

way is carried out step by step. Starting from x_0 , an iterative procedure updates the current design point x_k as

$$x_{k+1} = x_k + \alpha s_k \tag{2.7}$$

where k is the iteration index, α is a scalar displacement and s_k is the current search direction within the feasible region. Given x_{k+1} , the routine of field analysis generates a new finite-element grid, the field simulation is restarted and the evaluation of f(x) is so updated.
At the end of computation, the result could represent either a local minimum or a saddle point or simply a point better than the initial one because f has decreased; in the latter case, a mere improvement (and not the optimisation) of f has been achieved. In general, the optimisation trajectory can converge to different local minima, depending on the initial point, and the global optimum cannot be derived from the local behaviour of the objective function.

2.6 Constrained minimisation

In a more advanced formulation, the objective function should fulfil constraints, which may be expressed as inequalities, equalities and side bounds. Formally, the problem can be stated as follows:

$$\text{given } x_0 \in \Omega \subseteq R^{n_v} \tag{2.8}$$

$$\text{find } \inf_x f(x) \quad , \quad x \in \Omega \subseteq R^{n_v}$$

subject to

$$g_i(x) \le 0 \qquad i = 1,...,n_i \tag{2.9}$$

$$h_j(x) = 0 \qquad j = 1,...,n_e \tag{2.10}$$

$$\ell_k \le x_k \le u_k \qquad k = 1,...,n_v \tag{2.11}$$

Constraints and bounds set the boundary of the feasible region Ω and define implicitly its shape in the n_v-dimensional design space.

2.6.1 Classical optimality conditions

The Lagrangian function L associated to the constrained optimisation problem (2.8)-(2.11) is defined by

$$L(x,\lambda) = f(x) + \sum_{i=1}^{n_i} \lambda_i g_i(x) + \sum_{j=1}^{n_e} \lambda_j h_j(x) \tag{2.12}$$

where λ_i are called Lagrange's multipliers. Classical optimality dictates a first-order necessary condition (Kuhn-Tucker theorem): let $\tilde{x}$ be a local minimum point for problem (2.8)-(2.11) and let f, g_i , h_i differentiable functions. Then, there exists a vector $\tilde{\lambda} \in \Re^{n_i + n_e}$ of (unknown) multipliers such that $L(\tilde{x}, \tilde{\lambda})$ is steady, namely

$$\overline{\nabla}_x L(\tilde{x}, \tilde{\lambda}) = \overline{\nabla} f(\tilde{x}) + \sum_{i=1}^{n_i} \tilde{\lambda}_i \overline{\nabla} g_i(\tilde{x}) + \sum_{i=1}^{n_e} \tilde{\lambda}_j \overline{\nabla} h_j(\tilde{x}) = 0 \tag{2.13}$$

and, in addition,

$$g_i(\tilde{x}) \le 0 \ , \ i = 1,..., n_i \tag{2.14}$$

$$h_j(\tilde{x}) = 0 \ , \ j = 1,..., n_e \tag{2.15}$$

$$\tilde{\lambda}_i g_i(\tilde{x}) = 0 \ , \ i = 1,..., n_i \tag{2.16}$$

$$\tilde{\lambda}_i \ge 0 \ , \ i = 1,..., n_i \tag{2.17}$$

Condition (2.16) implies that, if the i-th inequality is not active (*i.e.* $g_i(\tilde{x}) < 0$), then the corresponding multiplier must be zero, and, conversely, if $\tilde{\lambda}_i > 0$, then the corresponding inequality must be active (*i.e.* $g_i(\tilde{x}) = 0$).
It can be proven that (2.13) is a sufficient condition for to $\tilde{x}$ be a global minimum point if f(x), g(x), h(x) are convex functions.
Despite the greatest theoretical importance of the aforementioned results, often their practical significance is modest. In computational electromagnetism, in fact, functions f, g_i and h_i are known only numerically; therefore, classical assumptions about differentiability and convexity cannot be assessed. From the computational viewpoint, the numerical approximation of the gradient is time consuming: moreover, it is a potential source of inaccuracies that could originate false steady-points when determining $\tilde{x}$ (see Section 2.7).

2.6.2 Managing constraints

Kuhn-Tucker theorem inspired a number of methods to incorporate constraints in the objective function. In fact, a simple technique to manage constraints is to transform the constrained problem into an unconstrained one, by adding a penalty term to the objective function when the design variables violate the constraints. This way, a sequence of unconstrained problems is solved, which is assumed to converge to the solution of the constrained problem.
A simple functional, which can be used when *e.g.* a set of equalities is prescribed, is the following:

$$\Phi(\mathrm{x},\lambda)=\mathrm{f}(\mathrm{x})+\frac{1}{2}\lambda[\mathrm{h}(\mathrm{x})]^{\mathrm{T}}[\mathrm{h}(\mathrm{x})] \tag{2.18}$$

where h is a column vector with entries $h_i(x)$ like in (2.10) and λ is a (known) multiplier. Intuitively, the idea is to balance the goal of reducing the objective function and staying inside the feasible region. Deriving a sequence of penalty functions as in (2.10) for increasing values of λ, it is expected that the approximated solution tends to the true solution as $\lambda \to \infty$. Traditionally, this procedure is implemented as follows:

i) initialize $\lambda_0 \in \Re^+$;

ii) choose a known sequence $\{\lambda_k\} \to \infty$;

iii) for each λ_k , find an unconstrained local solution x* minimising $\Phi(\mathrm{x},\lambda_k)$;

iv) stop when the residual $[\mathrm{h}(\mathrm{x}*)]^{\mathrm{T}}[\mathrm{h}(\mathrm{x}*)]$ is small enough.

Although the procedure described above is appealing, it suffers from numerical difficulties in practice: these are caused by the fact that, as $\lambda \to \infty$, it becomes increasingly difficult to perform the minimisation step iii) of the algorithm. Choosing a large λ_0 or a rapidly increasing sequence of λ_k gives origin to minimisation sub-problems which are difficult to solve accurately. The alternative of choosing a small λ_0 or increasing the sequence slowly makes it easier to achieve an accurate solution to sub-problems, but the procedure is not cost-effective. So, a trade-off is necessary.
In general, it can be stated that a cost-effective and accurate solution to the optimisation problem depends on the number of design variables and constraints, as well as on the properties of objective function and constraints.

2.7 Local *vs* global search

Several algorithms for both unconstrained and constrained optimisation are available; basically, they can be sorted into two broad classes (gradient-free and gradient-based methods) in terms of the derivative information which is not used or used, respectively.
Methods that use only function evaluations (zero-order methods) fall under the first class; they are suitable for problems characterized by non-linearity and discontinuities of the objective function, because the gradient might be critical to be evaluated or even not defined. The computational efficiency is low due to the repeated calls to the objective function.
In turn, gradient-based methods are more efficient than the zero-order ones: they are recommended when regular objective functions are dealt with. The simplest way to approximate the gradient of the objective function relies on finite differences; *e.g.* if $q_i(x) \equiv \frac{\partial f}{\partial x_i}$ is the i-th component of the gradient, a simple forward-difference scheme gives

$$q_i(x) \cong \frac{f(x_i + he_i) - f(x_i)}{h} \ , \ i = 1,..., n_v \tag{2.19}$$

where h is the incremental step and e_i is the unit vector along the i-th direction.

The following remarks can be put forward.

- The approximation of the gradient is expensive, since two function evaluations are needed for each gradient component
- Moreover, the appropriate choice of the incremental step implies repeated numerical experiments.
- Finally, the gradient computation might be an additional source of numerical ill-conditioning, due to round-off errors in the evaluation of both objective function and finite difference.

In the basic case of the gradient method, the minimisation trajectory follows the steepest descent: the algorithm starts from a given initial point $\bar{x}_0$, approximates the gradient of the objective function at $\bar{x}_0$, then finds the scalar $\tilde{\alpha}$ minimising the one-dimensional restriction $f_\alpha\left(\bar{x}_0 + \alpha \bar{\nabla} f\left(\bar{x}_0\right)\right)$ of the objective function $f\left(\bar{x}\right)$ along the direction of the gradient. This way, a new point $\bar{x}_1 = \bar{x}_0 + \tilde{\alpha} \bar{\nabla} f\left(\bar{x}_0\right)$ is found; the procedure is iterated until the

prescribed stopping criterion is fulfilled. This way, a zig-zag trajectory, joining x_0 to the final solution, is generated in the design space.
Higher-order methods, like Newton's method, are rarely used in practice, because they are suitable only when the Hessian matrix can be easily computed. In general, a minimiser of higher order starts from an initial point x_0 and iteratively selects a search direction in the n_v-dimensional space of the design variables, following an algorithm based on the definition of conjugate directions: vectors s_i and s_j are said to be A–conjugate if it exists a symmetric and positive-definite matrix A such that $s_i^T A s_j = 0$ with $i \neq j$; if A is the identity matrix, the usual definition of orthogonality results. Once a search direction is identified, a one-dimensional minimisation is performed to locate the point with the lowest value of the objective function, which will be the current point in the next iteration. The following pseudo-code implementing the search, can be given.

begin
select an initial design vector x_0 % initialization
initialize a set of n_v conjugate directions s_k
set $x_k = x_0$
10 while $1 \le k \le n_v$
find $\tilde{\alpha}_k$ minimising $f_{\alpha_k}(x_k + \alpha_k s_k)$ % n_v scalar minimisations
end while
if the terminating criterion is fulfilled then **stop** % convergence
else set $s_{n_v+1} = \sum_{k=1}^{n_v} \alpha_k s_k$ % search direction upgrade
$s_{k-1} = s_k \ , \ k = 2, n_v$
$s_{n_v} = s_{n_v+1}$
go to 10
end if
end

Independently of the order, all the aforementioned methods are local in a sense, because they are able to identify the closest minimum to the starting point, which is a local one unless f is convex. For this reason they are said to perform a deterministic search. To cope with these difficulties, non-deterministic minimisation algorithms, which are derivative-free and perform a stochastic search, have been developed.

2.7.1 A deterministic algorithm of lowest order: simplex method

The simplex method is based on the comparison among the cost function values at the n_v+1 vertices of a polytope (simplex), where n_v is equal to the dimension of the search space. In the case of $n_v=2$ ($n_v=3$), the polytope is a triangle (a tetrahedron).
The algorithm begins with n_v+1 points, which form the starting polytope, and the calculation of the associated objective function values. At each iteration a new polytope is set up by generating a new point to replace the worst vertex of the old polytope, *i.e.* the vertex corresponding to the highest value of objective function. Specifically, the worst vertex is replaced by its reflection with respect to the remaining n vertices. If the objective function evaluated at the new point is higher than at the worst vertex, then the new point is rejected and the vertex with the second worst value is reflected.
When it happens that a vertex belongs to the polytope longer than a given number of iterations, then the polytope is updated by contraction. The whole procedure is iterated until the diameter of the simplex is less than the specified tolerance.

2.7.2 Evolutionary computing

Darwinian evolution is intrinsically a robust search and has become the model of a class of optimisation methods for the solution of real-life problems in engineering. For the latter, the natural law of survival of the fittest in a given environment is the model to find the best design configuration fulfilling given constraints. As a matter of fact, the principle of natural evolution inspired a large family of algorithms that, through a procedure of self adaptation in an intelligent way, lead to an optimal result. A primary advantage of evolutionary computing is its conceptual simplicity: a very basic pseudo-code that describes this kind of algorithm for function optimisation is here reported:

i) initialize a population of individuals;
ii) randomly vary individuals;
iii) evaluate fitness of each individual;
iv) apply selection;
v) if the terminating criterion is fulfilled
then stop, else go to step ii).

The algorithm consists of initialization, which may be a purely random sampling of feasible solutions (step i), followed by iterative variation (step ii) and selection (step iv) based on a performance index (the fitness, step iii). This figure of merit attributes a numerical value to any feasible solution in such a way that two competing solutions can be hierarchically ranked. New solutions are generated by randomly varying existing solutions; this random variation may include mutation (as in evolution strategies) and recombination (like in genetic algorithms). Selection is applied to determine which solutions will be maintained into the next generation. Unlike deterministic methods, finer granularity in search, as gradient information, is not required. Over iterations of random variation and selection, the population can be made to converge to optimal solutions (step v).
It can be noted that the basic algorithm behind evolutionary computing is always the same. Formally, the procedure generating a new solution may be written as the difference equation

$$x(t+1) = s(v(x(t))) \tag{2.20}$$

$$x(0) = x_0 \tag{2.21}$$

where x(t) is the population at time t under a representation x, while v is an operator of random variation, and s is the selection operator. There are several possible representations, variation operators, and selection operators: in the literature, this gave rise to very many declinations of the same basic algorithm, under different names and in different contexts. The effectiveness of an evolutionary algorithm depends on the interdependence between the operators s and v applied to a given representation x of the evolving population, with initialization x_0 . In practice, this interdependence gives freedom to the designer to tailor the evolutionary approach for his/her special problem of interest. This feature gives an extra advantage over classical optimisation methods.

2.7.3 An evolution strategy of lowest order

Evolution strategy mimics the survival of the fittest individual that is observed in nature. An algorithm of the lowest order (*i.e.* a single parent generates a single offspring) is here shortly presented. The search in the design space begins in a region centred at the initial point m_0 and having radius $|d_0|$; m_0 is externally provided, while d_0 is internally calculated on the basis of the feasible-region size.

Mutating the parent configuration m means that a vector v, whose elements are characterized by a Gaussian distribution with a zero average and a standard deviation d, is added to the parent configuration itself, namely $x = m + v(0, d)$; in general, x, m, and d, as well as m_0 and d_0 , are to be considered as n_v-dimensional vectors.

The offspring configuration x is then compared with the parental configuration m in terms of objective function value, and the configuration yielding the best fitness is determined to be the parent for the next generation.

The next step is concerned with the size of the search region that will be used for the successive iteration. The underlying rationale is that when a point better than the current one is found, the radius of the search region is increased around the new point to search for further improvements; if no improvement is found, the radius of the search region is gradually decreased up to convergence (*annealing* process).

In this respect, the evolutionary algorithm substantially differs from a deterministic one, in which the search region would be narrowed around the better point in order to converge towards the corresponding, nearest minimum. On the contrary, the evolutionary algorithm, if successful in finding a better point, covers a larger region of search in order to see if there would be another good candidate in the neighbourhood, and then does the opposite when this is not believed possible. This way, there is a non-zero probability of finding the region where the global optimum of the objective function is located.

An iteration is said to be successful if x is feasible and improves the objective function This way, the history of the n_b previous iterations are used to establish a trend: if at least a fraction p of the last n_b iterations were successful, then the current trend is said to be positive, while it is negative otherwise. The *annealing* process is ruled just by the history of the minimisation procedure. If the current trend is positive, the radius $|d|$ of the search region is increased to $q^{-1}|d|$, $0 < q < 1$ and otherwise it is decreased to $q|d|$; during the first n_b iterations, d remains unchanged. The procedure

stops when the prescribed accuracy $|d_0|^{-1}|d|$ is achieved. Quantities p and q are named probability of success and rate of annealing, respectively and represent the "tuning knobs" of the algorithm; heuristic values for n_b, p and q are 50, 0.2 and $0.8 \div 0.9$, respectively.
A possible pseudo-code implementing the algorithm can be set up as follows:

```
    begin
    set "tuning knobs" values p, q, n_b
    initialize search radius d                                  % initialization
    initialize a population of feasible individuals
    take individuals as parents
10  generate a vector of Gaussian samples
    for each individual                                         % mutation + generation
        mutate the parent configuration
        generate the offspring configuration
        if the offspring is unfeasible then go to 10
        end if
    end for
    for each parent
        evaluate the objective function f_par
    end for
    for each offspring
        evaluate the objective function f_off
    end for
    if f_off < f_par then select the offspring as a new individual      % selection
        else select the parent as a new individual
    end if
    evaluate the current probability p' of success
    if p' > p then update search radius as q^-1 d                       % annealing
        else update search radius as qd
    end if
    if the terminating criterion is fulfilled then stop                 % convergence
        else go to 10
    end if
    end
```

2.7.4 No free-lunch

It is natural to ask whether there is a best evolutionary algorithm that would always give better results across the possible range of optimisation problems. In other words, the question is whether there is a choice of variation and selection operators that will always outperform all other choices regardless of the given problem. The answer is that there is no best evolutionary algorithms, and the result is known as the "no free-lunch" theorem (Wolpert and Macready, 1997).
In mathematical terms, let an algorithm a be represented as a mapping from previously-unvisited sets of points to a single previously-unvisited point ξ_k in the search space composed of all feasible points. Moreover, let $P(\xi_k|f,k,a)$ be the conditional probability of visiting point ξ_k when algorithm a is iterated k>1 times on objective function f. Then, for any pair of algorithms, a_1 and a_2 , it turns out to be:

$$\sum_f P(\xi_k|f,k,a_1) = \sum_f P(\xi_k|f,k,a_2) \tag{2.22}$$

In other words, the sum of the conditional probabilities of visiting point ξ_k is the same over all possible objective functions f, regardless the algorithm chosen (either a_1 or a_2).
Since no restrictions on the mapping operated by algorithm a_i on feasible points are assumed, it follows easily that all optimisation algorithms, both evolutionary and non-evolutionary, have identically mean performance across all possible objective functions.

Summing up, two remarks can be put forward.
i) there is no best algorithm, whether or not it is evolutionary;
ii) whatever an algorithm gains in performance on one class of problem is necessarily offset by that algorithm performance in the remaining problems.

The simple conclusion of no-free-lunch theorem has originated a great deal of controversy in the area of evolutionary computing, and some misunderstanding too. In the eighties through the nineties of the last century, there has been a considerable effort in finding the best set of operators and 'tuning knobs' of algorithms. In genetic algorithm area, for instance, these efforts have involved the probabilities of crossover and mutation operators, the representation of a population, its size and so forth. In particular, most of this research has stimulated numerical experiments on benchmark functions. However, the no-free-lunch theorem essentially states that conclusions drawn just on the basis of such trials are limited only to benchmark functions studied.

2.8 Solving inverse problems by means of rectangular systems of algebraic equations

In general, the numerical solution of field problems leads to a system of algebraic equations of the type:

$$Ax = b \tag{2.23}$$

where A is a rectangular $m \times n$ matrix, x is the unknown n-vector and b the known m-vector.
If m < n, the system is called under-determined. If, on the contrary, m > n, the system is called over-determined; the latter case is the most frequent when dealing with inverse problems, because one normally has more conditions to be fulfilled than degrees of freedom available.
Finally, if m = n, the matrix A is square. In this case, if $\det(A) \neq 0$, then A is non-singular; therefore A^{-1} exists and the corresponding system of equations has a unique solution for any b. This is the typical case when dealing with direct problems.
As far as the effect of a small perturbation of b on x is concerned, supposing m=n, let the condition number of A be defined as follows:

$$\mathrm{cond}(A) \equiv \left\| A \right\| \left\| A^{-1} \right\| = \frac{\lambda_{max}}{\lambda_{min}} \geq 1 \tag{2.24}$$

where λ_{max} and $\lambda_{min} \neq 0$ are the maximum and minimum eigenvalues of matrix A, respectively. If cond(A) is large, then the matrix is called ill-conditioned and the solution may be perturbed substantially by even a small change of b.
If A is rectangular, theoretically the inverse of A does not exist and the system of equations has no or infinite solutions. However, if m > n and the rank of A is equal to n (*i.e.* the n columns of A are linearly independent), a pseudo-inverse of A can be looked for, by means of suitable numerical techniques like least-squares or singular-value decomposition.

2.8.1 Least-squares

If A is a $m \times n$ matrix (m >n) of rank n and b is a given m-vector, then a solution to (2.23) can be found by minimising a norm, for instance the Euclidean or two-norm, of the residual Ax-b. The latter is defined as

$$r(x) = \|Ax - b\|_2^2 = x^T A^T Ax - 2x^T A^T b + b^T b \tag{2.25}$$

The gradient of the residual is

$$\overline{\nabla} r(x) = \|Ax - b\|_2^2 = 2A^T Ax - 2A^T b \tag{2.26}$$

Apparently, the residual has a unique minimum point $\tilde{x}$ such that $\overline{\nabla} r(\tilde{x}) = 0$. The so-called normal equations associated to (2.23) are obtained forcing just the latter condition, giving

$$2A^T Ax - 2A^T b = 0 \tag{2.27}$$

and, therefore,

$$A^T Ax = A^T b \tag{2.28}$$

where $A^T A$ is a square $n \times n$ matrix. It can be proven that the vector

$$x^* = \left(A^T A\right)^{-1} A^T b \tag{2.29}$$

fulfils the condition

$$\left\|Ax^* - b\right\|_2 \le \|Ax - b\|_2 \tag{2.30}$$

for each n-dimensional vector x and so x* is the least-square solution to (2.23); matrix $\left(A^T A\right)^{-1} A^T$ is called pseudo-inverse of A.

In principle, if A has full-column rank, $A^T A$ is positive definite; however, from the numerical viewpoint, solving (2.28) might fail for a twofold reason:

- the magnification of ill-conditioning when passing from A to $A^T A$, resulting in cond($A^T A$)>>1;
- the round-off errors after calculating the entries of $A^T A$.

Therefore, the use of normal equations is not recommended because it might lead to instability.

2.8.2 Singular-value decomposition

A more effective approach is given *e.g.* by the Singular Value Decomposition (SVD) method; basically, it consists of decomposing the matrix A, which is assumed to be full-column rank (m > n), into the product of three matrices, *i.e.* a $m \times m$ orthogonal matrix U, a $m \times n$ block diagonal matrix S, a $n \times n$ orthogonal matrix V, such that $A = USV^T$. In particular, it results

$$S = \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} \tag{2.31}$$

with $\Sigma = \mathrm{diag}(\sigma_1,...,\sigma_n)$. The diagonal entries of Σ are the singular values of A.
The solution to the least-square problem is then given by

$$x^* = V S^{-1} U^T b \tag{2.32}$$

with

$$S^{-1} = \begin{bmatrix} \Sigma^{-1} & 0 \\ 0 & 0 \end{bmatrix} \tag{2.33}$$

and $\Sigma^{-1} = \mathrm{diag}\left(\sigma_1^{-1},...,\sigma_n^{-1}\right)$. Also matrix $VS^{-1}U^T$ is a pseudo-inverse of A.

2.8.3 Regularization

The regularization method was proposed as a way to stabilize the solution of the Fredholm's equation of the first kind (2.1). For this class of integral equations, in fact, the solution f(y) does not depend continuously on the given function g(x). Since the output is not stable against small perturbation of the input, problem (2.1) violates the Hadamard's conditions of well-posedness (see Section 2.2).

In (2.1) the integral operator $\int \mathrm{K}(\mathrm{x},\mathrm{y})\cdot \mathrm{dy}$ can be discretized by means of a finite-difference grid composed of n nodes, while the known term g(x) can be discretized on another grid of m > n nodes. This gives rise to matrix A and vector b, approximating the integral operator and the source term, respectively.

Due to the ill-posedness of the continuous problem (2.1), also the discretized problem, *i.e.* the resulting set of linear algebraic equations (2.23), where vector x approximates function f(y), is ill-posed. Nonetheless, taking into account some *a priori* information about the solution, it is possible to convert (2.23) into a well-posed problem: for instance, if the norm of the solution x should be bounded, it makes sense to incorporate a penalty term into the problem formulation.

To this end, let the so-called Tikhonov's functional be defined as

$$\begin{aligned}\mathrm{T}(\mathrm{x}_\alpha) &\equiv \|\mathrm{A}\mathrm{x}_\alpha - \mathrm{b}\|_2^2 + \alpha\|\mathrm{x}_\alpha\|_2^2 = \\ &= \mathrm{x}_\alpha^{\mathrm{T}}\mathrm{A}^{\mathrm{T}}\mathrm{A}\mathrm{x}_\alpha - 2\mathrm{x}_\alpha^{\mathrm{T}}\mathrm{A}^{\mathrm{T}}\mathrm{b} + \mathrm{b}^{\mathrm{T}}\mathrm{b} + \alpha\mathrm{x}_\alpha^{\mathrm{T}}\mathrm{x}_\alpha\end{aligned} \tag{2.34}$$

Then, the regularization problem reads

$$\text{find} \inf_{\mathrm{x}_\alpha \in \mathrm{X}} \mathrm{T}(\mathrm{x}_\alpha) \tag{2.35}$$

Forcing the equilibrium condition $\overline{\nabla}\mathrm{T}(\mathrm{x}_\alpha) = 0$, one finds a unique minimum point; in fact, one has

$$\overline{\nabla}\left[\|\mathrm{A}\mathrm{x}-\mathrm{b}\|_2^2 + \alpha\|\mathrm{x}_\alpha\|_2^2\right] = 2\mathrm{A}^{\mathrm{T}}\mathrm{A}\mathrm{x}_\alpha - 2\mathrm{A}^{\mathrm{T}}\mathrm{b} + 2\alpha\mathrm{x}_\alpha = 0 \tag{2.36}$$

The solution $\tilde{\mathrm{x}}_\alpha$ of (2.36) is the so-called quasi-solution to problem (2.23). Therefore, $\tilde{\mathrm{x}}_\alpha$ solves the system of linear equations $\alpha\mathrm{x}_\alpha + \mathrm{A}^{\mathrm{T}}\mathrm{A}\mathrm{x}_\alpha = \mathrm{A}^{\mathrm{T}}\mathrm{b}$, or, equivalently,

$$\left(\mathrm{A}^{\mathrm{T}}\mathrm{A} + \alpha\mathrm{I}\right)\mathrm{x}_\alpha = \mathrm{A}^{\mathrm{T}}\mathrm{b} \tag{2.37}$$

The latter is the Euler's equation associated to Tikhonov's functional. If columns of the augmented matrix $A^T A + \alpha I$ are linearly independent, then the solution $\tilde{x}_\alpha$ is unique and it can be proven that it depends continuously on $A^T b$.

In other words, $\tilde{x}_\alpha$ keeps the residual $\|Ax_\alpha - b\|_2^2$ small in a stable way, which is controlled by the penalty term $\alpha\|x_\alpha\|_2^2$. As far as numerical aspects are concerned, the optimal value of the regularization parameter α is critical: if too small, the solution x_α will be oscillatory; if, on the contrary, too large, the solution will be over-smoothed.

There is another viewpoint to consider problem (2.35), *i.e.* in terms of a two-objective minimisation. In fact, the norm $\|Ax - b\|_2^2$ in the Tikhonov's functional (2.34) accounts for the agreement of the field model to the supplied data. When the norm itself is minimised, the agreement becomes very good, but the solution is unstable. That is where the second norm $\|x\|_2^2$ appearing in (2.34) comes in, in order to control the smoothness of the solution, *i.e.* its stability with respect to perturbations in the data. In turn, minimising the second norm by itself gives a very smooth solution that has nothing in common with the given data. Therefore, the trade-off curve of the best compromises between agreement and smoothness is to be sought for, by varying the regularization parameter α in a suitable way and then selecting an equilibrium point along the curve.
Likewise, in previous Section 2.6.2, functional (2.18) aiming at reducing both objective function and constraint violation was another example of two-objective minimisation. A full overview of multi-objective optimisation theory will presented in Chapter 3.