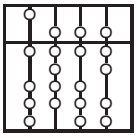
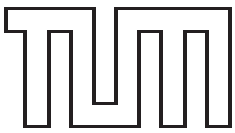


TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÄT FÜR INFORMATIK



Lehrstuhl für Effiziente Algorithmen

Design of Algorithms for Motion Planning and Motion Prediction

Dmitry Chibisov

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. A. Knoll
Prüfer der Dissertation:	
1.	Univ.-Prof. Dr. E. W. Mayr
2.	Univ.-Prof. Dr. H.-J. Bungartz

Die Dissertation wurde am 10.03.2008 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 04.12.2009 angenommen.

Abstract

This work is devoted to design of efficient algorithms for special instances of robot motion planning problems and prediction of motion of fluids. The intricate nature of these problems problems may manifest itself in enhanced computational complexity. For instance, the well-known NP- and PSPACE-hardness results for various classes of motion planning and motion optimization problems seem to imply exponential worst-case running time. Although these results characterize worst case instances, they nevertheless are indicative of the computational difficulty of the problem. The goal of this work is to develop a framework of symbolic-numerical algorithms and software packages using Computer Algebra System Maple, which would allow to increase the efficiency of solution of special motion planning and motion prediction problems and provide the possibility to develop the efficient approximation methods for computationally hard problems.

The main contribution of this work is a novel approach for motion planning problem for multiple tasks distributed in space, which have to be executed by the robot. We describe an algorithm which computes the minimum-time robot motion due to given velocity and orientation constraints of the robot end-effector during task execution, and limits on velocities and accelerations during the overall work-cycle. Given equations of robot motion in matrix form, we shall utilize the freedom in position and orientation of the robot end-effector during task execution and express the solution space for our optimization task explicitly using computation of Moore-Penrose pseudoinverses of matrices with polynomial entries. Furthermore, we discuss the usage of polynomials to speed-up algorithms for grid generation and numerical solution of two-dimensional incompressible Navier–Stokes equations.

Contents

Chapter 1

Problem Formulation and State of the Art

1.0.1 Robot Motion Planning

Many practical geometric problems for industrial applications deal with moving objects. In this section we describe the classification of motion planning problems due to [44]. The position and orientation of the geometric object to be moved in the real space may be manifested as an individual point in a configuration space, in which each coordinate represents a degree of freedom in the position or orientation of this object (see [43]). More generally, in the case of a number of distinct objects we denote with configuration space a topological space C which is spanned by all the parameters that uniquely determine the positions of the movable objects. The configurations which, due to the presence of obstacles, are forbidden to the object can be characterized as regions in the configuration space C_{obs} called *configuration space obstacles* (see Fig. 1.1).

The free space $\mathcal{C}_{free}$ remains after removing all the obstacles from the workspace

$$C_{free} = C - \bigcup_{j=1}^{M} C_{obsj}. \tag{1.1}$$

The problem to find a collision-free trajectory of an object is called the piano mover's problem:

Formulation 1.1. The Piano Mover's Problem

- A world W in which either $W = R^2$ or $W = R^3$.
- A semi-algebraic obstacle region $O \subset W$ in the world.
- A semi-algebraic robot is defined in W. It may be a rigid robot A or a collection of m links, $A_1, A_2, \ldots, A_m$.

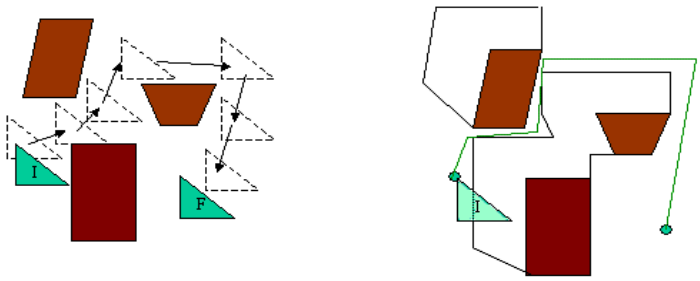


Figure 1.1: Configuration space approach - enlarging obstacles: a problem of motion planning is reduced to finding a curve (motion of an individual point) in the configuration space

- The configuration space C determined by specifying the set of all possible transformations that may be applied to the robot. From this, C_{obs} and C_{free} are derived.
- A configuration, $q_I \subset C_{free}$ designated as the initial configuration.
- A configuration $q_G \subset C_{free}$ designated as the goal configuration. The initial and goal configurations together are often called a query pair (or query) and designated as (q_I, q_G).
- A complete algorithm must compute a (continuous) path, $\pi : [0, 1] \rightarrow C_{free}$, such that $\pi(0) = q_I$ and $\pi(1) = q_G$, or correctly report that such a path does not exist.

The algorithm which solves the translational and rotational collision-free motion or safe placement problem when the objects are polygons or polyhedra was first presented in [43]. This algorithm computes configuration space obstacles using the notion of the Minkowski sum. After the configuration space obstacles have been calculated, the problem of motion planning is reduced to finding a path in the so-called *visibility graph*. In the presence of rotational motion, the induced configuration space obstacles may be represented as nonlinear constraints, which can be approximated by linear constraints. As noted in [52], the fundamental difficulty is that an exponential number of linear constraints would be required to approximate even a quadratic surface within an accuracy of 2^{-n}, resulting in an exponential time algorithm.

The exact computation of configuration space obstacles can be done with the aid of real quantifier elimination methods. The configuration space obstacles are semi-algebraic sets and the task of collision-free motion planning is then reduced to the problem of constructing a semi-algebraic curve between initial and final configurations, such that the intersection of this curve with the interior of semi-algebraic set is empty. This purely geometric problem has been solved in [60] using Cylindrical Algebraic Decomposition ([20]) of semi-algebraic sets. The latter algorithm can be performed in time polynomial in the number of polynomials as well as their maximal degree and double exponential in the number of variables. More efficient algorithms for the path calculation are presented in [3], [10]

and have single exponential bounds in the number of variables. One of disadvantages of the mentioned algorithms is that they follow the boundary of configuration space and may produce paths, which touch obstacles. To calculate paths with maximal clearance from obstacles several methods based on Voronoi diagrams have been proposed (see [10], [61]). Since these algorithms are exponential in the number of variables they can not be applied to problems with many degrees of freedom.

The following extension of the Piano Mover's problem can be formulated as follows (see [44]):

Formulation 1.2. Feedback Motion Planning

- A state space, X, which is a smooth manifold. The state space will most often be C_{free}, as defined previously.
- For each state, $x \in X$, an action space, $U(x) = T_x(X)$, where T_x is the tangent space at a point x on a manifold X. The zero velocity, $0 \in T_x(X)$, is designated as the termination action, u_T . Using this model, the robot is capable of selecting its velocity at any state.
- An unbounded time interval, $T = [0, \infty)$.
- A differential state transition equation

$$\dot{x} = u,$$

  which is expressed using a coordinate neighborhood and yields the velocity, $\dot{x}$, directly assigned by the action u. The velocity produced by u_T is $0 \in T_x(X)$, which means "stop".
- A goal set $X_G \subset X$.

The task is to compute a feedback plan, π, which is defined as a function π , which produces an action $u \in U(x)$ for each $x \in X$. A feedback plan can equivalently be considered as a vector field on X because each $u \in U(x)$ specifies a velocity vector. Further extension is motion planning under differential constraints. Motion planning under differential constraints can be considered as a variant of classical two-point boundary value problems (BVPs). Given initial and goal states, the task is to compute a path through a state space that connects initial and goal states while satisfying differential constraints ([44]).

Formulation 1.3. Motion Planning Under Differential Constraints

- A world W, a robot A (or $A_1, \ldots, A_m$ for a linkage), an obstacle region O, and a configuration space C, which are defined the same as in Formulation 1.1.
- An unbounded time interval $T = [0, \infty)]$.

- A smooth manifold X, called the state space, which may be $X = C$ or it may be a phase space derived from C if dynamics is considered;
- Let $\kappa : X \rightarrow C$ denote a function that returns the configuration $q \in C$ associated with $x \in X$. Hence, $q = \kappa(x)$.
- An obstacle region X_{obs} is defined for the state space. If $X = C$, then $X_{obs} = C_{obs}$.
- For each state $x \in X$, a bounded action space $U(x) \subseteq R^n \cup u_T$, which includes a termination action u_T and n is some fixed integer called the number of action variables. If the termination action is applied, it is assumed that $f(x, u_T) = 0$ (and no cost accumulates, if a cost functional is used). Let U denote the union of $U(x)$ over all $x \in X$.
- A system is specified using a state transition equation $x = f(x, u)$, defined for every $x \in X$ and $u \in U(x)$.
- A state $x_I \in X_{free}$ is designated as the initial state.
- A set $X_G \subset X_{free}$ is designated as the goal region.
- A complete algorithm must compute an action trajectory $u : T \rightarrow U$, for which the state trajectory satisfies:
  - $x(0) = x_I$,
  - there exists some $t > 0$ for which $u(t) = u_T$ and $x(t) \in X_G$.

The only known methods for exact planning under differential constraints in the presence of obstacles are for the double integrator system $\ddot{q} = u$, for $\mathcal{C} = \mathbb{R}$ ([48]) and $\mathcal{C} = \mathbb{R}^2$ ([11]). Powerful numerical solver for planning under differential constraints was proposed in [59] and is based on the multiple shooting method for solving two-point boundary value problems.

Lower Bounds for Motion Planning Problems

The general motion planning problem, Formulation 1.1, was shown in 1979 to be PSPACE-hard by Reif ([52]). The problem was restricted to polyhedral obstacles and a finite number of polyhedral robot bodies attached by spherical joints. The coordinates of all polyhedra are assumed to be in $\mathbb{Q}$. The proof introduces a motion planning instance with many attached robot parts that work their way through a complicated system of tunnels, which simulates the operation of a symmetric Turing machine. Canny established that the problem in Formulation 1.1 (expressed using polynomials that have rational coefficients) lies in PSPACE [10]. Therefore, the general motion planning problem is PSPACE-complete. Many other lower bounds have been shown for a variety of planning problems. One famous example is the Warehousemans problem (see [44]). This problem involves a finite number of translating, axis-aligned rectangles in a rectangular world. It was shown in [33] to be

PSPACE-hard. It was even shown that planning for Sokoban, which is a warehousemans problem on a discrete 2D grid, is also PSPACE-hard ([23]). Other general motion planning problems that were shown to be PSPACE-hard include motion planning for a chain of bodies in the plane ([32], [34]) and motion planning for a chain of bodies among polyhedral obstacles in $\mathbb{R}^3$.

Upper Bounds for Motion Planning

The first upper bound for the motion planning problem in Formulation 1.1 results from the application of cylindrical algebraic decomposition [20]. Let n be the dimension of C. Let m be the number of polynomials in F, which are used to define C_{obs}. Let d be the maximum degree among the polynomials in F. The maximum degree of the resulting polynomials is bounded by $O(d^{2^{n-1}})$, and the total number of polynomials is bounded by $O((md)^{3^{n-1}})$. The total running time required to use cylindrical algebraic decomposition for motion planning is bounded by $(md)^{O(1)^n}$. Since the general problem is PSPACE-complete, it appears unavoidable that a complete, general motion planning algorithm will require a running time that is exponential in dimension. Since cylindrical algebraic decomposition is doubly exponential, it led many in the 1980s to wonder whether this upper bound could be lowered. This was achieved by Canny's roadmap algorithm, for which the running time is bounded by $m^n(\lg m)d^{O(n^4)}$. Hence, it is singly exponential, which appears very close to optimal because it is up against the lower bound that seems to be implied by PSPACE-hardness. Another single exponential roadmap algorithm has been introduced in [3], and its running time is bounded by $m^{k+1}d^{O(n^2)}$. This is the best-known upper bound for the problems in Formulation 1.1.

1.0.2 Prediction of Motion of Fluids

In this section we discuss the methods of motion prediction for incompressible fluids. Non-stationary incompressible viscous fluids will be described by the Navier-Stokes equations. For simplicity, we shall limit our consideration to the two-dimensional case and carry out our analysis in Cartesian coordinates. Then we obtain a system of partial differential equations consisting of two momentum equations

$$\frac{\partial u}{\partial t} + \frac{\partial p}{\partial x} = \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) - \frac{\partial (u^2)}{\partial x} - \frac{\partial (uv)}{\partial y} + g_x,$$

$$\frac{\partial v}{\partial t} + \frac{\partial p}{\partial y} = \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) - \frac{\partial (uv)}{\partial x} - \frac{\partial (v^2)}{\partial y} + g_y$$

and the continuity equation

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0,$$

The quantities to be found are

- $u : \Omega \times [0, t_{end}] \to R$, the fluid velocity in x–direction,
- $v : \Omega \times [0, t_{end}] \to R$, the fluid velocity in y–direction,
- $p : \Omega \times [0, t_{end}] \to R$ the pressure.
- g_x und g_y : $\Omega \times [0, t_{end}] \to R$ denote the external forces, either the Earth gravity or other body forces acting throughout the bulk of the system and producing acceleration in its parts.

The dimensionless real quantity Re is called *the Reynolds number*; it characterizes the fluid flow. The Reynolds number depends on viscosity and on average velocity of the fluid. The lower the Re value, the more viscous is the fluid.

At the initial moment ($t = 0$), initial conditions $u = u_0(x, y)$ and $v = v_0(x, y)$ satisfying (5.3) are given. Besides, supplementary conditions holding at all four boundaries of the region for all times are required, so that we arrive at an*initial-boundary value problem.*

The numerical solution of Navier–Stokes equations is simplified greatly if they are discretized on a uniform rectangular spatial grid in Cartesian coordinates. It is natural and convenient to use such grids at the solution of problems in regions of rectangular shape. Many applied problems are, however, characterized by the presence of curved boundaries. In such cases, other grid types are often used: curvilinear grids, structured and unstructured triangular and polygonal grids. Although such grids simplify the implementation of boundary conditions, their use leads to new difficulties, such as the extra (metric) terms in equations, extra interpolations, etc. ([38]).

During the last decade, a new method for numerical solution of the Navier-Stokes equations in regions with complex geometry has enjoyed a powerful development: the immersed boundary method (IBM). In this method, the computation of gas motion is carried out on a rectangular grid, and the curved boundary is interpreted as an interface. The grid cells lying outside the region occupied by the fluid are classified as the ghost cells in which the Navier–Stokes equations are, however, also solved numerically. A survey of different recent realizations of the IBM may be found in [46, 65, 68]. The immersed boundary method has extended significantly the scope of applicability of the rectangular Cartesian grids at the numerical solution of applied problems of the incompressible fluid dynamics.

For instance, the difference scheme proposed in [37] is often used within the IBM framework. The convective terms are approximated in this scheme with the aid of the explicit three-level Adams–Bashforth second-order scheme, and the viscous terms are approximated by the implicit second-order Crank–Nicolson scheme. Despite the popularity of this and other schemes [37], its stability was not investigated even in the case of two spatial variables. As will be shown in Chapter 3 the stability investigation can be reduced to the problem of variables elimination from algebraic equations, which turns out to be a hard task even for modern computer algebra software.

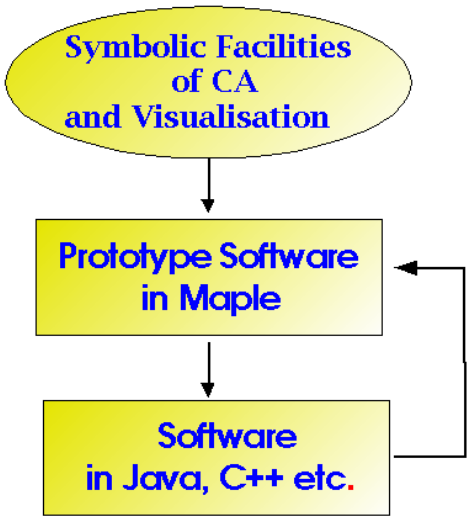


Figure 1.2: Computer Algebra for Motion Planning and Optimization

1.1 Contributions of this Work

This work is devoted to design of efficient algorithms for robot motion planning and prediction of motion of fluids. The intricate nature of these problems may manifest itself in enhanced computational complexity. For instance, the well-known NP- and PSPACE-hardness results for various classes of motion planning and motion optimization problems seem to imply exponential worst-case running time. Although these results characterize worst case instances, they nevertheless are indicative of the computational difficulty of the particular problem at hand. The goal of this work is to develop a framework of symbolic-numerical algorithms (in particular, using Computer Algebra System Maple as shown in Fig. 1.2) which would allow to increase the efficiency of solution for special motion planning and motion prediction problems, and provide the possibility to develop the efficient approximation methods for computationally hard problems.

Due to the high computational complexity of finding the exact solution of the Piano Mover's problem (Formulation 1.1), the widely used heuristic approach is based on so-called navigation functions. The construction of a scalar valued navigation function for the specification of robot tasks is a well-known problem. Given the initial and final position of a robot as well as a set of semi-algebraic obstacles, the navigation function is required to rise in the vicinity of obstacles in the direction towards them and to decrease monotonously along some path from the initial to the final position, if and only if the path does not intersect any obstacle. In this way the problem of calculation of the collision-free path can be solved in a computationally efficient manner by reduction to the task of following the gradient of the navigation function. In Chapter 2, we present a new family of analytic navigation functions and investigate their properties for a large class of geometric optimization problems ([19]).

In Chapter 3, we consider a special case of a general robot motion planning problem

(Formulations 1.2, 1.3) for multiple tasks distributed in space, which have to be executed by the robot and describe an approach for computing the minimum-time robot motion due to given velocity and constrained orientation of the robot end-effector during the task execution, and limits on velocities and accelerations during the overall work-cycle ([17], [18]). Since the velocity of task execution is given, the time needed for execution of given tasks can be reduced using the freedom in orientation of the end-effector in such a way as to reduce the motion time between tasks. Given equations of robot motion in matrix form, we shall utilize the freedom in position and orientation of the robot end-effector during the task execution and express the solution space for our optimization task explicitly using computation of Moore-Penrose pseudoinverses of matrices with polynomial entries. Finally, joint displacements for offsets between tasks are described using cubic B-Splines such that a number of well-known numerical methods can be applied to compute minimum-time B-Spline curves. The application of this approach will be applied to remote laser welding used in automotive industry in order to compute minimum-time robot trajectories.

Finally, in Chapter 4 and Chapter 5 we discuss the usage of polynomials to speed-up algorithms for grid generation and numerical solution of two-dimensional incompressible Navier–Stokes equations (see [14], [16], [15]).

Chapter 2

Navigation Functions for Piano Mover's Problem

Many practical geometric problems for industrial applications deal with placing and moving an object without colliding with nearby objects. The intricate nature of such problems manifests itself in enhanced computational complexity, whereas in industrial applications these problems must often be solved in real time. In this chapter, we consider two main types of spatial planning problems in a common framework:

- **FindSpace**: optimal placement of geometric objects, for example, maximizing the number of objects of similar shape that can be cut out from a piece of material, minimizing the quantity of material needed to produce certain shapes, various packing problems, etc. (see, for example, [22]);
- **FindPath**: finding a collision-free motion path of an object amidst some obstacles of a particular shape, for example, an automatic assembly using an industrial robot, which requires grasping objects, moving them without collisions, and ultimately bringing them together.

The position and orientation of the geometric object to be moved or placed in the real space may be manifested as an individual point in a configuration space, in which each coordinate represents a degree of freedom in the position or orientation of this object ([43]). The configurations which, due to the presence of obstacles, are forbidden to the object can be characterized as regions in the configuration space called *configuration space obstacles* (see Fig. 1.1). The algorithm which solves the translational and rotational collision-free motion or safe placement problem when the objects are polygons or polyhedra was first presented in [43]. This algorithm computes configuration space obstacles using the notion of the Minkowski sum. After the configuration space obstacles have been calculated, the problem of motion planning is reduced to finding a path in the so-called *visibility graph*. In the presence of rotational motion, the induced configuration space obstacles may be represented as nonlinear constraints, which can be approximated by linear constraints. As noted in [52], the fundamental difficulty is that an exponential number of linear constraints

would be required to approximate even a quadratic surface within an accuracy of 2^{-n}, resulting in an exponential time algorithm.

The exact computation of configuration space obstacles can be done with the aid of real quantifier elimination methods, as will be discussed in Section 2.2. The configuration space obstacles are semi-algebraic sets and the task of collision-free motion planning is then reduced to the problem of constructing a semi-algebraic curve between two points, such that the intersection of this curve with the interior of semi-algebraic set is empty. This purely geometric problem has been solved in [60] using Cylindrical Algebraic Decomposition ([20]) of semi-algebraic sets. The latter algorithm can be performed in time polynomial in the number of polynomials as well as their maximal degree and double exponential in the number of variables. More efficient algorithms for the path calculation are presented in [3], [10] and have single exponential bounds in the number of variables. One of disadvantages of the mentioned algorithms is that they follow the boundary of configuration space and may produce paths, which touch obstacles. To calculate paths with maximal clearance from obstacles several methods based on Voronoi diagrams have been proposed (see [10], [61]).

In contrast to all these approaches, the objective of the present work is the generalization of finding a geometric path in order to

- find paths, which guarantee a certain minimum clearance from obstacles,
- provide the possibility to incorporate nonholonomic motion constraints (velocities, acceleration, etc.).

For this purpose, we shall describe a family of analytic functions with the property to rise in the vicinity of obstacles of arbitrary shape in the direction towards them. Using such "obstacle functions" (sometimes called "distance function"), we shall show how a "goal function" (sometimes called "target function") can be constructed, which decreases monotonously along some path from the initial to the final position, if and only if the path does not intersect any obstacle. Combining obstacle and goal function, we shall obtain a scalar-valued "navigation function" such that the problem of motion planning can be reduced to the task of following the gradient of the navigation function.

To our knowledge, the idea of using scalar valued functions for the obstacle avoidance was pioneered in [36]. The author proposed the navigation functions for the case the obstacles are a parallelepiped, a finite cylinder, and a cone. However, these geometric primitives do not form a sufficient set to describe the images of obstacles in the configuration space. The first construction of a general analytic navigation function is due to [54]. The authors show how a smooth navigation function can be constructed for the case when obstacles are smooth manifolds. In this chapter, we describe the construction of a more general family of navigation functions for arbitrary semi-algebraic objects. For this purpose, we shall use the functional representation of semi-algebraic point sets defined by so-called R-functions ([56], [62]) and reduce the problem of path finding to the solution of the Newton's equations of motion in a field of forces that can be done numerically. The obstacle and goal

functions play the role of repulsive and attractive forces that push the object away from obstacles and pull it towards the goal position. As will be shown, the R-functions exhibit a wide range of differential properties, which can be used for the purpose of nonholonomic motion control. The implementation of our approach and computational examples will be presented.

2.1 Construction of the Navigation Function

The theory of R-functions ([56],[62]) provides the methodology of constructing an implicit functional representation for any semi-algebraic set using logical (set-theoretical) operations. In this section we shall briefly introduce this concept and some results from the theory of R-functions, which shall be used in Section 2.3 for the purpose of the collision-free motion planning.

Let $F(X_1, ..., X_n)$ be a Boolean function with truth value 1 and false value 0 built using logical operations $\wedge$, $\vee$ and $\neg$. A real valued function $f(x_1, ..., x_n)$ is called an R-function if its sign is completely determined by the signs of its arguments. More precisely, f is an R-function if there exists a Boolean function F such that

$$sign(f(x_1, ..., x_n)) = F(sign(X_1), ..., sign(X_n)). \tag{2.1}$$

In other words, f works as a Boolean switching function, changing its sign only when its arguments change their signs. For example, logical operations on Boolean variables X_1, X_2 may be performed on real-valued variables x_1, x_2 such that (2.1) is satisfied using the following rules:

$$\begin{array}{l} x_1 \wedge \ x_2 \equiv x_1 + x_2 - \sqrt{x_1^2 + x_2^2} \\ x_1 \vee x_2 \equiv x_1 + x_2 + \sqrt{x_1^2 + x_2^2} \\ \neg x_1 \equiv -x_1. \end{array} \tag{2.2}$$

Consider, e.g., the Boolean function defined by

$$F(X_1, X_2, X_3, X_4) = X_1 \wedge X_2 \wedge X_3 \wedge X_4 \wedge X_5.$$

The corresponding real valued function f may be defined recursively according to (2.2):

$$\begin{array}{l} f_1(x_1, x_2) = x_1 + x_2 - \sqrt{x_1^2 + x_2^2} \\ f_2(x_3, x_4) = x_3 + x_4 - \sqrt{x_3^2 + x_4^2} \\ f(x_1, x_2, x_3, x_4, x_5) = f_1 + f_2 - \sqrt{{f_1}^2 + {f_2}^2} + x_5 - \sqrt{\left(f_1 + f_2 - \sqrt{{f_1}^2 + {f_2}^2}\right)^2 + {x_5}^2} \end{array} \tag{2.3}$$

This R-function can be used to describe point sets bounded by four arbitrary polynomials:

$$R(x, y) = \{(x, y) | \phi_1(x, y) \geq 0 \wedge \phi_2(x, y) \geq 0 \wedge \phi_3(x, y) \geq 0 \wedge \phi_4(x, y) \geq 0 \wedge \phi_5(x, y) \geq 0\} \tag{2.4}$$

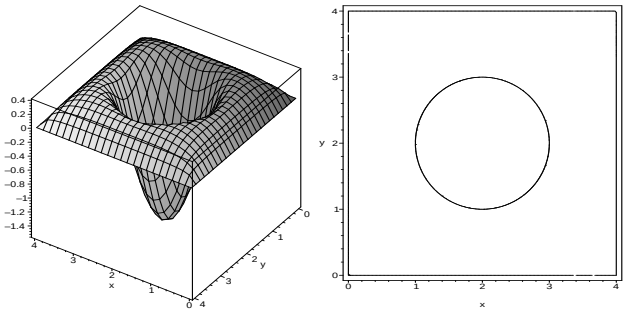


Figure 2.1: O(x,y) and its roots.

For example, let four lines in the plane be given by the roots of the polynomials ϕ_i, $i = 1...4$,

$$\begin{aligned}\phi_1(x,y) &= x\\ \phi_2(x,y) &= x-4\\ \phi_3(x,y) &= y\\ \phi_4(x,y) &= y-4\end{aligned}$$

and a circle be given by

$$\phi_5(x,y) = (x-2)^2 + (y-2)^2 - 1.$$

The object shown in Fig. 2.1 can be described as semi-algebraic set (2.4) or, alternatively, with the help of analytic function $f(\phi_1(x,y), \phi_2(x,y), \phi_3(x,y), \phi_4(x,y), \phi_5(x,y))$, that is equal to zero on the boundary of the object, positive inside and negative outside the object. In this way, any complicated semi-algebraic object can be constructed from primitive algebraic objects. Thus, R-functions enable one to write easily an equation for an object of arbitrary shape in the same way as one forms geometric objects by logical or set theoretic operations ([53]). Therefore R-functions are helpful in describing complicated semi-algebraic objects as an analytic function having the following sign property

$$\begin{aligned}f(\mathbf{x}) &> 0 \quad \text{if } \mathbf{x} \text{ is inside the object}\\ f(\mathbf{x}) &= 0 \quad \text{if } \mathbf{x} \text{ is on the boundary of the object}\\ f(\mathbf{x}) &< 0 \quad \text{if } \mathbf{x} \text{ is outside the object}\end{aligned}$$

Alternatively to (2.2), the following rules described in [56] can be used to form union and intersection of geometric objects:

$$R_\alpha : \frac{1}{1+\alpha}(x_1 + x_2 \pm \sqrt{x_1^2 + x_2^2 - 2\alpha x_1 x_2}),$$

where $\alpha(x_1, x_2)$ is an arbitrary symmetric function such that $-1 < \alpha(x_1, x_2) < 1$.

$$R_0^m : (x_1 + x_2 \pm \sqrt{x_1^2 + x_2^2})(x_1^2 + x_2^2)^{\frac{m}{2}},$$

where m is any even positive integer.

$$R_p : x_1 + x_2 \pm (x_1^p + x_2^p)^{(\frac{1}{p})},$$

for any even positive integer p.

In each case above, choosing the $+/-$ sign determines the type of an R-function: $(+)$ corresponds to R-disjunction and $(-)$ sign gives the R-conjunction. The given families of R-functions exhibit a wide range of differential properties, which are studied in [63]. The change of parameters α, m and p leads to different characteristics of the navigation function, which will be described in Section 2.4 and allows to control the velocity or acceleration of the object.

The following theorem about the derivative of R_α-functions at the boundary has been proven in [56]. It states that the absolute value of the derivative of an R-function at the boundary point $\mathbf{p}$ in the given vector direction is equal to the absolute value of the derivative of the polynomial ϕ_i, which describes this part of boundary, provided the boundary part ϕ_i does not intersect with any other boundary boundary part ϕ_j in $\mathbf{p}$. The sign of the derivative is determined by the number of logical negations of x_i, called inversion degree.

Theorem 1 (Rvachev [56], [62]) *Let $f(x_1, ..., x_N)$ be such R_α-function that argument x_i appears in f only once and has the inversion degree m. Suppose the functions $\phi_1, ..., \phi_N$ and f are continuously differentiable and satisfy the following condition at point $\mathbf{p}$:*

$$\phi_i(\mathbf{p}) = 0; \phi_j(\mathbf{p}) \neq 0, i \neq j;$$
$$f(\phi_1, ..., \phi_N)|_{\mathbf{p}} = 0.$$

Then, for any vector direction l, the following equality holds

$$\left.\frac{\partial f(\phi_1, ..., \phi_N)}{\partial l}\right|_{\mathbf{p}} = (-1)^m \left.\left(\frac{\partial \phi_i}{\partial l}\right)\right|_{\mathbf{p}.}$$

For example, for any point $\mathbf{p}$ on the boundary part ϕ_i, $i = 1...5$, shown in Fig. 2.1, the following condition is satisfied

$$\left.\frac{\partial f(\phi_1, ..., \phi_5)}{\partial l}\right|_{\mathbf{p}} = \left.\left(\frac{\partial \phi_i}{\partial l}\right)\right|_{\mathbf{p}.}$$

This condition allows one to use the gradient of R-functions to predict the presence of obstacles and avoid collisions, as will be described in Section 2.4.

2.2 Computing Configuration Space Obstacles

As mentioned above, an important part in our approach to motion planning is a configuration space method ([43]). We propose to use the following two solutions:

- exact computation of configuration space obstacles based on quantifier elimination methods ([41]);
- approximation of configuration space obstacles by nonlinear constraints, which can be calculated in a more efficient manner ([49]).

In the following paragraphs we shall briefly describe both approaches.

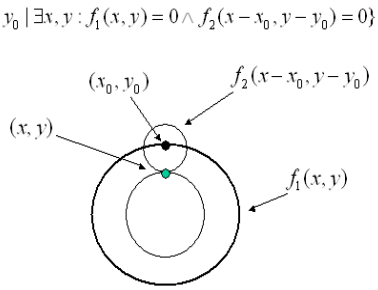


Figure 2.2: Calculation of configuration space obstacles by quantifier elimination

Exact computation of configuration space obstacles This algorithmic problem can be formulated as a decision problem for the first-order theory of real fields. The real numbers constitute an ordered field, which is closed under addition and multiplication. The formulas in the first-order theory of reals, defined by A. Tarski in 1930 and called the Tarski formulas, are composed from equalities and inequalities. Such formulas may be constructed by introducing logical connectives (conjunction, disjunction and negation) and the universal and existential quantifiers to the atomic formulas.

For example, let some geometric object representing obstacle O be bounded by roots of finitely many polynomials $O_{i,j}(x, y, z)$. The inequalities $O_{i,j}(x, y, z) \geq 0$ and $O_{i,j}(x, y, z) \leq 0$ can be used to describe the exterior and interior of the object. Choosing the suitable sign of the polynomials we may use only "$\geq$" to describe any geometric object. The Tarski formula describing the set of points, which belong to the object, can be written as follows:

$$O(x, y, z) \equiv \bigvee_i \bigwedge_j O_{i,j}(x, y, z) \geq 0$$

The object P to be moved is given by roots of polynomials $P_i(x, y, z)$ and can be described with a Tarski formula in the same way. A shift of the object by x_0, y_0, z_0 units can be written as:

$$P(x, y, z) \equiv \bigvee_i \bigwedge_j P_{i,j}(x - x_0, y - y_0, z - z_0) \geq 0$$

As can be seen in Fig. 1.1 and 2.2, in the two-dimensional case the configuration space obstacle corresponding to O can be calculated for a particular orientation of P by contacting P with O and moving P along the boundary of O keeping them in contact. The resulting geometric object is the configuration space obstacle O_ϕ^{Conf} corresponding to the orientation ϕ of P.

The contact of O and P can be expressed in terms of common roots of bounding polynomials. Thus, O_ϕ^{Conf} corresponds to such shifts x_0, y_0, z_0 of P where some of polynomials P_i and O_i have common roots. This can be formalized with a Tarski sentence as follows:

$$\{(x_0, y_0, z_0) | \exists x, y, z : P(x - x_0, y - y_0, z - z_0) = 0 \wedge O(x, y, z) = 0\}$$

Eliminating ∃-quantifiers with existing methods ([12]) produces the semi-algebraic set that corresponds to O_ϕ^{Conf}. The latter quantity can also be described with the aid of R-functions, as explained in Section 2.2. In Section 2.4 we shall show how such description can be used to predict collisions.

Approximate computation of configuration space obstacles The configuration space obstacles can be calculated using the notion of the Minkowski sum. An algorithm for the approximation of the Minkowski sum with the help of R-functions has been proposed in [49]. Suppose P and O are defined by R-functions $P(x_1, ..., x_d) \geq 0$ and $O(x_1, ..., x_d) \geq 0$, respectively. The intersection of P shifted by $s_1, ..., s_d$ units and O can be written as

$$F(x_1, ..., x_d, s_1, ..., s_d) = P(x_1 - s_1, ..., x_d - s_d) \wedge O(x_1, ..., x_d)$$

As explained above, O_ϕ^{Conf} consists exactly of such shifts $s_1, ..., s_d$, which produce the contact between P and O. The contact of P and O means that their intersection is not empty. In this case $F \geq 0$, otherwise, if P does not touch O, $F < 0$.

Thus, the projection of $F(x_1, ..., x_d, s_1, ..., s_d)$ must be calculated: find such $s_1, ..., s_d$ so that there exist some $x_1, ..., x_d$ with $F(x_1, ..., x_d, s_1, ..., s_d) \geq 0$. As shown in [49], this projection can be computed by solving the following maximization problem:

$$O_\phi^{Conf}(s_1, ..., s_d) = \mathbf{max}\{F_3(x_1, ..., x_d, s_1, ..., s_d)\}.$$

The necessary condition for a point $(x_1, ..., x_{2d})$ where the maximum is attained:

$$\frac{\partial F_3}{\partial s_i} = 0, i = 1...d;$$

These equations can be solved numerically, for example, with the help of Newton's method. In this manner, the configuration space obstacles can be represented as R-functions and used to predict collisions with obstacles.

2.3 Navigation in the Configuration Space

As mentioned above, the calculated configuration space obstacles can be represented with the help of R-functions. It follows from Theorem 1 that in the vicinity of obstacles the R-function increases towards them (see Fig. 2.3). Such "obstacle function" is therefore useful in order to predict collisions and determine the direction of the motion in order to avoid obstacles. Apart from the "obstacle function" O, we introduce the "goal function" G, which is decreasing monotonously along the path π that connects the initial position $\mathbf{s} = (s_1, ..., s_N)$ and the target position $\mathbf{g} = (g_1, ..., g_N)$. The goal function is required to have only one minimum value in $\mathbf{g}$. As we shall describe below, the sum of both functions defines the potential field U, which is used for motion planning (see Fig. 2.4):

$$U(x_1, ..., x_N) = O(x_1, ..., x_N) + G(x_1, ..., x_N). \tag{2.5}$$

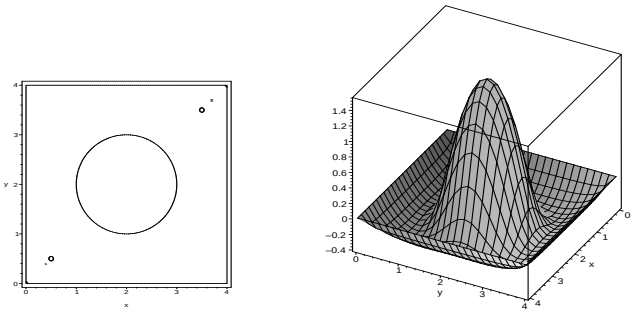


Figure 2.3: The region with obstacles (colored black) and its obstacle function. The path from s to g must be calculated.

Different functions with only one minimum value in the goal position and different differential properties can be used. In general, the following conditions must be satisfied:

- G is decreasing monotonously along the shortest path π connecting $(s_1, ..., s_N)$ with $(g_1, ..., g_N)$, e.g. the sign of the derivatives should be constant:

$$sign\left(\left.\frac{\partial G(x_1, ..., x_N)}{\partial x_i}\right|_{\pi}\right) = const. \tag{2.6}$$

- U is decreasing monotonously in all points $\mathbf{p} \in \pi$, which lie not too close to any obstacle ($|O(\mathbf{p})| < \epsilon$). From (2.5), (2.6) and from the fact that $sign(\frac{\partial O}{\partial x_i}) \neq const$, it follows that the derivatives of G should be greater than those of O:

$$|O(\mathbf{p})| < \epsilon \Leftrightarrow \left|\left.\frac{\partial G(x_1, ..., x_N)}{\partial x_i}\right|_{\mathbf{p}}\right| > \left|\left.\frac{\partial O(x_1, ..., x_N)}{\partial x_i}\right|_{\mathbf{p}}\right| \tag{2.7}$$

- U has a minimum value in some point $\mathbf{p} \in \pi$ in the vicinity of an obstacle ($|O(\mathbf{p})| \geq \epsilon$) and increases towards the obstacle:

$$|O(\mathbf{p})| \geq \epsilon \Leftrightarrow \left|\left.\frac{\partial G(x_1, ..., x_N)}{\partial x_i}\right|_{\mathbf{p}}\right| \leq \left|\left.\frac{\partial O(x_1, ..., x_N)}{\partial x_i}\right|_{\mathbf{p}}\right| \tag{2.8}$$

The following functions, which have only one minimum value in the goal position $\mathbf{g}$, can be used as goal functions:

$$G_0(x_1, ..., x_N) = \alpha(\epsilon)\sqrt{|g_1 - x_1| + ... + |g_N - x_N|},$$

$$G_d(x_1, ..., x_N) = \alpha(\epsilon)((g_1 - x_1)^{2d} + ... + (g_N - x_N)^{2d})^{\frac{1}{2d}},$$

where d and $\alpha(\epsilon)$ are the parameters to be chosen in order to satisfy the conditions (2.6)-(2.8). Using this function, the potential field U can be constructed according to (2.5).

The collision-free path from the initial to the final position corresponds to the direction of the gradient of U. In other words, we must simply follow the gradient of U. In this way, the purely geometric problem of path calculation can be reduced to the physical problem formulated with the help of the Newton's equations, which describe the motion of an object in the field of some forces $\mathbf{F}$:

$$m\mathbf{a} + \lambda\mathbf{v} = \mathbf{F},$$

where m is a mass of the object to be moved, $\mathbf{a}$ and $\mathbf{v}$ are acceleration and velocity, respectively, and λ is a so-called dissipation coefficient. Large values of λ correspond to the motion in a highly viscous environment. To describe the motion we may write the following differential equation:

$$m\frac{d^2x_i(t)}{dt^2} + \lambda\frac{dx_i(t)}{dt} = -\frac{\partial U(x_1, ..., x_d)}{\partial x_i} \tag{2.9}$$

(for simplicity, we do not consider curvilinear coordinates here). The force due to the environment "resistance" in our model is taken to be $\mathbf{R} = -\lambda\mathbf{v}$. However, other models, in particular those that account for the resistance increasing with velocity, can also be formulated, e.g. $\mathbf{R} = -C|\mathbf{v}|\mathbf{v}$ or, in the component form, $R^j = -(Cg_{ik}\dot{x}^i\dot{x}^k)^{\frac{1}{2}}\dot{x}^j$. Here g_{ik} is a metric tensor and C is the drag coefficient, which in general depends on the object's geometry and on the Reynolds number. The first term in (2.9) corresponds to the inertial motion. In our primary example, we assume this term to be small as compared to the dissipative term, which impedes the object's when the object approaches the obstacle. This is justifiable when the inertia coefficient m is small compared to $\lambda\tau_0$ where τ_0 is the characteristic time of object motion. The equations

$$\lambda\frac{dx_i(t)}{dt} = -\frac{\partial U(x_1, ..., x_d)}{\partial x_i} \tag{2.10}$$

can be solved numerically, e.g. using the finite difference techniques. Numerical methods of solution of the motion equations are mostly based on evaluating the fist derivatives as

$$\frac{df(x)}{dx} = \frac{f(x + \Delta x) - x(x)}{\Delta x} + O(\Delta x).$$

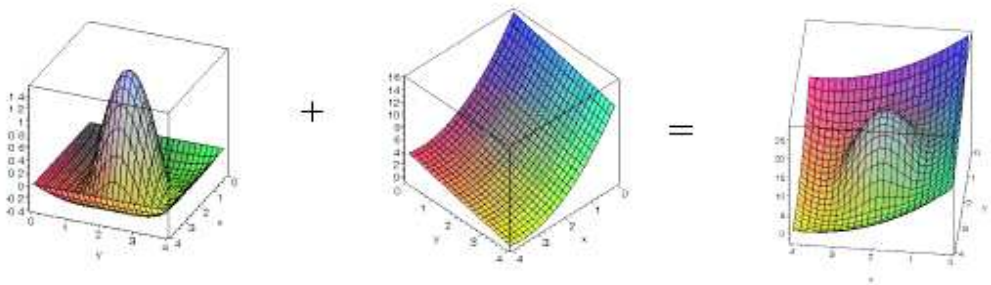


Figure 2.4: Addition of the obstacle and the goal functions

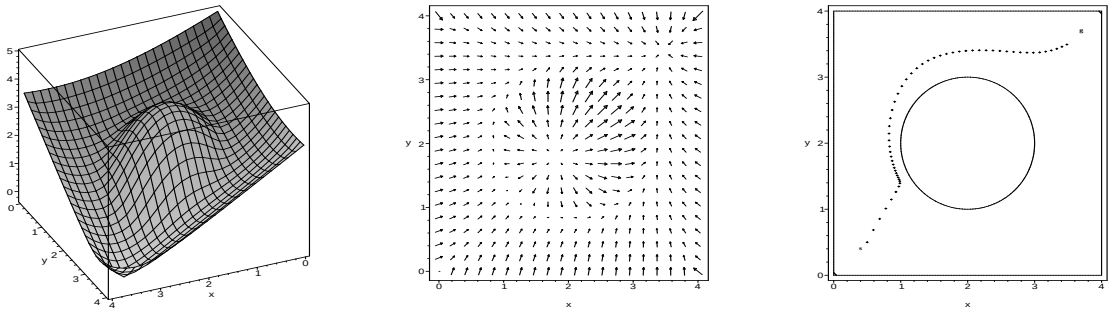


Figure 2.5: The potential field (left), its gradient field (in the middle) and the path (right) calculated by following the gradient according to (2.12). $\frac{\Delta t}{\Delta x} = 0.1$; number of time steps: 54; computational time (using Maple): 0.121 sec

Such discretization of (2.10) leads to

$$\lambda x_i(t_{j+1}) = \lambda x_i(t_j) - \frac{\Delta t}{\Delta x_i}(U(x_1, ..., x_i + \Delta x_i, ..., x_n) - U(x_1, ..., x_i, ..., x_n)). \tag{2.11}$$

According to (2.11), the object position $x(t_{j+1})$ at the time step t_{j+1} can be calculated from the previous position x_i at the time step t_j and the approximation of the gradient of U at $x(t_j)$. Initial values $x_i(0)$ designate the initial positions of an object. Solving the equations

$$\begin{aligned} x_{j+1} &= \lambda x_j - \frac{\Delta t}{\Delta x}(U(x_j + \Delta x, y_j) - U(x_j, y_j)) \\ y_{j+1} &= \lambda y_j - \frac{\Delta t}{\Delta y}(U(x_j, y_j + \Delta y) - U(x_j, y_j)) \end{aligned} \tag{2.12}$$

leads to the motion shown in Fig. 2.5, 2.6. An example with three degrees of freedom demonstrated in Fig. 2.7 can be produced by solving

$$\begin{aligned} x_{j+1} &= \lambda x_j - \frac{\Delta t}{\Delta x}(U(x_j + \Delta x, y_j, \phi_j) - U(x_j, y_j, \phi_j)) \\ y_{j+1} &= \lambda y_j - \frac{\Delta t}{\Delta y}(U(x_j, y_j + \Delta y, \phi_j) - U(x_j, y_j, \phi_j)) \\ \phi_{j+1} &= \lambda \phi_j - \frac{\Delta t}{\Delta \phi}(U(x_j, y_j, \phi_j + \Delta \phi) - U(x_j, y_j, \phi_j)). \end{aligned}$$

2.4 Conclusion

Our approach to spatial planning and associated geometrical problems presented in this chapter is based on the object motion representation in a configuration space, which has

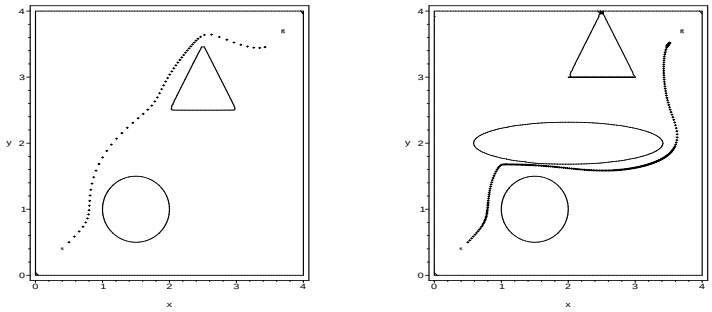


Figure 2.6: Examples of calculated paths from the initial to final positions. Left: $\frac{\Delta t}{\Delta x} = 0.1$; number of time steps: 60; computational time (using Maple): 0.251 sec. Right:$\frac{\Delta t}{\Delta x} = 0.035$; number of steps: 300; computational time (using Maple): 1.562 sec.

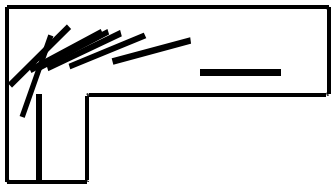


Figure 2.7: An example of calculated path with three degrees of freedom: x and y translations and rotation.

a dimensionality equal to the number of independent coordinates describing the object position and orientation in the real space. The advantage of such a method is due to the fact that in the configuration space an object's motion corresponds to that of a fictitious material point moving in a potential field combined with viscous (dissipative) forces. This allows one to employ powerful numerical algorithms to compute collision-free trajectories. The potential field configuration is defined with the help of R-function techniques, which seems to be a convenient method for the functional (analytical) representation of complex geometries. The potential force field defined by R-functions has an attractive and a repulsive part whose competition determines the goal function and the obstacle function, respectively. As it is typical of such situations, certain extremal properties arise defining the optimal path. The future work will be devoted to the extremal properties of the obstacle and goal functions. Possible applications of presented techniques, apart from robot motion planning, may include medical kinesiology, biomechanics of human motion, rendering of human body positions, velocities and accelerations, joint simulations - all being modeled with the help of motion equations.

Chapter 3

Motion Planning for Multiple Tasks

In this chapter we consider the kinematic properties of robots with 6 rotational joints and describe an approach for optimization of robot motion due to given geometric and differential constraints (i.e. constraints on position and orientation of the robot end-effector during the execution of some tasks, and limits on velocities and accelerations during the overall workcycle respectively). Several methods for control of robotic manipulators and for solving optimal point-to-point-trajectory problems have been suggested in literature, e.g. in [13], [45], [59], to cite only a few of many papers. However, these methods either do not allow to restrict the resulting motion with respect to both geometric and differential constraints at the same time or become very inefficient. The approach presented in our paper [18] overcomes these limitations by utilizing the particular kinematic model of a robot at hand as well as the particular form of geometric constraints, which occur, for example, in laser welding. In the present chapter several important extensions and improvements of this approach will be presented. We consider the non-academic, highly nonlinear model of a commercially available robot (KUKA robot with 6 rotation joints, see [40]) and discuss several objectives for optimal motion. Given equations of robot motion in matrix form, we shall utilize the freedom in position and orientation of the robot end-effector during task execution and express the solution space for our optimization task explicitly using computation of Moore-Penrose pseudoinverses.

As an application we consider laser welding for automotive industry. Laser welding reduces the overall workcycle time needed per car significantly and is of great economic importance. However, remote welding using lasers leads to new requirements for motion planning. In order to reduce workcycle time the optimal trajectories must be calculated. Let us describe the technical conditions on laser welding in more detail and state the optimization problem. Consider a number of welding seams distributed on a surface S in the Cartesian space. Every seam can be welded from the given distance d. However, the angle between the normal to S, called $\mathbf{n}_s$, and the laser beam is constrained by ϕ_{max}. Thus, for every welding point $(x, y, z) \in S$ the end-effector of the robot can lie on a part of a sphere with the radius d centered in the point (x, y, z). The admissible part of a sphere lies inside the cone given by the vertex (x, y, z) and the angle ϕ_{max}. Let $\theta = \langle \theta_1, \cdots, \theta_6 \rangle$ denote angles at the joints of the robot. The angles and velocities of individual joints are

Figure 3.1: Industrial robot KUKA KR.

constrained by some constants

$$\begin{aligned} |\theta_i| &\leq \theta_i^{max} \\ \left|\dot{\theta}_i\right| &\leq \dot{\theta}_i^{max} \\ \left|\ddot{\theta}_i\right| &\leq \ddot{\theta}_i^{max} \end{aligned} \tag{3.1}$$

The optimization problem is to find a motion of robot joints $\theta_1(t), \ldots, \theta_6(t)$ such that

- $\theta_1(t), \ldots, \theta_6(t)$ are at least C^2,
- prescribed velocity v of the laser focus as well as the welding angle condition $\phi \leq \phi_{max}$ is satisfied for every seam,
- constraints (3.1) are satisfied both for seams and offsets,
- the total welding time is minimized.

In the following we shall assume the prescribed order for welding of individual seams and will consider optimal motion subproblems for seams and offsets separately.

Since the welding velocity is prescribed for individual seams, the minimum-time trajectories may be calculated using the freedom in the orientation of laser beam during welding in such a way as to produce minimum-time trajectories in offsets between seams. In this way the problem can be reduced to:

- calculation of orientations of the laser beam at the end points of seams (Fig. 3.2),
- calculation of velocities of individual joints at the end points of seams (Fig. 3.3),

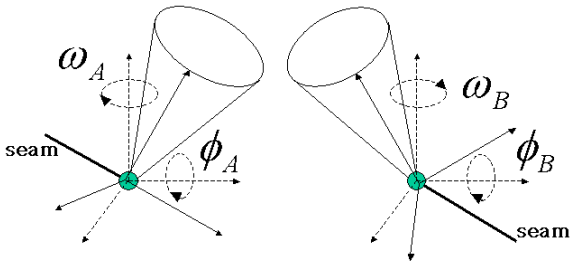


Figure 3.2: The optimal path problem for laser welding: calculate admissible orientations of laser beam such that the total time is minimized.

such that total time is minimized. As shown in Fig. 3.2, orientation of the laser beam can be described by angles ω and ϕ ($|\omega| \leq \phi_{max}$,$|\phi| \leq \phi_{max}$). Consider admissible orientations A and B at the end points of an offset. Let $\theta_1^A, \ldots, \theta_6^A$ and $\theta_1^B, \ldots, \theta_6^B$ denote the joint angles of the robot in positions corresponding to orientations A and B, respectively. The calculation of time optimal motion can be reduced to calculation of the appropriate

- orientations A and B for every offset, which determine $\theta_1^A, \ldots, \theta_6^A$, $\theta_1^B, \ldots, \theta_6^B$;
- velocities of individual joints at the end points of offsets ($\dot{\theta}_1^A, \ldots, \dot{\theta}_6^A$, $\dot{\theta}_1^B, \ldots, \dot{\theta}_6^B$) (see Fig. 3.3).

Since the inverse kinematics solution is known, the admissible orientations A and B, which minimize motion time can be found by a descent gradient method as will be shown in Section 3.4. In order to determine the admissible velocity space at end points of offsets ($\dot{\theta}_1^A, \ldots, \dot{\theta}_6^A$, $\dot{\theta}_1^B, \ldots, \dot{\theta}_6^B$) we shall describe tbe computation of the Moore-Penrose pseudoinverses of matrices with polynomial entries (see Section 3.3). The admissible velocity space is determined by constraints on velocity of the laser focus. If the location of the laser focus at the end points is specified as a vector $\mathbf{x} = \langle x, y, z \rangle$ then the kinematic equation can be written as

$$\mathbf{x} = \mathbf{F}(\theta), \tag{3.2}$$

where $\mathbf{F}$ is a smooth vector function. One of the popular techniques for controlling a manipulator is resolved motion rate control which calculates the joint velocities from the joint configuration and desired laser focus velocity. The underlying equation is the Jacobian equation which can be found by differentiating (3.2) to obtain

$$\dot{\mathbf{x}} = \mathbf{J}(\dot{\theta}), \tag{3.3}$$

where $\dot{\mathbf{x}}$ is the desired laser focus velocity. We shall consider the problem to move the laser focus along the given seam with the prescribed laser focus velocity $v = \sqrt{\dot{x}^2 + \dot{y}^2 + \dot{z}^2}$ by finding the controls $\dot{\theta}_1, \cdots, \dot{\theta}_6$ in (3.4), which steer the initial position of the laser focus at the begin of the seam (x_I, y_I, z_I) to the final position at its end (x_F, y_F, z_F).

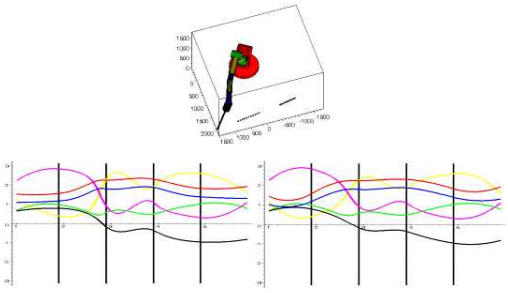


Figure 3.3: Changing velocities at the end points of offsets may increase or decrease the motion time.

Since in our case there is an infinite number of such control strategies, which correspond to different welding orientations one can take advantage of this freedom by choosing a control strategy which will provide the C^2 continuity of joint motions and will reduce the time required for motion between individual seams. The variety of all admissible controls satisfying prescribed welding velocity can be expressed explicitly using the generalized inverse strategies of the form

$$\dot{\theta} = \mathbf{J}^{+}(\dot{\mathbf{x}}),$$

where $\mathbf{J}^{+}$ is the Moore-Penrose pseudoinverse of $\mathbf{J}$ (see [4]).

3.1 Solving Kinematic Equations for Tasks with Constrained Orientation of the End-Effector

In this chapter we use Denavit-Hartenberg formalism ([24]) to model a 6R manipulator. Each robot link is represented by the line along its joint axis and the common normal to the next joint axis. In the case of parallel joints any of the common normals can be chosen. The links of the 6R manipulators numbered from 1 to 7. The base link is 1, and the outermost link or hand is 7. A coordinate system is attached to each link for describing the relative arrangements among the various links. The coordinate system attached to the ith link is numbered i. The 4x4 transformation matrix relating $i+1$ coordinate system to i coordinate system is given by (see [1] for more details):

$$\mathbf{R}_i = \begin{bmatrix} c_i & -s_i\lambda_i & s_i\mu_i & a_ic_i \\ s_i & c_i\lambda_i & c_i\mu_i & a_is_i \\ 0 & \mu_i & \lambda_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where $s_i = sin(\theta_i)$, $c_i = cos(\theta_i)$, θ_i is the ith joint rotation angle. $\mu_i = sin(\alpha_i)$, $\lambda_i = cos(\alpha_i)$, α_i is the ith twist angle, a_i is the length of link $i+1$, d_i is the offset distance at joint i.

The parameters α_i, a_i, d_i determine the particular kinematic of a 6R robot. Once these parameters are given the shifted and rotated coordinate system attached to the robot end-effector can be calculated for certain joint angles θ_i, $i = 1...6$ as a matrix product $\mathbf{R}_1\mathbf{R}_2\mathbf{R}_3\mathbf{R}_4\mathbf{R}_5\mathbf{R}_6$. The spatial coordinates of the end-effector $\mathbf{x}^{ee}$ can be computed by

$$\mathbf{x}^{ee} = \mathbf{R}_1\mathbf{R}_2\mathbf{R}_3\mathbf{R}_4\mathbf{R}_5\mathbf{R}_6\mathbf{v}, \tag{3.4}$$

where $\mathbf{v} = [0, 0, 0, 1]^T$, $\mathbf{x}^{ee} = [x, y, z, 1]^T$.

For the case of laser welding, described in the previous section, let us include the transformation $\mathbf{R}_l$ from the 6th link to the welding point in the kinematic equations (3.4). The complete homogeneous transformation, denoted by T_0^7, relates the original coordinate systems and the coordinate system attached to the welding point:

$$T_0^7 = \mathbf{R}_1\mathbf{R}_2\mathbf{R}_3\mathbf{R}_4\mathbf{R}_5\mathbf{R}_6\mathbf{R}_l = \begin{bmatrix} k_1 & l_1 & m_1 & x \\ k_2 & l_2 & m_2 & y \\ k_3 & l_3 & m_3 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.5}$$

where $\mathbf{k} = [k_1, k_2, k_3]^T$, $\mathbf{l} = [l_1, l_2, l_3]^T$, and $\mathbf{m} = [m_1, m_2, m_3]^T$ are the orthonormal basis of the transformed coordinate system, and $[x, y, z]$ is its origin. Let $\mathbf{R}_l$ be chosen in such a way as to direct the vector $\mathbf{m}$ along the laser beam. Consider the normal vector $\mathbf{n} = [n_1, n_2, n_3]^T$ to the welding surface. Rotating about the axes x and z in the original coordinate system by the angles ω and ϕ (Fig. 3.2), $|\omega| \leq \phi_{max}$,$|\phi| \leq \phi_{max}$, we obtain the vector:

$$\mathbf{n}^{\omega,\phi} = \begin{bmatrix} \cos(\omega)\, n_1 + \sin(\omega)\cos(\phi)\, n_2 + \sin(\omega)\sin(\phi)\, n_3 \\ -\sin(\omega)\, n_1 + \cos(\omega)\cos(\phi)\, n_2 + \cos(\omega)\sin(\phi)\, n_3 \\ -\sin(\phi)\, n_2 + \cos(\phi)\, n_3 \end{bmatrix}. \tag{3.6}$$

Given a particular robot position determined by $\theta_1, \ldots, \theta_6$, the resulting orientation of the laser beam with respect to the welding surface (given in terms of ω and ϕ) may be calculated by solving the equations (3.6) and (3.5):

$$\mathbf{n}_i^{\omega,\phi} = m_i. \tag{3.7}$$

These equations may be solved with the aid of Gröbner basis computation. Let us perform the change of variables $c_1 = cos(\omega), s_1 = sin(\omega), c_2 = cos(\phi), s_2 = sin(\phi)$ and introduce corresponding constraints: $c_1^2 + s_1^2 = 1$, $c_2^2 + s_2^2 = 1$. Eliminating variables by computing a Gröbner basis with lexicographic ordering $m_1 \prec m_2 \prec m_3 \prec n_1 \prec n_2 \prec n_3 \prec s_1 \prec c_1 \prec s_2 \prec c_2$ we obtain the solutions of (3.7):

$$\phi = \arcsin\left(1/2\, \frac{-2\, n_2 m_3 + 2\sqrt{{n_3}^4 - {n_3}^2{m_3}^2 + {n_2}^2{n_3}^2}}{{n_3}^2 + {n_2}^2}\right),$$

and

$$\omega = \arcsin\left(\frac{m_1 cos(\phi)\, n_2 + m_1 sin(\phi)\, n_3 - n_1 m_2}{{m_1}^2 + {m_2}^2}\right).$$

Let us write the kinematic equations in the following form.

$$\begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \begin{bmatrix} X(\theta_1 \dots \theta_6) \\ Y(\theta_1 \dots \theta_6) \\ Z(\theta_1 \dots \theta_6) \end{bmatrix} \tag{3.8}$$

where $X(\theta_1 \dots \theta_6)$, $Y(\theta_1 \dots \theta_6)$, $Z(\theta_1 \dots \theta_6)$ may be derived from (3.5).

To obtain the equations of motion of the end-effector we differentiate (3.8) with respect to t and apply the chain rule. The resulting system of equations may be written in the following form:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} X_{\theta 1} \\ Y_{\theta 1} \\ Z_{\theta 1} \end{bmatrix} \cdot \dot{\theta_1} + ... + \begin{bmatrix} X_{\theta 6} \\ Y_{\theta 6} \\ Z_{\theta 6} \end{bmatrix} \cdot \dot{\theta_6}, \tag{3.9}$$

where $X_{\theta i}, Y_{\theta i}, Z_{\theta i}$ denote derivatives with respect to θ_i. Note that in the general case the equations admit infinitely many solutions. In the following section we shall express the variety of all solutions explicitly.

3.2 Solving Equations of Motion using Generalized Inverses

In this section we describe the computation of pseudoinverse matrices and express the solution of equations (3.9) explicitly. The pseudoinverse for a general rectangular matrix may be defined (see [4]) as follows.

Definition 1 *Let $A \in R^{m \times n}$. The Matrix $X \in R^{n \times m}$ is called pseudoinverse of A provided $AXA = A$.*

Let

$$A = \begin{bmatrix} X_{\theta_1} & \cdots & X_{\theta_6} \\ Y_{\theta_1} & \cdots & Y_{\theta_6} \\ Z_{\theta_1} & \cdots & Z_{\theta_6} \end{bmatrix} \tag{3.10}$$

$$b = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}$$

Then (3.9) may be written as follows:

$$Ax = b \tag{3.11}$$

The following result has been shown in [5],[51]:

Theorem 2 *The system (3.11) has a solution if and only if*

$$AXb = b,$$

in which case the most general solution is

$$x = Xb + (I - XA)y,$$

where y is arbitrary.

Let $R_r^{m\times n}$ denote the class of $m \times n$ real matrices of rank r. Let I_r denote the unit matrix $r \times r$. Let P denote the permutation matrix. If P_j denotes the jth column of P, and e_j the jth column of I_n, we have $P_j = e_k$, where $k = e_j$, $j = 1, 2, \cdots, n$. The remaining columns of P are the remaining unit vectors in any order. The pseudoinverse X can be calculated by transforming a given matrix A into a Hermite normal form by a sequence of elementary row operations. Every elementary row operation (multiplication of a given row by a nonzero scalar and addition to a given row of a scalar multiple of another row) can be interpreted as a premultiplication of A by a suitable nonsingular matrix, called an elementary row matrix. Any matrix can be transformed into its Hermite normal form by a finite sequence of elementary row operations. Therefore, for any matrix $A^{m\times n}$ there is a nonsingular $m \times m$ matrix E (the product of the elementary row matrices) and a permutation matrix P of order n, such that

$$EAP = \begin{bmatrix} I_r & K \\ 0 & 0 \end{bmatrix}$$

According to [4] the following theorem holds:

Theorem 3 *Let $A \in R^{m\times n}$, and let $E \in R^{m\times m}$ and $P \in R^{n\times n}$ be such that*

$$EAP = \begin{bmatrix} I_r & K \\ 0 & 0 \end{bmatrix}.$$

Then for any $L \in R^{(n-r)\times(m-r)}$, the $n \times m$ matrix

$$X = P \begin{bmatrix} I_r & 0 \\ 0 & L \end{bmatrix} E$$

is a pseudoinverse of A.

Let $T_0 = [A \quad I_m]$. A matrix E transforming A into a Hermite normal form can be found by Gaussian Elimination on T_0, where after the elimination is completed,

$$ET_0 = [EA \quad E],$$

E being recorded as the right-hand $m \times m$ submatrix of ET_0. We use division-free Gaussian Elimination which can be used for calculations over an integral domain:

for *k = 0, 1, 2, ..., m-1*;
 for *i = k+1, ..., m*;
  for *j = k, ..., n*;
   $a_{i,j}^{k+1} = a_{k,k}^k a_{i,j}^k - a_{k,j}^k a_{i,k}^k$

In order to obtain a Hermite normal form each row can be divided by the diagonal elements $a_{i,i}^{m-1}$ such that diagonal elements become equal to 1. In the final step we eliminate all non-diagonal elements from each of first m rows of A:

for *k = m, ..., 1*;
 for *i = k-1, ..., 1*;
  for *j = 1, ..., n*;
   $a_{i,j}^{k+1} = a_{k,k}^k a_{i,j}^k - a_{k,j}^k a_{i,k}^k$

These algorithms can be applied for matrices with polynomial entries. The resulting analytic inverse for the matrix (3.10) is too large to be reproduced here. Let us consider the following numerical example for $\theta_1, ..., \theta_6$ equal to 1.341683383, −.2875582134, −1.510743899, .7523051353, 1.025718970, and −.9611240531. Then A reduces to

$$A = \begin{bmatrix} -853.9784765 & 48.84911674 & -749.6416328 \\ 263.3407003 & -513.7069127 & -518.5943929 \\ -270.0005971 & 66.87701330 & -159.6786724 \end{bmatrix} \cdots$$

$$\begin{bmatrix} -1785.623805 & -80.72274815 & -135.4730149 \\ 701.0000547 & -346.1407570 & -580.9109943 \\ 0 & -1548.169056 & -733.0706917 \end{bmatrix}$$

Performing Gaussian Elimination on $T_0 = [A \quad I_m]$ we obtain:

$$\begin{bmatrix} 1.0 & -0.05720181256 & 0.8778226307 \\ 0.0 & 1.0 & 1.503601368 \\ 0.0 & 0.0 & 1.0 \end{bmatrix} \cdots$$

$$\begin{bmatrix} -0.001170989700 & -0.0 & -0.0 \\ -0.0006184164479 & -0.002005441375 & 0.0 \\ -5687224.412 & 2062896.860 & 20000000.0 \end{bmatrix}$$

Elimination of elements above the diagonal yields:

$$\left[\begin{array}{ccc} 1.0 & 0 & 0 \\ 0 & 1.0 & 0 \\ 0.0 & 0.0 & 1.0 \end{array}\right. \cdots$$

$$\left.\begin{array}{ccc} 5481525.206 & -1988284.675 & -19276627.08 \\ 8551318.405 & -3101774.543 & -30072027.36 \\ -5687224.412 & 2062896.860 & 20000000.0 \end{array}\right]$$

Now the matrix E can be built from last three columns of the above matrix and X can be calculated according to Theorem 3:

$$X = \left[\begin{array}{ccc} 5481525.206 & -1988284.675 & -19276627.08 \\ 8551318.405 & -3101774.543 & -30072027.36 \\ 0.0 & 0.0 & 0.0 \\ -5687224.412 & 2062896.860 & 20000000.0 \\ 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 \end{array}\right].$$

The parameterized solution of (3.11) may be expressed according to Theorem 2:

$$\begin{aligned} \dot{\theta}_1 &= 5481525.206\,\dot{x} - 1988284.675\,\dot{y} - 19276627.08\,\dot{z} + 2.0 \\ &\gamma + 11181729560.0\,\delta - 30089220140.0\,\omega - 14543548020.0\,\xi \\ \dot{\theta}_2 &= 8551318.405\,\dot{x} - 3101774.543\,\dot{y} - 30072027.36\,\dot{z} - 5.0 \\ &\gamma + 17443781820.0\,\delta - 46939946910.0\,\omega - 22688303970.0\,\xi \\ \dot{\theta}_3 &= \gamma \\ \dot{\theta}_4 &= -5687224.412\,\dot{x} + 2062896.860\,\dot{y} + 20000000.0\,\dot{z} - 2.0 \\ &\gamma - 11601334119.0\,\delta + 31218345390.0\,\omega + 15089307840.0\,\xi \\ \dot{\theta}_5 &= \omega \\ \dot{\theta}_6 &= \xi \end{aligned}$$

In general, computation of Moore-Penrose pseudoinverse of the Jacobian with polynomial entries allows us to parameterize the joint velocity space at the end points of offsets between

tasks:

$$
\begin{aligned}
&\dot{\theta_1} = U_1(\theta_1,\ldots,\theta_6)\dot{x} + U_2(\theta_1,\ldots,\theta_6)\dot{y}+\\
&U_3(\theta_1,\ldots,\theta_6)\dot{z}+\\
&(U_1(\theta_1,\ldots,\theta_6)X_{\theta 4} + U_2(\theta_1,\ldots,\theta_6)Y_{\theta 4}+\\
&U_3(\theta_1,\ldots,\theta_6)Z_{\theta}4)\xi+\\
&(U_1(\theta_1,\ldots,\theta_6)X_{\theta 5} + U_2(\theta_1,\ldots,\theta_6)Y_{\theta 5}+\\
&U_3(\theta_1,\ldots,\theta_6)Z_{\theta}5)\eta+\\
&(U_1(\theta_1,\ldots,\theta_6)X_{\theta 6} + U_2(\theta_1,\ldots,\theta_6)Y_{\theta 6}+\\
&U_3(\theta_1,\ldots,\theta_6)Z_{\theta 6})\delta,\\
\\
&\dot{\theta_2} = V_1(\theta_1,\ldots,\theta_6)\dot{x} + V_2(\theta_1,\ldots,\theta_6)\dot{y}+\\
&V_3(\theta_1,\ldots,\theta_6)\dot{z}+\\
&(V_1(\theta_1,\ldots,\theta_6)X_{\theta 4} + V_2(\theta_1,\ldots,\theta_6)Y_{\theta 4}+\\
&V_3(\theta_1,\ldots,\theta_6)Z_{\theta 4})\xi+\\
&(V_1(\theta_1,\ldots,\theta_6)X_{\theta 5} + V_2(\theta_1,\ldots,\theta_6)Y_{\theta 5}+\\
&V_3(\theta_1,\ldots,\theta_6)Z_{\theta 5})\eta+\\
&(V_1(\theta_1,\ldots,\theta_6)X_{\theta 6} + V_2(\theta_1,\ldots,\theta_6)Y_{\theta 6}+\\
&V_3(\theta_1,\ldots,\theta_6)Z_{\theta 6})\delta,\\
\\
&\dot{\theta_3} = W_1(\theta_1,\ldots,\theta_6)\dot{x} + W_2(\theta_1,\ldots,\theta_6)\dot{y}+\\
&W_3(\theta_1,\ldots,\theta_6)\dot{z}+\\
&(W_1(\theta_1,\ldots,\theta_6)X_{\theta 4} + W_2(\theta_1,\ldots,\theta_6)Y_{\theta 4}+\\
&W_3(\theta_1,\ldots,\theta_6)Z_{\theta 4})\xi+\\
&(W_1(\theta_1,\ldots,\theta_6)X_{\theta 5} + W_2(\theta_1,\ldots,\theta_6)Y_{\theta 5}+\\
&W_3(\theta_1,\ldots,\theta_6)Z_{\theta 5})\eta+\\
&(W_1(\theta_1,\ldots,\theta_6)X_{\theta 6} + W_2(\theta_1,\ldots,\theta_6)Y_{\theta 6}+\\
&W_3(\theta_1,\ldots,\theta_6)Z_{\theta 6})\delta,\\
\\
&\dot{\theta_4} = \xi,\\
\\
&\dot{\theta_5} = \eta,\\
\\
&\dot{\theta_6} = \delta,
\end{aligned}
\tag{3.12}
$$

where $U_1 = \frac{-Y_{\theta 3}Z_{\theta 2}+Z_{\theta 3}Y_{\theta 2}}{D}$, $U_2 = \frac{X_{\theta_3}Z_{\theta 2}-X_{\theta 2}Z_{\theta 3}}{D}$, $U_3 = \frac{-X_{\theta_3}Y_{\theta 2}+X_{\theta 2}Y_{\theta 3}}{D}$, $V_1 = \frac{Y_{\theta_3}Z_{\theta 1}-Z_{\theta 3}Y_{\theta 1}}{D}$, $V_2 = \frac{-X_{\theta_3}Z_{\theta 1}+Z_{\theta 3}X_{\theta 1}}{D}$, $V_3 = \frac{X_{\theta 3}Y_{\theta 1}-Y_{\theta 3}X_{\theta 1}}{D}$, $W_1 = \frac{Y_{\theta 1}Z_{\theta 2}-Z_{\theta 1}Y_{\theta 2}}{D}$, $W_2 = \frac{X_{\theta 2}Z_{\theta 1}-Z_{\theta 2}X_{\theta 1}}{D}$, $W_3 = \frac{-X_{\theta 2}Y_{\theta 1}+Y_{\theta 2}X_{\theta 1}}{D}$. D is equal to $X_{\theta 3}Y_{\theta 1}Z_{\theta 2}+Y_{\theta 3}X_{\theta 2}Z_{\theta 1}-Y_{\theta 3}X_{\theta 1}Z_{\theta 2}-X_{\theta_3}Z_{\theta 1}Y_{\theta 2}-Z_{\theta 3}X_{\theta 2}Y_{\theta 1}+Z_{\theta 3}X_{\theta 1}Y_{\theta_2}$ and is known to be non-vanishing except a finite number of singular positions of the robot. $X_{\theta i}$, $Y_{\theta i}$, $Z_{\theta i}$ denote derivatives of the components of forward kinematic mapping (3.8) with respect to θ_i. In the following Section we shall use these equations in order to compute minimum-time motion of the robot for prescribed tasks.

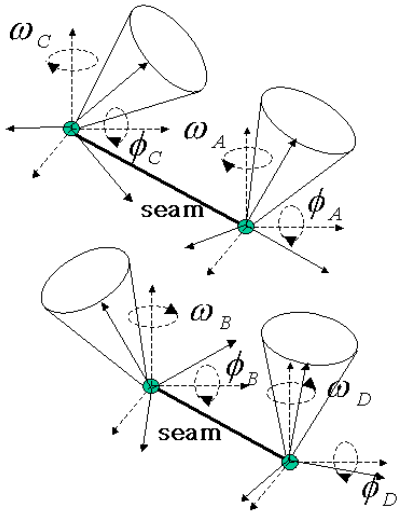


Figure 3.4: The optimal path problem for laser welding: calculate admissible orientations of laser beam such that the total time is minimized.

3.3 Computing Minimum-Time Motion

Since the velocity of the end-effector is prescribed for individual tasks, the minimum-time trajectories may be calculated using the freedom in the orientation of end-effector during the task execution in such a way as to produce minimum-time trajectories in offsets between tasks. As shown in Fig. 3.4, orientation of the end-effector can be described by angles ω, ϕ ($|\omega| \leq \phi_{max}$,$|\phi| \leq \phi_{max}$). Consider admissible orientations A, B at the end points of an offset, and admissible orientations C, D at the end points of corresponding seams. Let $\theta_1^A, \ldots, \theta_6^A$, $\theta_1^B, \ldots, \theta_6^B$, $\theta_1^C, \ldots, \theta_6^C$, $\theta_1^D, \ldots, \theta_6^D$ denote the joint angles of the robot in positions corresponding to orientations A, B, C, and D, respectively. The joint trajectories between these positions are described by curves $\theta_1^{AB}(t), \ldots, \theta_6^{AB}(t)$, $\theta_1^{CA}(t), \ldots, \theta_6^{CA}(t)$, $\theta_1^{BD}(t), \ldots, \theta_6^{BD}(t)$, such that

$$\begin{array}{ll} \theta_i^{CA}(1) = \theta_i^{AB}(0); & \theta_i^{AB}(1) = \theta_i^{BD}(0); \\ \dot{\theta}_i^{CA}(1) = \dot{\theta}_i^{AB}(0); & \dot{\theta}_i^{AB}(1) = \dot{\theta}_i^{BD}(0); \\ \ddot{\theta}_i^{CA}(1) = \ddot{\theta}_i^{AB}(0); & \ddot{\theta}_i^{AB}(1) = \ddot{\theta}_i^{BD}(0); \end{array} \tag{3.13}$$

Since for the motion between tasks we do not have any restrictions for the position and velocity of the end-effector we use the well known approach ([31], [42]) to describe the vector of joint displacements $\theta(t) = [\theta_i^{AB}(t) | i = 1 \ldots 6\ ]$ by cubic B-splines with respect to the sequence of knots t_j, $t_j < t_{j+1}$, $j = 1 \ldots N$ distributed in the interval $[0, 1]$, :

$$\theta(t) = \sum_{j=0}^{N} \mathbf{x}_j N_j^3(t) \tag{3.14}$$

where N_j^3 denote the B-spline basis functions of order 3; t_j, $\mathbf{x}_j$ denote control points, which determine the shape of the spline curve, and $t \in [t_1, t_N]$ is a parameter. The derivatives of

the spline curve are given by

$$\dot{\mathbf{x}} = 3\sum_{1}^{N} \mathbf{x}_j^{(1)} N_j^2, \quad \mathbf{x}_j^{(1)} = \frac{\mathbf{x}_j - \mathbf{x}_{j-1}}{t_{j+3} - t_j}, \tag{3.15}$$

as well as

$$\ddot{\mathbf{x}} = 6\sum_{1}^{N} \mathbf{x}_j^{(2)} N_j^1, \quad \mathbf{x}_j^{(2)} = \frac{\mathbf{x}_j^{(1)} - \mathbf{x}_{j-1}^{(1)}}{t_{j+2} - t_j}. \tag{3.16}$$

According to the approach presented in ([31], [42]) the problem of computing minimum-time spline curve satisfying acceleration and velocity conditions (3.1) reduces to minimization of $t_N - t_3$ in such a way that $|\dot{\mathbf{x}}_i| < \dot{\theta}_i^{max}$, $|\ddot{\mathbf{x}}_i| < \ddot{\theta}_i^{max}$, $i = 1 \dots 6$. The following necessary optimality condition derived in [31] tells that that maximal acceleration has to be reached at least at one of four consecutive knots (otherwise, due to the locality property of B-Splines, the fourth and all consecutive knots could be shifted to the left meaning decreasing the time without violation of velocity and acceleration constraints) :

Theorem 4 *A spline curve is time optimal with respect to acceleration and velocity constraints if* $\|\dot{\mathbf{x}}_i\| < \dot{\theta}_i^{max}$ *In this case, we have that for at least one* i, $i = 1 \dots 6$,

$$\prod_{k=0}^{3} (\frac{1}{6}\dot{\theta}_i^{max} - |\mathbf{x}_{j+k}^{(2)}|_i) = 0, j = 2, \dots, N,$$

where $\mathbf{x}^{(2)}$, $\mathbf{x}^{(1)}$ *are defined in (3.15), (3.16).*

In order to obtain the minimum-time spline curve for individual offsets between tasks the knot-shifting algorithms presented in [31], [42] start with an arbitrary distribution of knots and try to get the distances between knots $|t_{j+1} - t_j|$ as small as possible. Due to the Theorem 4 this can be done by considering $|\mathbf{x}_{j+k}^{(2)}|$ as function in t_j solving the equation $(\frac{1}{6}\dot{\theta}_i^{max} - |\mathbf{x}_{j+k}^{(2)}|_i) = 0$ numerically.

The complete optimization problem reduces to calculation of admissible A, B, $\theta_i^{CA}(t)$, $\theta_i^{BD}(t)$ for all tasks, and to approximate $\theta_i^{AB}(t)$ by time optimal spline curves such that the total motion time is minimized, and constraints

$$\begin{aligned} |\theta_i| &\leq \theta_i^{max} \\ \left|\dot{\theta}_i\right| &\leq \dot{\theta}_i^{max} \\ \left|\ddot{\theta}_i\right| &\leq \ddot{\theta}_i^{max} \end{aligned}$$

are satisfied. Thus, the solution space for this optimization problem is determined by:

1. admissible orientations A and B for every offset (and corresponding joint positions $\theta_1^A, \dots, \theta_6^A$, $\theta_1^B, \dots, \theta_6^B$)

2. admissible velocities $\dot{\theta}_1^A, \ldots, \dot{\theta}_6^A$, $\dot{\theta}_1^B, \ldots, \dot{\theta}_6^B$ at the end points of offsets due to constraints on velocity of the end-effector during execution of corresponding tasks

3. curves $\theta_1^{A,B}(t), \ldots, \theta_6^{A,B}(t)$ such that (3.13) is satisfied.

We use the method of steepest descent, and the cost function $F(\omega_A, \phi_A, \omega_B, \phi_B, \xi_A, \eta_A, \delta_A, \xi_B, \theta_B, \delta_B)$, where ω_A, ϕ_A, ω_B, ϕ_B determine orientations of the end-effector at the end points of the offsets, and ξ_A, η_A, δ_A , ξ_B, η_B, δ_B determine the possible velocities of joints at the end points of the offsets. Given ξ_A, η_A, δ_A , ξ_B, η_B, δ_B, we then get $\dot{\theta}_1^A, \ldots, \dot{\theta}_6^A$, $\dot{\theta}_1^B, \ldots, \dot{\theta}_6^B$ using (12). Since the time required for task execution is prescribed, the value of the cost function F is determined by the motion time for offsets. This quantity can be calculated by computing time optimal B-spline curve (3.14) for every offset. Several methods for non-linear programming, e.g. the method introduced in [31], can be used for this purpose. Consider spline curve with boundary conditions given by (3.13). The problem can be formulated as follows: given a velocity bound, and an acceleration bound, determine knots t_j, $j = 3, \ldots, N-3$, such that $t_{N-3} - t_3$ is minimized under the restrictions on velocity, and acceleration. The method proposed in [31] begins with an arbitrary distribution of knots and then attempts to get any successive knots as close as possible while still maintaining the velocity and acceleration constraints (3.1).

3.4 Conclusion

In this chapter we described an approach to calculation of minimum-time trajectories for robots with 6 rotation joints for multiple tasks with constrained orientation and velocity of the end-effector. In order to reduce the time required for execution of prescribed tasks with such type of constraints we parameterized admissible orientation and velocity spaces using computation of Moore-Penrose pseudoinverses of matrices with polynomial entries. We used computer algebra system Maple to perform these calculations and derive formulae and procedures for numerical optimization by the method of steepest descent. Finally, the computational examples have been presented.

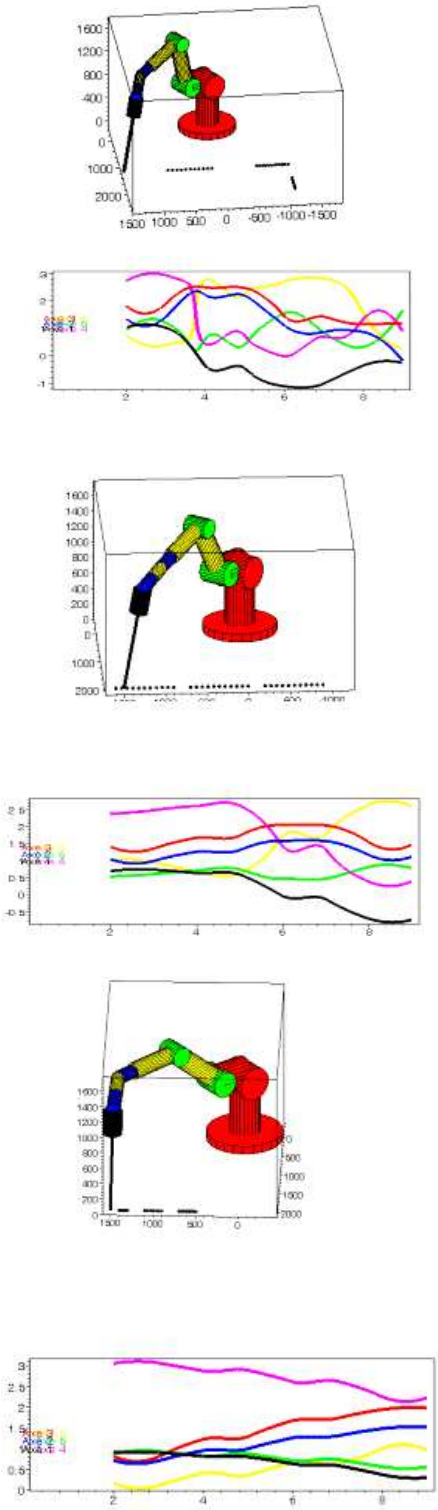


Figure 3.5: Computational examples.

Chapter 4

Grid Generation

Advanced computer technologies and parallel architectures allow one to solve time dependent problems with 10^9 and more unknowns on rectangular regions in realistic time using hierarchical and adaptive approaches [9, 69]. In order to handle problems of such order of computational complexity on arbitrary regions and, in particular, with moving boundaries, we are interested to have efficient grid generation techniques, which would support hierarchical approach to computing and provide the possibility of adaptive mesh refinement as well as remeshing, due to the changes of boundaries, with minimal computational costs.

Considered as optimization task the problem may be reduced to finding a minimizer of the weighted combination of so-called length, area, and orthogonality functionals. Unfortunately, it has been shown that on the one hand, certain weights of the individual functionals do not admit the unique optimizer on certain geometric domains. On the other hand, some combinations of these functionals lead to the lack of ellipticity of corresponding Euler-Lagrange equations, and finding the optimal grid becomes computationally too expensive for practical applications. Choosing the right functional for the particular geometric domain of interest may improve the grid generation very much, but choosing the functional parameters is usually done in the trial and error way and depends very much on the geometric domain. This makes the automatic and robust grid generation difficult. In this chapter we present a framework for generating structured and unstructured grids implemented in Computer Algebra System Maple.

4.1 Structured Grids

The problem of grid generation on an arbitrary region Ω in the (x, y) plane can be solved by giving a map $x(\xi, \eta), y(\xi, \eta)$ from the unit square in the plane (ξ, η) onto Ω. By choosing a uniform grid (ξ_i, η_j) in the unit square, the map $x(\xi_i, \eta_j), y(\xi_i, \eta_j)$ would transform the grid (ξ_i, η_j) to the region of interest. The required map may be computed in a number of ways. The variational grid generation is one of the most established approaches for this purpose, due to high quality of resulting grids. It provides the possibility to control the grid properties by choosing appropriate grid functionals to be minimized. The basic

functionals are Length (I_L), Area (I_A), and Orthogonality (I_O) functionals, which can be written in the form (see [39]):

$$I_L(x,y) = 1/2 \iint ({x_\xi}^2 + {y_\xi}^2 + {x_\eta}^2 + {y_\eta}^2) d\xi \, d\eta; \tag{4.1}$$

$$I_A(x,y) = 1/2 \iint ({x_\xi}^2 {y_\eta}^2 + {y_\xi}^2 {x_\eta}^2 - 2\, x_\xi x_\eta y_\xi y_\eta) d\xi \, d\eta; \tag{4.2}$$

$$I_O(x,y) = 1/2 \iint ({x_\xi}^2 {x_\eta}^2 + 2\, x_\xi x_\eta y_\xi y_\eta + {y_\xi}^2 {y_\eta}^2) d\xi \, d\eta. \tag{4.3}$$

The map $x(\xi,\eta), y(\xi,\eta)$ minimizing each of above functionals can be found by by solving corresponding Euler–Lagrange equations, which can be written in general form

$$\mathcal{T}_{1,1}\mathbf{x}_{\xi,\xi} + \mathcal{T}_{1,2}\mathbf{x}_{\xi,\eta} + \mathcal{T}_{2,2}\mathbf{x}_{\eta,\eta} + \mathcal{S} = 0,$$

where $\mathcal{T}_{i,j}$ are 2 x 2 matrices and $\mathcal{S}$ is a 2 x 1 vector. The terms in $\mathcal{T}_{i,j}$ and $\mathcal{S}$ depend on the particular functional and are nonlinear in the case of Area and Orthogonality Functionals. In the case of the Length functional I_L, $T_{i,j}$ can be shown to be constant, and the Euler–Lagrange equations reduce to the simplest one:

$$x_{\xi,\xi} + x_{\eta,\eta} = 0, \qquad y_{\xi,\xi} + y_{\eta,\eta} = 0.$$

Minimizing I_L by solving above equations leads to smooth grids. However, intersections of grid lines may occur (Fig. 4.1). The folding of resulting grids by using the Length functional is inadmissible for practical applications. The Area functional leads to the following Euler–Lagrange equations, which produce unfolded but, unfortunately, nonsmooth grids:

$$\begin{aligned} &x_{\xi,\xi}{y_\eta}^2 + y_\eta x_\xi y_{\xi,\eta} - y_\eta y_{\xi,\xi} x_\eta - 2\, y_\eta y_\xi x_{\xi,\eta} - y_\xi x_\xi y_{\eta,\eta} + y_\xi y_{\xi,\eta} x_\eta + {y_\xi}^2 x_{\eta,\eta} = 0, \\ &-x_\eta x_{\xi,\xi} y_\eta - 2\, x_\eta x_\xi y_{\xi,\eta} + y_{\xi,\xi}{x_\eta}^2 + x_\eta y_\xi x_{\xi,\eta} + x_\xi x_{\xi,\eta} y_\eta + {x_\xi}^2 y_{\eta,\eta} - x_\xi y_\xi x_{\eta,\eta} = 0. \end{aligned}$$

As described in [39], the further shortcoming of this method is that available numerical procedures for solving the above equations do not converge for certain domains. The Orthogonality functional produces orthogonal and sufficiently smooth grids on many domains, however, fails to converge in certain cases. Euler-Lagrange equations for the Orthogonality functional are:

$$\begin{aligned} &x_{\xi,\xi}{x_\eta}^2 + 4\, x_\eta x_\xi x_{\xi,\eta} + x_\eta y_{\xi,\xi} y_\eta + x_\eta y_\xi y_{\xi,\eta} + 2\, x_{\xi,\eta} y_\xi y_\eta + {x_\xi}^2 x_{\eta,\eta} + x_\xi y_{\xi,\eta} y_\eta \\ &\qquad + x_\xi y_\xi y_{\eta,\eta} = 0, \\ &y_\eta x_{\xi,\xi} x_\eta + y_\eta x_\xi x_{\xi,\eta} + y_{\xi,\xi}{y_\eta}^2 + 4\, y_\eta y_\xi y_{\xi,\eta} + 2\, y_{\xi,\eta} x_\xi x_\eta + y_\xi x_{\xi,\eta} x_\eta + y_\xi x_\xi x_{\eta,\eta} \\ &\qquad + {y_\xi}^2 y_{\eta,\eta} = 0. \end{aligned}$$

In order to obtain smooth, orthogonal, and unfolded grids, the weighted combination of Length, Area, and Orthogonality functionals may be used:

$$I(x,y) = \omega_A I_A(x,y) + \omega_L I_L(x,y) + \omega_O I_I(x,y) \tag{4.4}$$

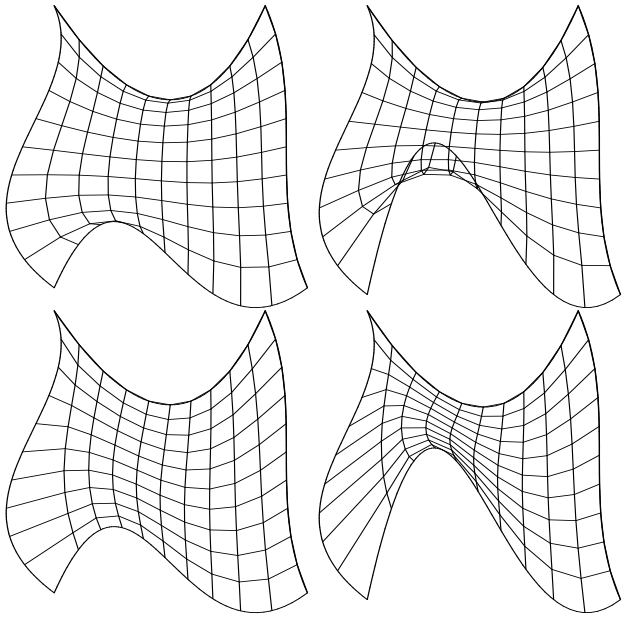


Figure 4.1: Grid generation by minimizing the Length functional (top), and by minimizing the Winslow functional (bottom)

.

In particular, Area-Length combination overcomes the limitation of individual functionals because of avoiding grid folding produced by Length functional and producing smooth grids in contrast to the Area functional. However, the corresponding equations do not admit the continuous solution on many practically important domains like airfoil, backstep, and "C"-domains (see [39]). In order to preserve the advantages of the Length functional and avoid the grid foldings the famous Winslow grid generator has been proposed. The Winslow functional

$$I_W(x,y) = \iint \frac{{x_\eta}^2 + {y_\eta}^2}{\left(x_\xi y_\eta - x_\eta y_\xi\right)^2} + \frac{{x_\xi}^2 + {y_\xi}^2}{\left(x_\xi y_\eta - x_\eta y_\xi\right)^2} d\xi\, d\eta$$

leads to equations:

$$\begin{aligned}
\left({x_\xi}^2 + {y_\eta}^2\right) x_{\xi,\xi} - 2\left(x_\xi x_\eta + y_\xi y_\eta\right) x_{\xi,\eta} + \left({x_\xi}^2 + {x_\eta}^2\right) x_{\eta,\eta} = 0,\\
\left({x_\xi}^2 + {y_\eta}^2\right) y_{\xi,\xi} - 2\left(x_\xi x_\eta + y_\xi y_\eta\right) y_{\xi,\eta} + \left({x_\xi}^2 + {x_\eta}^2\right) y_{\eta,\eta} = 0.
\end{aligned}$$

The Winslow generator inherits the grid smoothness from the Length functional and tends to produce smooth non-folded grids (see Fig. 4.1). However, the lack of orthogonality may lead, for example, to high truncation errors by using the Winslow grids for numerical solution of PDE's. Further modifications of the presented functionals may be found in the literature (see [39]), which tend to produce good meshes in certain cases and fail to admit the solution in other cases. Choosing the right functional for a certain geometric domain, or, in particular, choosing optimal weights in (4.4) may improve the resulting grids significantly. The optimal choice, however, depends on the particular domain very

much and is usually performed in the trial-and-error way. All this makes the automatic and robust grid generation difficult.

For solving the Euler–Lagrange equations corresponding to the individual functionals we use the *Alternating Direction Implicit* (ADI) method introduced in [50].

For instance, consider the Winslow grid generator, which is based on the solution of the following system of nonlinear coupled PDE's:

$$\begin{aligned}\left({x_\xi}^2 + {y_\eta}^2\right) x_{\xi,\xi} - 2\left(x_\xi x_\eta + y_\xi y_\eta\right) x_{\xi,\eta} + \left({x_\xi}^2 + {x_\eta}^2\right) x_{\eta,\eta} = 0,\\ \left({x_\xi}^2 + {y_\eta}^2\right) y_{\xi,\xi} - 2\left(x_\xi x_\eta + y_\xi y_\eta\right) y_{\xi,\eta} + \left({x_\xi}^2 + {x_\eta}^2\right) y_{\eta,\eta} = 0.\end{aligned}$$

As described in [26], we use the following second-order approximation for the partial derivatives of the function $f(\xi,\eta)$:

$$\begin{aligned}(f_\xi)_{i,j} &= 1/2(f_{i+1,j} - f_{i-1,j}),\\ (f_\eta)_{i,j} &= 1/2(f_{i,j+1} - f_{i,j-1}), \quad (f_{\xi,\xi})_{i,j} = (f_{i+1,j} - 2f_{i,j} + f_{i-1,j}),\\ (f_{\eta,\eta})_{i,j} &= (f_{i,j+1} - 2f_{i,j} + f_{i,j-1}),\\ (f_{\eta,\xi})_{i,j} &= 1/4(f_{i+1,j+1} - f_{i+1,j-1} - f_{i-1,j+1} + f_{i-1,j-1})\,.\end{aligned}$$

Let us introduce the following difference operators:

$$\Lambda^n_\xi f_{i,j} = \left[(x^2_\eta)^n_{i,j} + (y^2_\eta)^n_{i,j}\right](f_{i+1,j} - 2f_{i,j} + f_{i-1,j})\,,$$

$$\Lambda^n_\eta f_{i,j} = \left[(x^2_\xi)^n_{i,j} + (y^2_\xi)^n_{i,j}\right](f_{i,j+1} - 2f_{i,j} + f_{i,j-1})\,,$$

$$\Lambda^n_{\eta,\xi} f_{i,j} = -1/2\left[(x_\xi x_\eta)^n_{i,j} + (y_\xi y_\eta)^n_{i,j}\right](f_{i+1,j+1} - f_{i+1,j-1} - f_{i-1,j+1} + f_{i-1,j-1})\,.$$

The superscript denotes the number of iterations. Then the ADI difference scheme, which converges to the solution of Winslow equations using pseudo-time steps τ, may be written as follows:

$$\frac{\tilde{x}_{i,j} - x^n_{i,j}}{0.5\tau} = \Lambda^n_\xi \tilde{x}_{i,j} + \Lambda^n_{\xi,\eta} x^n_{i,j} + \Lambda^n_\eta x^n_{i,j}\,,$$

$$\frac{x^{n+1}_{i,j} - \tilde{x}_{i,j}}{0.5\tau} = \Lambda^n_\xi \tilde{x}_{i,j} + \Lambda^n_{\xi,\eta} \tilde{x}_{i,j} + \Lambda^n_\eta x^{n+1}_{i,j}\,,$$

$$\frac{\tilde{y}_{i,j} - y^n_{i,j}}{0.5\tau} = \Lambda^n_\xi \tilde{y}_{i,j} + \Lambda^n_{\xi,\eta} y^n_{i,j} + \Lambda^n_\eta y^n_{i,j}\,,$$

$$\frac{y^{n+1}_{i,j} - \tilde{y}_{i,j}}{0.5\tau} = \Lambda^n_\xi \tilde{y}_{i,j} + \Lambda^n_{\xi,\eta} \tilde{y}_{i,j} + \Lambda^n_\eta y^{n+1}_{i,j}\,.$$

Using this scheme, for example, the grid in Fig. 4.1 has been obtained.

4.2 Unstructured Grids

In our work we are interested in the integration of computer aided geometric design (CAGD) and numerical simulation in such a way that would allow us to design robust, efficient, and reliable scientific software. On the one hand the physical or numerical properties of the computational problem make demands on the possible geometric representation of an object under consideration. On the other hand different topological and geometric representation of an object exist, which can not be converted to each other in a simple and efficient way.

By far the most common representation for curves and surfaces in CAGD is the parametric representation (Bezier, NURBS or BSPline curves). But the researchers recognized early the power of implicit curves and surfaces for the purpose of modeling and simulation. The present paper shows how the complex geometric regions whose boundaries are given as implicit algebraic curves can be subdivided into symmetric parts to speed-up the computationally expensive advancing front triangulation and finite element computation on the resulting grid. We calculate for the geometric region given as an implicit curve the symmetry axes by computing the invariant finite matrix group of reflections. The advancing front triangulation can also be performed for only one of the symmetric parts, and the resulting grid is assembled. It will be shown that the use of symmetry properties of a given planar region enables the CPU time savings by a factor from 3 to 8.

4.2.1 Hierarchical Methods in Computer Aided Geometric Design and Symmetry

In this section we consider the constructive hierarchical geometry representations. A constructive representation defines an object by the sequence of operations for constructing an object [53]. The most common constructive representation is called Constructive Solid Geometry (CSG) and uses the boolean (set theoretic) operations. The operation sequence is typically stored as a tree. For example, the object shown on the right hand side in the figure 4.2 can be constructed from rectangle, circle, and cone using set union and difference operations.

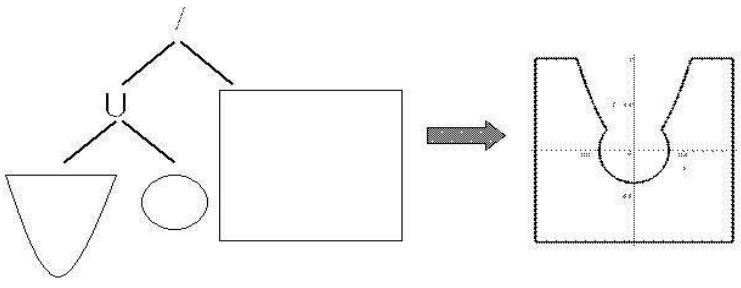


Figure 4.2: A geometric object formed by set theoretic operations

To convert this set theoretic operations to a real valued functions the R-Functions

proposed in [56] (a short introduction and basic applications of R-Functions can be found in [62] too) can be used. R-Functions allow us to write easily an equation for an object of arbitrary shape, in the same way as one forms the solid by the boolean operations. If $\mathbf{x} = (x_1, ..., x_n)$ is a point in R^n, then:

$$\begin{array}{ll} f(\mathbf{x}) > 0 & \text{if } \mathbf{x} \text{ is inside the object} \\ f(\mathbf{x}) = 0 & \text{if } \mathbf{x} \text{ is on the boundary of the object} \\ f(\mathbf{x}) < 0 & \text{if } \mathbf{x} \text{ is outside the object} \end{array} \tag{4.5}$$

The set-theoretic operations on objects described as R-Functions can be defined as follows

$$\begin{aligned} f_1(\mathbf{x}) \cup f_2(\mathbf{x}) &= f_1(\mathbf{x}) + f_2(\mathbf{x}) + \sqrt{f_1^2(\mathbf{x}) + f_2^2(\mathbf{x})} \\ f_1(\mathbf{x}) \cap f_2(\mathbf{x}) &= f_1(\mathbf{x}) + f_2(\mathbf{x}) - \sqrt{f_1^2(\mathbf{x}) + f_2^2(\mathbf{x})} \\ f_1(\mathbf{x}) \backslash f_2(\mathbf{x}) &= f_1(\mathbf{x}) - f_2(\mathbf{x}) - \sqrt{f_1^2(\mathbf{x}) + f_2^2(\mathbf{x})} \end{aligned} \tag{4.6}$$

Note that the boundary of the geometric region is represented as roots of the R-Functions $f(\mathbf{x}) = 0$. We can isolate the squared roots in (4.6) and square left and right hand side respectively, in case of the intersection, for example :

$$\begin{gathered} -\sqrt{f_1^2(\mathbf{x}) + f_2^2(\mathbf{x})} = f_1(\mathbf{x}) + f_2(\mathbf{x}) \\ f_1^2(\mathbf{x}) + f_2^2(\mathbf{x}) = f_1^2(\mathbf{x}) + 2f_1(\mathbf{x})f_2(\mathbf{x}) + f_2^2(\mathbf{x}) \\ 2f_1(\mathbf{x})f_2(\mathbf{x}) = 0 \end{gathered}$$

In this way we obtain the the point set containing boundary given by the equation $f_1(\mathbf{x})f_2(\mathbf{x}) = 0$:

$$\begin{aligned} \partial(f_1(\mathbf{x}) \cup f_2(\mathbf{x})) &\subseteq \{\mathbf{x} : f_1(\mathbf{x}) * f_2(\mathbf{x}) = 0\} \\ \partial(f_1(\mathbf{x}) \cap f_2(\mathbf{x})) &\subseteq \{\mathbf{x} : f_1(\mathbf{x}) * f_2(\mathbf{x}) = 0\} \\ \partial(f_1(\mathbf{x}) \backslash f_2(\mathbf{x})) &\subseteq \{\mathbf{x} : f_1(\mathbf{x}) * f_2(\mathbf{x}) = 0\} \end{aligned} \tag{4.7}$$

In the next section we will show, how the domain boundary obtained according to (4.7) can be used to compute finite symmetry groups of the domain.

The rectangle in Fig. 2.7 can be constructed as the intersection of 4 half-spaces according to (4.6) :

$$\begin{aligned} f_1(x, y) &= 1 - x \\ f_2(x, y) &= 1 + x \\ f_3(x, y) &= 1 - y \\ f_4(x, y) &= 1 + y \end{aligned}$$

Then we obtain:

$$\mathrm{Rect}(x,y) = (f_1 \cap f_2) \cap (f_3 \cap f_4) = f1 + f2 - \sqrt{f1^2 + f2^2} + f3 + f4 - \sqrt{f3^2 + f4^2} - \sqrt{\left(f1 + f2 - \sqrt{f1^2 + f2^2}\right)^2 + \left(f3 + f4 - \sqrt{f3^2 + f4^2}\right)^2}$$

Another representation of the rectangle boundary is the one according to (4.2.1):

$$\partial\mathrm{Rect}(x,y) \subseteq (x-1)(1+x)(y-1)(1+y)$$

Other primitives used in the above example are:

$$\mathrm{parabola}(x,y) = y - 3\,x^2; \qquad \mathrm{circle}(x,y) = -x^2 - y^2 + \frac{1}{8}.$$

The complete object is given by

$$O(x,y) = Rect \backslash (circle \cup parabola) =$$
$$Rect - parabola - circle - \sqrt{parabola^2 + circle^2} - \sqrt{Rect^2 + \left(parabola + circle + \sqrt{parabola^2 + circle^2}\right)^2}$$

Obviously the following symmetry properties hold:

$$\mathrm{Rect}(x,y) = \mathrm{Rect}(\pm x, \pm y)$$
$$\mathrm{Rect}(x,y) = \mathrm{Rect}(\pm y, \pm x)$$

or in the matrix form

$$\mathrm{Rect}(D_4\mathbf{x}) = \mathrm{Rect}(\mathbf{x})$$
$$D_4 = \left\{ \begin{bmatrix} \pm 1 & 0 \\ 0 & \pm 1 \end{bmatrix}, \begin{bmatrix} 0 & \pm 1 \\ \pm 1 & 0 \end{bmatrix} \right\}$$

D_4 is the well known dihedral group whose elements correspond to rotations and reflections in the plane. The circle has a symmetry group SO_2

$$SO_2 = \begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix}$$

and the parabola is a reflection symmetric with respect to the y-axis:

$$Ry = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

In the present paper we present an algorithm for computation of the symmetric decomposition shown in the following figure and show how the costs of advancing front triangulation can be reduced by performing them on the symmetric parts only marked in the figure.

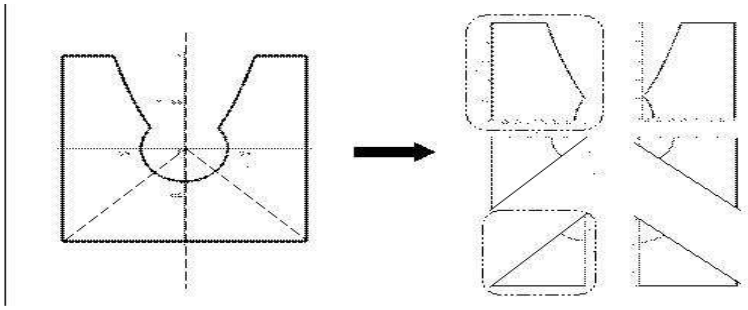


Figure 4.3: Decomposition into symmetric parts

4.2.2 Computing the Invariant Matrix Group

We start with the decomposition of single polynomials from which our region was built. Consider the polynomial

$$f(x,y) = \sum_{i,j=0}^{N} a_{i,j} x^i y^j$$

Let the transformation matrix be given by

$$G = \begin{bmatrix} g_{1,1} & g_{1,2} \\ g_{2,1} & g_{2,2} \end{bmatrix}$$

Then the polynomial remains invariant iff

$$f(G * \mathbf{x}) = f(G * (x,y)^T) = \sum_{i,j=0}^{N} a_{i,j} (g_{1,1}x + g_{1,2}y)^i (g_{2,1}x + g_{2,2}y)^j = f(\mathbf{x})$$

Exponentiating and collecting coefficients of like power leads to

$$0 f(G * \mathbf{x}) = \sum_{i,j=0}^{N} x^i y^j f_{i,j}(g_{1,1}, g_{1,2}, g_{2,1}, g_{2,2}) = \sum_{i,j=0}^{N} a_{i,j} x^i y^j = f(\mathbf{x}).$$

In this way we obtain the following system of N^2 equations ([21])

$$f_{i,j}(g_{1,1}, g_{1,2}, g_{2,1}, g_{2,2}) = a_{i,j},$$

where $f_{i,j}$ are some functions, which depend on $(g_{1,1}, g_{1,2}, g_{2,1}, g_{2,2})$ and can be computed, for example, with Maple, as follows:

```
# transform the polynomial according to (10) and expand it
 [> fG:=expand(f(op(convert(G.xx,list))),{x,y});

# calculate the coefficients of f(G*x)
```

```
[> f:=[coeffs(fG,{x,y})];

# calculate the coefficients of f(x)
[> a:=[coeffs(f,{x,y})];
```

The last step is to solve the system of equations to obtain the invariant matrix group.

For example, for the $Rect(x,y)\backslash circle(x,y) = -(x-2)*(y-2)*(x+2)*(y+2)*(x^2+y^2-1)$ we obtain the following system of equations

$$\begin{aligned}
-20 &= -20\,g_{1,1}{}^2 - 20\,g_{2,1}{}^2, 16 = 16, 4 = 4\,g_{1,1}{}^4 + 4\,g_{2,1}{}^4 + 9\,g_{1,1}{}^2 g_{2,1}{}^2,\\
9 &= 9\,g_{1,2}{}^2 g_{2,1}{}^2 + 24\,g_{2,1}{}^2 g_{2,2}{}^2 + 9\,g_{1,1}{}^2 g_{2,2}{}^2 + 36\,g_{1,1} g_{2,1} g_{2,2} g_{1,2} + 24\,g_{1,1}{}^2 g_{1,2}{}^2\\
,-1 &= -6\,g_{1,2}{}^2 g_{2,1}{}^2 g_{2,2}{}^2 - 8\,g_{1,1} g_{2,2}{}^3 g_{1,2} g_{2,1} - 6\,g_{1,1}{}^2 g_{2,2}{}^2 g_{1,2}{}^2 - g_{1,2}{}^4 g_{2,1}{}^2 -\\
& g_{1,1}{}^2 g_{2,2}{}^4 - 8\,g_{1,1} g_{2,1} g_{2,2} g_{1,2}{}^3,\\
-1 &= -8\,g_{1,1}{}^3 g_{2,1} g_{2,2} g_{1,2} - 6\,g_{1,1}{}^2 g_{2,2}{}^2 g_{2,1}{}^2 - 6\,g_{1,2}{}^2 g_{2,1}{}^2 g_{1,1}{}^2 - g_{1,2}{}^2 g_{2,1}{}^4 -\\
& g_{1,1}{}^4 g_{2,2}{}^2 - 8\,g_{1,2} g_{2,1}{}^3 g_{1,1} g_{2,2},\\
20 &= -20\,g_{1,2}{}^2 - 20\,g_{2,2}{}^2, 4 = 4\,g_{2,2}{}^4 + 4\,g_{1,2}{}^4 + 9\,g_{1,2}{}^2 g_{2,2}{}^2.
\end{aligned}$$

Note, we are looking for symmetric decomposition and, therefore, are interested in reflections groups only. According to [7] the following condition must be satisfied for any reflection transformation:

$$g_{1,1}g_{2,2} - g_{2,1}g_{1,2} = -1 \tag{4.8}$$

The system of equations (4.8), (4.2.2) has the solutions

$$\{g_{2,2}=0, g_{1,2}=1, g_{2,1}=1, g_{1,1}=0\}, \{g_{2,2}=0, g_{1,1}=0, g_{1,2}=-1, g_{2,1}=-1\},$$
$$\{g_{1,2}=0, g_{2,2}=1, g_{2,1}=0, g_{1,1}=-1\}, \{g_{1,2}=0, g_{2,1}=0, g_{2,2}=-1, g_{1,1}=1\},$$

which correspond to the reflection of part R_4 of G_4 given by:

$$R_4 = \left\{ \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \right\}.$$

Note that R_4 does not satisfy the closure property and, therefore, is not a group.

Obviously, the symmetry axes are given by those eigenvectors of these matrices, which correspond to the eigenvalue 1:

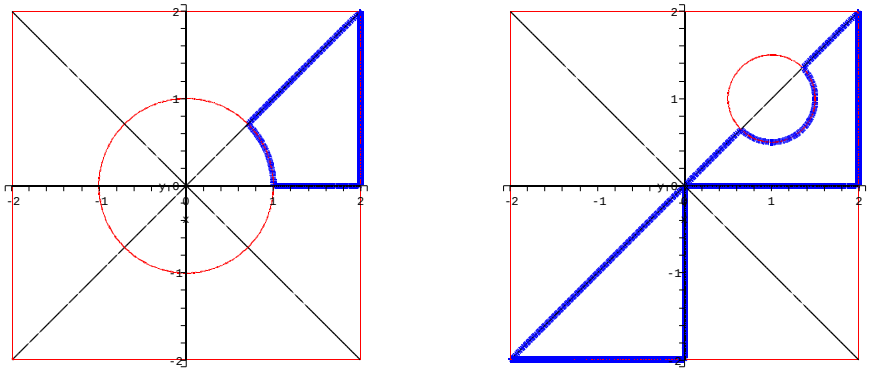


Figure 4.4: Decomposition of $Rect(x,y)\backslash circle(x,y)$ and $Rect(x,y)\backslash(circle(x-1/2, y-1/2)+3/4)$

$$\left\{\begin{bmatrix}0\\1\end{bmatrix},\begin{bmatrix}1\\0\end{bmatrix},\begin{bmatrix}1\\1\end{bmatrix},\begin{bmatrix}-1\\1\end{bmatrix}\right\}.$$

Solving $\mathbf{R}_i\mathbf{x}^T = \mathbf{x}^T$ we obtain four symmetry lines:

$$\begin{aligned} l_1(x,y) &= x,\\ l_2(x,y) &= y,\\ l_3(x,y) &= x-y,\\ l_4(x,y) &= x+y. \end{aligned} \tag{4.9}$$

As shown in Fig. 4.4, these four lines decompose the initial domain $O(x,y)$ given by

$O(x,y) = Rect(x,y)\backslash circle(x,y) = Rect(x,y) - circle(x,y) - \sqrt{Rect(x,y)^2 + circle(x,y)^2}$

in 8 congruent parts $O_i(x,y)$, which can be obtained using R-intersection (4.6) of O(x,y) and eight halfspaces given by 4 lines (4.9) as follows:

$$\begin{aligned} O_1(x,y) &= O(x,y) \cap l_2(x,y) \cap l_3(x,y)\\ O_2(x,y) &= O(x,y) \cap l_1(x,y) \cap -l_3(x,y)\\ O_3(x,y) &= O(x,y) \cap -l_1(x,y) \cap l_4(x,y)\\ O_4(x,y) &= O(x,y) \cap l_2(x,y) \cap -l_4(x,y)\\ O_5(x,y) &= O(x,y) \cap -l_2(x,y) \cap -l_3(x,y)\\ O_6(x,y) &= O(x,y) \cap -l_1(x,y) \cap l_3(x,y)\\ O_7(x,y) &= O(x,y) \cap l_1(x,y) \cap -l_4(x,y)\\ O_8(x,y) &= O(x,y) \cap -l_2(x,y) \cap l_4(x,y) \end{aligned} \tag{4.10}$$

In this way the finite symmetry group of simple geometric regions given as roots of polynomial equalities can be calculated. The decomposition of the region shown in Fig. 4.3 can be derived using symmetry axes of such primitive regions in the same way.

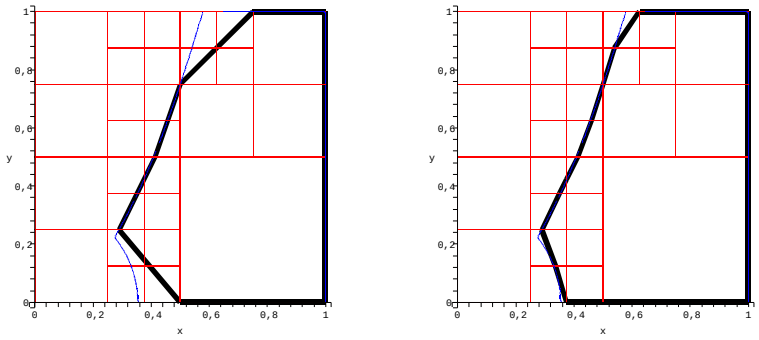


Figure 4.5: Curve approximation with quadtrees

```
while  AF ≠ ∅ do
       v := compute_next_candidate_vertex (E(a,b) ∈ AF, δ(x,y))
       while not visible(E(a,b), v) or min_angle(E ∪ (a,v) ∪ (v,b) ) ≤ θ_min
       or min_distance( V ∪ v) ) ≤ l_min do
       v := find_nearest_vertex (V, v)
       od:
    E := E ∪ {(a,v),(v,b)}
    V := V ∪ {v}
    AF := AF ∪ {(a,v),(v,b)} − {(a,b)}
    od:
```

Figure 4.6: A quasi-Maple description of the basic structure of advancing front algorithm

4.2.3 Boundary Discretization Using Quadtrees

As shown in section 4.2.2, the implicit curves can be used to describe complex geometric regions. In order to perform the advancing front triangulation needed for the FEM calculations on such regions enclosed by $R(x, y) = 0$ we need to partition the curve $R(x, y) = 0$ into linear segments. The term *quadtree* (or *octtree* in 3-dimensional case) is used to describe a well-known class of hierarchical data structures whose common property is that they are based on the principle of recursive decomposition of space [57]. As shown in Fig. 4.5 we start with the root rectangular element enclosing the geometric region of interest and subdivide it successively into four equal-sized quadrants. Each of these quadrants can be entirely contained in the region ($R(x, y) > 0$), entirely disjoint from it ($R(x, y) < 0$) or crossed by the boundary curve ($R(x, y)$ changes the sign along some quadrant edge). Checking the sign of $R(x, y)$ in the quadrant nodes one can determine the edges wich are crossed by the boundary curve and approximate the curve in the particular quadrant as shown in Fig. 4.5. The boundary quadrants can be successively subdivided to achieve a better approximation of the region boundary.

We have implemented a package `SpaceTrees` for Maple, that provides the following features:

- generating and refinement of quadtrees
- performing the set operations on quadtrees (union, intersection, difference)
- generating the discretization for numerical methods: both initial front for advancing front method described bellow and rectangular elements

This package is implemented in object-oriented way. For example, the following command generates the quadtree with top left corner with coordinates `0 , 1` and widths `1, 1` in x- and y-direction:

```
[> root:=quadtree(0, 1, 1, 1, O(x,y));
```

$O(x, y) = 0$ is the implicit function that bounds the region to be partitioned.
To refine the generated quadtree the method `refine` can be invoked:

```
[> ``||root||refine();
```

After several refinements one obtains the result shown in Fig. 4.5.
To obtain the approximation of $O(x, y) = 0$ corresponding to a particular depth of the quadtree use:

```
[> ``||root||getBoundary(depth_level);
```

The line segments generated in this way approximate boundary and are now used as initial front for the advancing front triangulation method described bellow.

4.2.4 Advancing Front Triangulation

This method [28] starts with the initial front $\mathcal{AF}$ obtained in the previous section. Then, it adds triangles into the domain, with at least one edge on the front. At each step, this will update the front. When the front is empty, the mesh generation is completed. This requires that the domain be bounded, but for unbounded domain the front can be advanced until it is at some large distance from the object. As the algorithm progresses, the front will advance to fill the remainder of the area with triangles.
In Fig. 4.6 the algorithm that we use is shown in more detail. Let three sets be given:

$\mathcal{AF}$	– current advancing front, consisting of edges
$\mathcal{V}$	– the set of all triangulation vertices
$\mathcal{E}$	– the set of oriented triangulation edges
$E(a, b)$	– the edge connecting vertices a and b

For each edge $E = (a, b) \in \mathcal{AF}$ of the front the algorithm calculates candidate vertex v lying in the vertex of an equilateral triangle with the base E. The triangles can be stretched by the user defined parameter $\delta(x, y)$. Adapting $\delta(x, y)$ the size of triangles can be adapted through the region.

`compute_next_candidate_vertex`(E, δ) - returns the point lying in the vertex of an equilateral triangle with the base E, to the left from E at the distance $dist(x, y)$

$\delta(x, y)$ - determines the stretching factor

Before new candidate edges (a, v), (v, b) are inserted in the current triangulation, we perform the intersection tests with existing edges using the procedure `visible`. Furthermore we calculate the minimal distance and minimal angle between (a, v), (v, b) and existing triangulation edges using `min_distance`, `min_angle`:

`visible(E::edge, v::vertex, s::set)`– tests, wether the generated edge is crossed by any other edge of the set `s`

`min_distance(s::set)` – computes the minimal distance between points of the set `s`

`min_angle(s::set)` – computes the minimal distance between edges of the set `s`

If (a, v), (v, b) does not intersect any other edge and minimal distance and angle condition are not violated, they will be added to the triangulation. If this is not the case, the next candidate vertex will be chosen from the existing triangulation vertices $\mathcal{V}$ using `find_nearst_vertex`:

`find_nearest_vertex(s::set, v::vertex)`– finds the vertex in `s` nearest to v

In this way the triangulation result depends on the choice of the following parameters:

l_{min} – the minimum distance allowed between vertices
θ_{min} – the minimum angle allowed between edges
$\delta(x, y)$ – stretching parameter

Compare, for example, grids obtained with $\delta(x, y) = 1$ (on the left hand side in Fig. 4.8) and δ given by

$$\begin{cases} 2\,x + \frac{17}{8} & x \le -\frac{13}{16} \\ 1 & \text{otherwise} \end{cases}$$

(on the right hand side in Fig. 4.8) In Figs. 4.7 and 4.9 the different grids for different choices of these parameters are shown. For two parts of the region of Fig.4.2 their triangulation as well as composite grid for the entire region are depicted in Fig. 4.9. In this case there are six symmetric subregions according to Fig. 2, but the advancing front triangu-

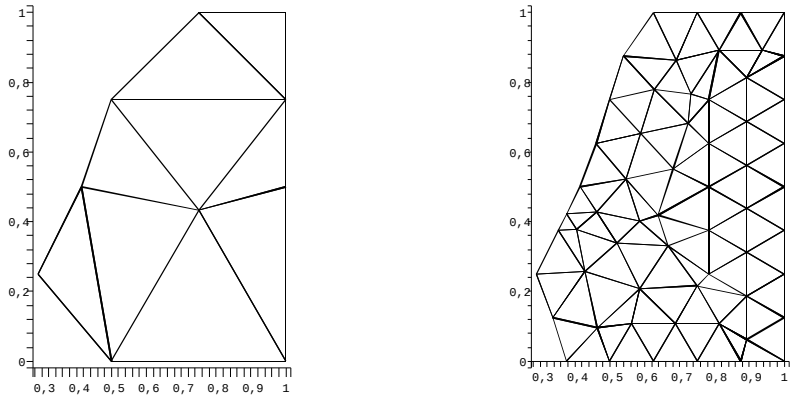


Figure 4.7: Advancing front triangulation with two different (coarse and fine) initial fronts

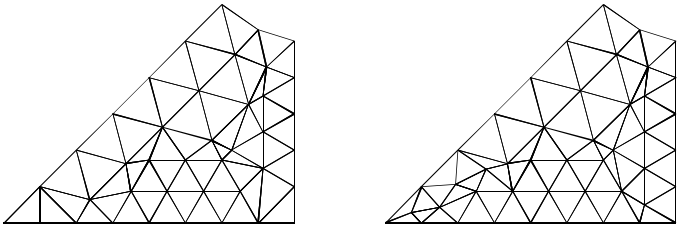
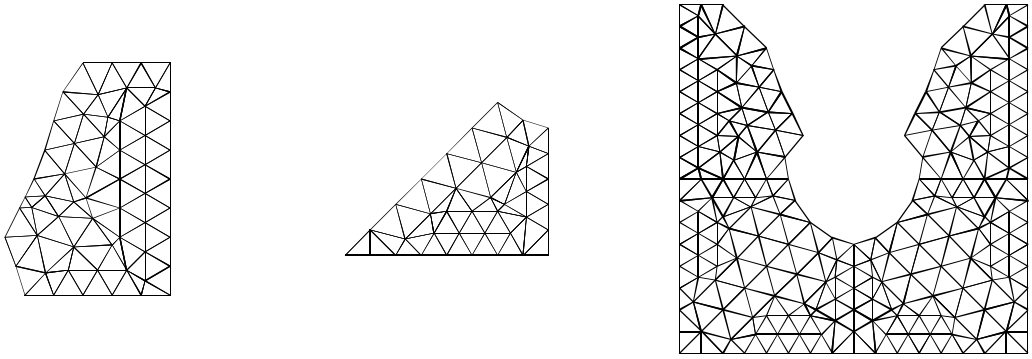


Figure 4.8: The triangulation of parts of our region obtained with different $\delta(x, y)$



(a) (b) (c)

Figure 4.9: The triangulation of parts of our region obtained with $\delta = 1, l_{min} = 0.3, \theta_{min} = 0.6$ (a), (b) and the derived triangulation of the whole region (c).

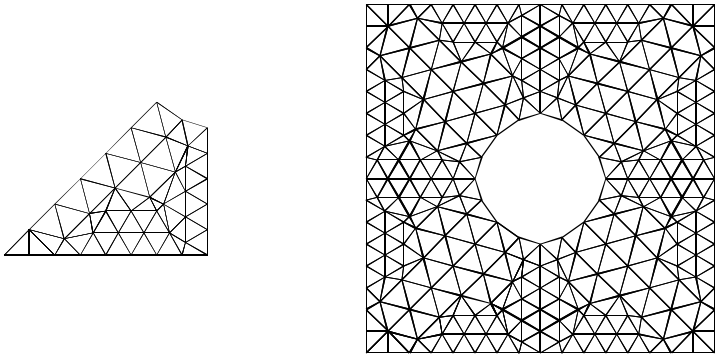


Figure 4.10: Decomposing of a channel with a cylinder leads to speed-up factor 8.

lation is executed only in two of them as shown in Fig. 4.9. The filling of the remaining symmetric counterpart subregions by a grid is a mere reflection, thus, no costly operations of the advancing front triangulation are performed at this reflection. Therefore, we can neglect the CPU time needed for these reflections. As a result, we obtain for the region of Fig. 4.9 the speed-up factor of $6/2 = 3$. For another region shown in Fig. 4.10 the speed-up factor is obviously equal to 8.

Chapter 5

Prediction of Fluid Motion

The Navier–Stokes equations governing two-dimensional unsteady flows of an incompressible viscous fluid may be written in vector form as follows:

$$\frac{\partial u}{\partial t} + \frac{\partial p}{\partial x} = \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) - \frac{\partial (u^2)}{\partial x} - \frac{\partial (uv)}{\partial y} + g_x, \tag{5.1}$$

$$\frac{\partial v}{\partial t} + \frac{\partial p}{\partial y} = \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) - \frac{\partial (uv)}{\partial x} - \frac{\partial (v^2)}{\partial y} + g_y \tag{5.2}$$

and *the continuity equation*

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{5.3}$$

The quantities to be found are

- $u : \Omega \times [0, t_{end}] \to R$, the fluid velocity in x–direction,
- $v : \Omega \times [0, t_{end}] \to R$, the fluid velocity in y–direction,
- $p : \Omega \times [0, t_{end}] \to R$ the pressure.
- g_x und g_y : $\Omega \times [0, t_{end}] \to R$ denote the external forces, either the Earth gravity or other body forces acting throughout the bulk of the system and producing acceleration in its parts.

The dimensionless real quantity Re is called *the Reynolds number*; it characterizes the fluid flow. The Reynolds number depends on viscosity and on average velocity of the fluid. The lower the Re value, the more viscous is the fluid.

At the initial moment ($t = 0$), initial conditions $u = u_0(x, y)$ and $v = v_0(x, y)$ satisfying (5.3) are given. Besides, supplementary conditions holding at all four boundaries of the region for all times are required, so that we arrive at an *initial-boundary value problem.*

There exist a number of approaches for the discretization of the Navier-Stokes equations. A stable finite difference method is based on using so called staggered grids, where the unknown variables u, v and p lie at different grids shifted with respect to each other.

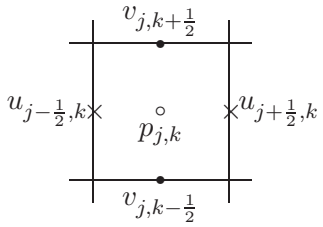


Figure 5.1: The staggered grid in two dimensions.

Primarily, a reference grid is chosen whose lines subdivide the whole region into cells. The cell characterized by index (i, j) corresponds to the rectangle $[(i-1)\,\delta x, i\,\delta x] \times [(j-1)\,\delta y, j\,\delta y]$. In such a grid the pressure p will be related to the cell midpoint, the u velocity at the midpoint of vertical cell edges, and the v velocity at the midpoints of the horizontal cell edges. The index (i, j) is assigned to the pressure at the cell center as well as to the u-velocity at the right edge and the v-velocity at the upper edge of (i, j) cell. Thus, the pressure $p_{i,j}$ is located at coordinate points $((i-0.5)\,\delta x, (j-0.5)\,\delta y)$, the horizontal velocity $u_{i,j}$ at points with the coordinates $(i\,\delta x, (j-0.5)\,\delta y)$ and the vertical velocity $v_{i,j}$ at coordinate points $((i-0.5)\,\delta x, j\,\delta y)$. So the points for u, v and p belong to three different grids (lattices), each being shifted by half a lattice period with respect to the reference grid.

In order to discretize the continuity equation on staggered grid one approximates the second derivatives by taking finite differences of the first derivatives. Such operation gives:

$$\begin{aligned}\left[\frac{\partial^2 u}{\partial x^2}\right]_{i,j} &:= \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\delta x)^2},\\ \left[\frac{\partial^2 u}{\partial y^2}\right]_{i,j} &:= \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\delta y)^2}.\end{aligned}$$

This discretization leads to a large system of linear equations, which can be approximately solved by a variety of methods developed in numerical mathematics (see below).

Now the Navier-Stokes equations can be discretized in the following fashion: at first, we shall handle the spatial derivatives. The momentum equation (5.1) will be evaluated at the vertical edge midpoints, the momentum equation (5.2) at the horizontal edge midpoints, and the continuity equation (5.3) at the cell midpoints. Thus, it remains to be proved that equation (5.1) is associated with velocity u, equation (5.2) with velocity v, and equation (5.3) with pressure p considered as unknowns. We replace the expression in equation (5.1) taken at the midpoint of the right edge of cell (i, j), $i = 1, \ldots, imax - 1$, $j = 1 \ldots, jmax$, by

$$\begin{aligned}\left[\frac{\partial(u^2)}{\partial x}\right]_{i,j} := \frac{1}{\delta x}\left(\left(\frac{u_{i,j}+u_{i+1,j}}{2}\right)^2 - \left(\frac{u_{i-1,j}+u_{i,j}}{2}\right)^2\right) +\\ \alpha\frac{1}{\delta x}\left(\frac{|u_{i,j}+u_{i+1,j}|}{2}\frac{(u_{i,j}-u_{i+1,j})}{2} - \frac{|u_{i-1,j}+u_{i,j}|}{2}\frac{(u_{i-1,j}-u_{i,j})}{2}\right),\end{aligned}$$

$$\left[\frac{\partial(uv)}{\partial y}\right]_{i,j} := \frac{1}{\delta y}\left(\frac{(v_{i,j}+v_{i+1,j})}{2}\frac{(u_{i,j}+u_{i,j+1})}{2} - \frac{(v_{i,j-1}+v_{i+1,j-1})}{2}\frac{(u_{i,j-1}+u_{i,j})}{2}\right) + \alpha\frac{1}{\delta y}\left(\frac{|v_{i,j}+v_{i+1,j}|}{2}\frac{(u_{i,j}-u_{i,j+1})}{2} - \frac{|v_{i,j-1}+v_{i+1,j-1}|}{2}\frac{(u_{i,j-1}-u_{i,j})}{2}\right) \quad (5.4)$$

$$\left[\frac{\partial^2 u}{\partial x^2}\right]_{i,j} := \frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{(\delta x)^2},$$

$$\left[\frac{\partial^2 u}{\partial y^2}\right]_{i,j} := \frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{(\delta y)^2}, \qquad \left[\frac{\partial p}{\partial x}\right]_{i,j} := \frac{p_{i+1,j}-p_{i,j}}{\delta x}.$$

For the expressions in equation (5.2), we set at the midpoint of the upper edge of cell (i,j), $i = 1,\ldots,imax$, $j = 1\ldots,jmax-1$

$$\left[\frac{\partial(uv)}{\partial x}\right]_{i,j} := \frac{1}{\delta x}\left(\frac{(u_{i,j}+u_{i,j+1})}{2}\frac{(v_{i,j}+v_{i+1,j})}{2} - \frac{(u_{i-1,j}+u_{i-1,j+1})}{2}\frac{(v_{i-1,j}+v_{i,j})}{2}\right) + \alpha\frac{1}{\delta x}\left(\frac{|u_{i,j}+u_{i,j+1}|}{2}\frac{(v_{i,j}-v_{i+1,j})}{2} - \frac{|u_{i-1,j}+u_{i-1,j+1}|}{2}\frac{(v_{i-1,j}-v_{i,j})}{2}\right),$$

$$\left[\frac{\partial(v^2)}{\partial y}\right]_{i,j} := \frac{1}{\delta y}\left(\left(\frac{v_{i,j}+v_{i,j+1}}{2}\right)^2 - \left(\frac{v_{i,j-1}+v_{i,j}}{2}\right)^2\right) + \alpha\frac{1}{\delta y}\left(\frac{|v_{i,j}+v_{i,j+1}|}{2}\frac{(v_{i,j}-v_{i,j+1})}{2} - \frac{|v_{i,j-1}+v_{i,j}|}{2}\frac{(v_{i,j-1}-v_{i,j})}{2}\right), \quad (5.5)$$

$$\left[\frac{\partial^2 v}{\partial x^2}\right]_{i,j} := \frac{v_{i+1,j}-2v_{i,j}+v_{i-1,j}}{(\delta x)^2},$$

$$\left[\frac{\partial^2 v}{\partial y^2}\right]_{i,j} := \frac{v_{i,j+1}-2v_{i,j}+v_{i,j-1}}{(\delta y)^2}, \qquad \left[\frac{\partial p}{\partial y}\right]_{i,j} := \frac{p_{i,j+1}-p_{i,j}}{\delta y}$$

The terms in the continuity equation (5.3) are replaced at the midpoint of cell (i,j), $i = 1,\ldots,imax$, $j = 1\ldots,jmax$, by

$$\left[\frac{\partial u}{\partial x}\right]_{i,j} := \frac{u_{i,j}-u_{i-1,j}}{\delta x}, \qquad \left[\frac{\partial v}{\partial y}\right]_{i,j} := \frac{v_{i,j}-v_{i,j-1}}{\delta y}. \quad (5.6)$$

Here α is a parameter with values between 0 and 1. For $\alpha = 0$, one gets the second-order approximation for differential operators, i.e. the approximation error has the accuracy $O((\delta x)^2)$ or $O((\delta y)^2)$. However, for small viscosity values, this approximation can result in oscillations in the solution. In such cases one must resort to a so-called donor-cell scheme ($\alpha = 1$), which only produces the first-order approximation. In practice, a mixture of both techniques is used, with $\alpha \in [0,1]$. The parameter α should be selected slightly larger than the maximal grid value of $|u\,\delta t/\delta x|$ and $|v\,\delta t/\delta y|$.

While discretizing the momentum equations (5.1) and (5.2) with respect to time, the terms on the left-hand side should be evaluated at time point t_{n+1} and those on the right-hand side at time point t_n. The time derivative $\partial u/\partial t$ taken at time t_{n+1} will be replaced

by a finite difference quotient of the first order, $(u^{(n+1)} - u^{(n)})/\delta t$. This corresponds to the explicit time-stepping Euler scheme. There exist also implicit schemes, where on the right-hand side also the values appear which are associated with future temporal points, t_{n+1}. Implicit methods allow considerably bigger time steps. However, for each time step a large (nonlinear) system of equations arises, and the necessity to solve the latter almost totally annihilates the advantages of applying bigger time steps..

The continuity equation (5.3) will be related to time point t_{n+1}.

The Algorithm The method of solution of the discrete equations obtained above can be described by the following sequence of steps.

The Time-Stepping Loop

In the outer iteration loop, the time is incremented at each step, starting from the time point $t = 0$, by a given δt, until the the final time point t_{end} is attained. At each step n $(n = 0, 1, \ldots)$ the differential equations will be discretized as described above. The values of all variables at any time step t_n are known, and those at time t_{n+1} should be computed.

The Discrete Momentum Equations

The discrete momentum equations at each time step are to be solved again for newly determined velocities $u_{i,j}^{(n+1)}$ and $v_{i,j}^{(n+1)}$. Then one obtains

$$u_{i,j}^{(n+1)} = F_{i,j}^{(n)} - \frac{\delta t}{\delta x}(p_{i+1,j}^{(n+1)} - p_{i,j}^{(n+1)}) \tag{5.7}$$
$$i = 1, \ldots, imax - 1, \quad j = 1, \ldots, jmax;$$
$$v_{i,j}^{(n+1)} = G_{i,j}^{(n)} - \frac{\delta t}{\delta y}(p_{i,j+1}^{(n+1)} - p_{i,j}^{(n+1)}) \tag{5.8}$$
$$i = 1, \ldots, imax, \quad j = 1, \ldots, jmax - 1.$$

Here, the terms $F_{i,j}^{(n)}$ und $G_{i,j}^{(n)}$ contain the discretized right-hand sides of momentum equations (5.1) and (5.2) as well as the current time-level velocities u^n and v^n. Using the discretized equations (5.4) and (5.5), we obtain

$$F_{i,j} := u_{i,j} + \delta t \left(\frac{1}{Re} \left(\left[\frac{\partial^2 u}{\partial x^2} \right]_{i,j} + \left[\frac{\partial^2 u}{\partial y^2} \right]_{i,j} \right) - \left[\frac{\partial (u^2)}{\partial x} \right]_{i,j} - \left[\frac{\partial (uv)}{\partial y} \right]_{i,j} + g_x \right) \tag{5.9}$$
$$i = 1, \ldots, imax - 1, \quad j = 1, \ldots, jmax;$$
$$G_{i,j} := v_{i,j} + \delta t \left(\frac{1}{Re} \left(\left[\frac{\partial^2 v}{\partial x^2} \right]_{i,j} + \left[\frac{\partial^2 v}{\partial y^2} \right]_{i,j} \right) - \left[\frac{\partial (uv)}{\partial x} \right]_{i,j} - \left[\frac{\partial (v^2)}{\partial y} \right]_{i,j} + g_y \right) \tag{5.10}$$
$$i = 1, \ldots, imax, \quad j = 1, \ldots, jmax - 1.$$

The pressure Equation

Equations (5.7) and (5.8) give the closed formulae to determine the new velocities $u_{i,j}^{(n+1)}$ und $v_{i,j}^{(n+1)}$ in terms of the old velocities $u_{i,j}^{(n)}$ and $v_{i,j}^{(n)}$. However, the pressure $p^{(n+1)}$ remains

so far unknown. Before the quantities $u_{i,j}^{(n+1)}$ and $v_{i,j}^{(n+1)}$ could be determined, one should first provide an equation for $p^{(n+1)}$ and solve it. Such equation can be obtained by putting the expressions (5.7) and (5.8) for $u_{i,j}^{(n+1)}$ und $v_{i,j}^{(n+1)}$ into the discrete continuity equation related to time point t_{n+1}:

$$\frac{p_{i+1,j}^{(n+1)} - 2p_{i,j}^{(n+1)} + p_{i-1,j}^{(n+1)}}{(\delta x)^2} + \frac{p_{i,j+1}^{(n+1)} - 2p_{i,j}^{(n+1)} + p_{i,j-1}^{(n+1)}}{(\delta y)^2} = \frac{1}{\delta t}\left(\frac{F_{i,j}^{(n)} - F_{i-1,j}^{(n)}}{\delta x} + \frac{G_{i,j}^{(n)} - G_{i,j-1}^{(n)}}{\delta y}\right), \quad i = 1, \ldots, imax, \quad j = 1, \ldots, jmax. \tag{5.11}$$

This is the familiar form of the discretized Poisson equation for the quantity $p^{(n+1)}$

$$\frac{\partial^2 p^{(n+1)}}{\partial x^2} + \frac{\partial^2 p^{(n+1)}}{\partial y^2} = rs$$

on a domain Ω, with an arbitrary right-hand side rs. To ensure the uniqueness of the solution, we also need the boundary conditions $p_{i,j}$ ($i \in \{0, imax+1\}, j \in \{0, jmax+1\}$), $F_{i,j}$ ($i \in \{0, imax\}$) und $G_{i,j}$ ($j \in \{0, jmax\}$), which we can obtain from the momentum equations being considered at the boundary (see below).

Now it is possible to solve the pressure equation (5.11) using any solution techniques developed for systems of linear equations. Since the direct methods, as e.g. Gauss elimination, lead to high computational costs for large problems (here $imax \cdot jmax$ is unknown), it is more customary to use iterative procedures while solving numerically partial differential equations. An example is the Gauss-Seidel method, in which, starting from some initial value, all the cells are successively updated in each cycle, and the pressure at (i, j) cell is adjusted in such a way that the corresponding equation should be exactly satisfied.

An improved variant is given by the SOR (successive over-relaxation) method, when the iteration step is given by the following loop over all cells:

$$\begin{aligned} &i = 1, \ldots, imax \\ &j = 1, \ldots, jmax \\ &p_{i,j}^{it+1} := (1-\omega)\, p_{i,j}^{it} + \frac{\omega}{2(\frac{1}{(\delta x)^2} + \frac{1}{(\delta y)^2})}\left(\frac{p_{i+1,j}^{it} + p_{i-1,j}^{it+1}}{(\delta x)^2} + \frac{p_{i,j+1}^{it} + p_{i,j-1}^{it+1}}{(\delta y)^2} - rs_{i,j}\right) \end{aligned} \tag{5.12}$$

The upper indices it und $it+1$ designate the iteration step number. Important: the old pressure value p^{it} will be right away overwritten by the updated value, $pit+1$, i.e. there remains no saved copy of the pressure field.

The quantity $rs_{i,j}$ is the right-hand side of the pressure equation (5.11 for the (i, j) cell and ω is a parameter (relaxation factor), which must be chosen from the interval $]0, 2]$

(often the value $\omega = 1.7$ is used). For $\omega = 1$, the method is reduced to that of Gauss-Seidel. The iteration process stops either once the maximum number of iterations, $itmax$, is reached or when the residual

$$res := \left(\sum_{i=1}^{imax} \sum_{j=1}^{jmax} \left(\frac{p_{i+1,j} - 2p_{i,j} + p_{i-1,j}}{(\delta x)^2} + \frac{p_{i,j+1} - 2p_{i,j} + p_{i,j-1}}{(\delta y)^2} - rs_{i,j} \right)^2 / (imax \cdot jmax) \right)^{1/2} \tag{5.13}$$

becomes smaller than the tolerance value ε defined by the user.

As a starting value for the iteration process to calculate the pressure $p^{(n+1)}$, any pressure value related to time level n can be taken.

Using the calculated pressure values related to time point t_{n+1}, one can then, with the help of (5.7) and (5.8), compute the velocity values u and v for time point t_{n+1}

The Stability Condition In order to ensure the stability of the numerical algorithm and avoid oscillations, the following three stability conditions must be imposed on the stepsizes δx, δy, and δt:

$$\frac{2}{Re}\,\delta t < \frac{(\delta x)^2 (\delta y)^2}{(\delta x)^2 + (\delta y)^2}, \qquad |u_{max}|\,\delta t < \delta x, \qquad |v_{max}|\,\delta t < \delta y. \tag{5.14}$$

Here $|u_{max}|$ and $|v_{max}|$ are the maximal absolute values of the respective velocities. The latter two inequalities in (5.14) are called the Courant–Friedrichs–Levi (CFL) conditions.

One can use an adaptive stepsize control based on the above stability conditions. This is implemented by choosing δt for the next time step in such a way that each of the three conditions (5.14) is satisfied:

$$\delta t := \tau \, \min \left(\frac{Re}{2} \left(\frac{1}{\delta x^2} + \frac{1}{\delta y^2} \right)^{-1}, \frac{\delta x}{|u_{max}|}, \frac{\delta y}{|v_{max}|} \right). \tag{5.15}$$

The coefficient $\tau \in]0, 1]$ is a safety factor. This stepsize control ensures, however, only the stability of the metod. In order to specify the accuracy, the stepsize control should be based on some error estimation procedure, which allows one to appraise the difference between numerical and analytical solutions.

5.1 Second Order Approximation

During the last decade, a new method for numerical solution of the Navier-Stokes equations in regions with complex geometry has enjoyed a powerful development: the immersed boundary method (IBM). In this method, the computation of gas motion is carried out on a rectangular grid, and the curved boundary is interpreted as an interface. The grid cells lying outside the region occupied by the fluid are classified as the ghost cells in which the Navier–Stokes equations are, however, also solved numerically. A survey of different recent

realizations of the IBM may be found in [46, 65, 68]. The immersed boundary method has extended significantly the scope of applicability of the rectangular Cartesian grids at the numerical solution of applied problems of the incompressible fluid dynamics.

The difference scheme proposed in [37] is often used within the IBM framework. The convective terms are approximated in this scheme with the aid of the explicit three-level Adams–Bashforth second-order scheme, and the viscous terms are approximated by the implicit second-order Crank–Nicolson scheme. Despite the popularity of scheme [37], its stability was not investigated even in the case of two spatial variables.

The purpose of the present work is the stability investigation of a modified scheme from [37]. This investigation is carried out at first by the Fourier method. Since this analysis method is applicable only to linear difference schemes with constant coefficients we employ one more method for stability analysis of nonlinear difference equations approximating the Navier–Stokes equations. This method was proposed in [47] and reduces to the investigation of the behaviour of solution of difference equations in the case when the oscillating velocity profiles are specified on two lower time levels. The obtained stability conditions have been verified by computations of two test problems one of which is the lid-driven cavity problem.

Following [37] we will discretize the momentum equation in time by using a hybrid second-order scheme:

$$\frac{\vec{v}^* - \vec{v}^n}{\tau} + \frac{3}{2}H(\vec{v}^n) - \frac{1}{2}H(\vec{v}^{n-1}) + \frac{1}{\rho}Gp^n = \frac{\nu}{2}\left[L(\vec{v}^*) + L(\vec{v}^n)\right]. \tag{5.16}$$

Here τ is the time step, $H(\vec{v}^n)$ is the difference operator approximating the operator $(\vec{v}\nabla)\vec{v}$, G is the discrete gradient, L is the discrete Laplace operator, n is the time level. Thus, the convective terms in (5.16) are approximated explicitly by the second-order Adams–Bashforth scheme, and the diffusion terms $\nu\Delta\vec{v}$ are treated implicitly using second-order Crank–Nicolson scheme. The implicit approximation of viscous terms is applied according to [37] in order to eliminate a restriction for time step τ dictated by the computational stability.

At the second fractional step, the field of intermediate velocities $\vec{v}^*$ is corrected to ensure the mass conservation:

$$(\vec{v}^{n+1} - \vec{v}^*)/\tau_n = -Gp', \tag{5.17}$$

The pressure correction p' is computed in such a way that a divergence-free velocity field is obtained at the $(n+1)$th time step. To this end, let us apply the divergence operator to the both sides of equation (5.17):

$$(D\vec{v}^{n+1} - D\vec{v}^*)/\tau_n = -Lp', \tag{5.18}$$

where D is a discrete analog of the divergence operator. Since it is required that $D\vec{v}^{n+1} = 0$, we obtain from (5.18) the Poisson equation for the pressure correction:

$$Lp' = (1/\tau_n)D\vec{v}^*. \tag{5.19}$$

The correction p' found as the solution of equation (5.19) is then used for the correction of the velocity field according to (5.17): $\vec{v}^{n+1} = \vec{v}^{*} - \tau_n G p'$ and of the pressure field: $p^{n+1} = p^n + p'$. The Poisson equation (5.19) was solved by the BiCGSTAB method [66]. As was pointed out in [38], the pressure correction method was found to be the fastest of the methods tested by Armfield and Street [2] and is the method used here.

Following [37] we will approximate all spatial derivatives by second-order central differences on a staggered grid (see Fig. 5.1). The advantages of staggered grid at the numerical integration of the Navier–Stokes equations for incompressible fluid are discussed in detail in [37, 38]. For example, the term $\partial^2 v/\partial y^2$ is approximated on the staggered grid as follows:

$$\left(\partial^2 v/\partial y^2\right)_{j,k+1/2} = (v_{j,k+3/2} - 2v_{j,k+1/2} + v_{j,k-1/2})/(h_2^2),$$

where h_1, h_2 are the steps of uniform rectangular grid along the x- and y-axes, respectively; the subscripts j, k refer to the cell center. To approximate the convective terms $H(\vec{v}^n)$ we use in (5.16) the difference formulas of the MAC-method [29, 55, 37]. These formulas are applied to the divergence form of motion equations:

$$\frac{\partial u}{\partial t} + \frac{\partial (u^2)}{\partial x} + \frac{\partial uv}{\partial y} + \frac{1}{\rho}\frac{\partial p}{\partial x} = \nu \Delta u; \quad \frac{\partial v}{\partial t} + \frac{\partial uv}{\partial x} + \frac{\partial (v^2)}{\partial y} + \frac{1}{\rho}\frac{\partial p}{\partial y} = \nu \Delta v.$$

For example, $(\partial u^2/\partial x)_{j+1/2,k} = (u_{j+1,k}^2 - u_{j,k}^2)/h_1$, where $u_{j,k} = (1/2)(u_{j-1/2,k} + u_{j+1/2,k})$.

We now mention several stability conditions, which were used previously at the computation of time step τ entering the difference scheme (5.16). Roache [55] discussed the stability of the Adams–Bashforth scheme at its application for approximation of the one-dimensional advection-diffusion equation

$$\partial \zeta/\partial t + \partial (u\zeta)/\partial x = \nu \partial^2 \zeta/\partial x^2. \tag{5.20}$$

This scheme proved to be unconditionally unstable, and it has a weak divergence caused by the fact that the scheme amplification factor G obtained by the Fourier method has the form $G = 1 + O(\tau^2)$. It is, however, to be noted that the above scheme from [55] for equation (5.20) is explicit, whereas there are in scheme (5.16) also the implicit operators, which stabilize the numerical computation. It is to be noted here that since $\nu = O(1/\mathrm{Re})$, where Re is the Reynolds number, then at high Reynolds numbers, the stabilizing effect of the implicit term in (5.16) becomes insignificant. The computation nevertheless remains stable at the solution of practical problems by scheme (5.16) also for the value Re = 25 000, as this was shown in [35]. It was proposed in [35] to compute the time step τ at the computation by scheme (5.16) by using the formula

$$\tau = \min_{j,k} \left[\tau_{conv}^{-1}/C_{conv} + \tau_{diff}^{-1}/C_{diff}\right]^{-1}, \tag{5.21}$$

where the items are computed in each (j, k) cell as follows:

$$\tau_{conv}^{-1} = |u|/h_1 + |v|/h_2, \quad \tau_{diff}^{-1} = \nu \cdot \left(1/h_1^2 + 1/h_2^2\right).$$

For the diffusion component in (5.21) the Courant number $C_{diff} = 0.25$ according to [35], and for the convective component the values of C_{conv} were taken from 0.5 to 1. Note that formula (5.21) is similar to the one used in [6], but in [6], the common Courant number $C_{conv} = C_{diff} = 0.25$ was used. Owing to the application of formula (5.21) with different values of C_{conv} and C_{diff} the authors of [35] were able to reduce the required CPU time at the computations of unsteady flows by a factor of nearly four.

The stability analysis results were presented in [27] for the schemes of Runge–Kutta type with the stage numbers three and five for the two-dimensional advection-diffusion equation

$$\partial f/\partial t + u\partial f/\partial x + v\partial f/\partial y = \nu(\partial^2 f/\partial x^2 + \partial^2 f/\partial y^2).$$

It turned out that for the both studied schemes, the stability condition has the form

$$\left(\frac{|\kappa_1| + |\kappa_2|}{a}\right)^2 + \left(\frac{\kappa_3'}{b}\right)^2 \le 1, \tag{5.22}$$

where $\kappa_1 = u\tau/h_1$, $\kappa_2 = v\tau/h_2$, $\kappa_3' = \nu\tau(1/h_1^2 + 1/h_2^2) = \kappa_3(1 + \kappa_4^2)$, $\kappa_3 = \nu\tau/(h_1^2)$, $\kappa_4 = h_1/h_2$, a and b are certain constants depending on the specific method of the Runge–Kutta type. Despite the fact that condition (5.22) as well as the empirical stability condition (5.21) were obtained for different difference schemes their structure is similar. Formula (5.21) can indeed be written in terms of dimensionless quantities κ_1, κ_2 and κ_3' as

$$\frac{|\kappa_1| + |\kappa_2|}{C_{conv}} + \frac{\kappa_3'}{C_{diff}} \le 1.$$

5.1.1 Fourier Symbol

The stability analysis of difference schemes by the Fourier method is known to be applicable only to linear schemes with constant coefficients. Difference scheme (5.16) is nonlinear, therefore, prior to the Fourier method application it is necessary to linearize the scheme. Linearization may be implemented in two different ways. One of them consists of that the original differential equations (in our case these are equations (5.1), (5.1), (5.3) are at first linearized, and the difference scheme (5.16) is then applied to linearized differential equations. Another technique reduces to a direct linearization of difference equations (5.16). We use the first of the above techniques because it involves a slightly shorter calculation.

Thus, let us assume that $U(x, y, t), V(x, y, t), P(x, y, t)$ is an exact solution of equation (5.1), (5.1), (5.3), where U and V are the components of the velocity vector along the x- and y-axes, respectively, P is the pressure. According to difference equation (5.16), only the velocity components are varied at a passage from the nth time level to the $(n + 1)$th time level. We can, therefore, present solution $\vec{v}$ of system (5.1), (5.1), (5.3) as

$$u = U + \delta u, \quad v = V + \delta v, \quad p = P, \tag{5.23}$$

where δu and δv are the errors, which are small in their absolute values and which are caused by the approximation error, machine roundoff errors, etc. Since the "big" quantities

U, V, P satisfy equation (5.1), (5.1), (5.3), as a result of substituting formulas (5.23) in (5.1), (5.1), (5.3) and neglecting the second-order terms with respect to δu and δv we obtain the following linear differential equations:

$$\begin{aligned} &\frac{\partial \delta u}{\partial t} + U\frac{\partial \delta u}{\partial x} + V\frac{\partial \delta u}{\partial y} = \nu\left(\frac{\partial^2 \delta u}{\partial x^2} + \frac{\partial^2 \delta u}{\partial y^2}\right); \\ &\frac{\partial \delta v}{\partial t} + U\frac{\partial \delta v}{\partial x} + V\frac{\partial \delta v}{\partial y} = \nu\left(\frac{\partial^2 \delta v}{\partial x^2} + \frac{\partial^2 \delta v}{\partial y^2}\right). \end{aligned} \tag{5.24}$$

Let us now approximate system (5.24) by difference scheme (5.16) on a staggered grid. Since this difference scheme is a three-level scheme we introduce two auxiliary dependent variables δr^n and δs^n by formulas [26]: $\delta r^n = \delta u^{n-1}$, $\delta s^n = \delta v^{n-1}$ before the investigation of its stability. Let $\vec{V} = (U, V)^T$, $\delta\vec{v}^n = (\delta u^n, \delta v^n)^T$, $\delta\vec{r}^n = (\delta r^n, \delta s^n)^T$. We can then write difference scheme (5.16) as applied to system (5.24) in the form:

$$\frac{\delta\vec{v}^* - \delta\vec{v}^n}{\tau} + \frac{3}{2}(\vec{V}^n\nabla)\delta\vec{v}^n - \frac{1}{2}(\vec{V}^{n-1}\nabla)\delta\vec{r}^n = \frac{\nu}{2}\left[L(\delta\vec{v}^*) + L(\delta\vec{v}^n)\right]. \tag{5.25}$$

Thus, (5.25) is a two-layer difference scheme. Upon "freezing" its coefficients $\vec{V}^n$, $\vec{V}^{n-1}$ we can apply the von Neumann stability analysis [25, 26] to obtain the necessary stability condition. According to the procedure of this analysis we substitute into the system of difference equations

$$\begin{aligned} &\frac{\delta\vec{v}^* - \delta\vec{v}^n}{\tau} + \frac{3}{2}(\vec{v}^n\nabla)\delta\vec{v}^n - \frac{1}{2}(\vec{V}^{n-1}\nabla)\delta\vec{r}^n = \frac{\nu}{2}\left[L(\delta\vec{v}^*) + L(\delta\vec{v}^n)\right]; \\ &\delta r^{n+1} = \delta u^n; \\ &\delta s^{n+1} = \delta v^n \end{aligned} \tag{5.26}$$

the solution of the form

$$\delta\vec{w}^n_{j,k} = \delta\vec{w}_0\lambda^n \exp[i(jm_1h_1 + km_2h_2)], \tag{5.27}$$

where $\delta\vec{w}^n = (\delta u^n, \delta v^n, \delta r^n, \delta s^n)^T$, $\delta\vec{w}_0$ is a constant vector, m_1 and m_2 are real components of the wave vector, λ is a complex number, $i = \sqrt{-1}$. As a result of the substitution of particular solution of the form (5.27) into system (5.26) we obtain the system

$$A\delta\vec{w}^{n+1}_{j,k} = B\delta w^n_{j,k}, \tag{5.28}$$

where

$$A = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & a & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} b & 0 & c & 0 \\ 0 & b & 0 & c \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix},$$

$$\begin{aligned} &a = 1 + \kappa_3(1 - \cos\xi) + \kappa_4(1 - \cos\eta), \quad c = (1/2)i(\kappa_1 \sin\xi + \kappa_2 \sin\eta), \\ &b = 1 - 3c - \kappa_3(1 - \cos\xi) - \kappa_4(1 - \cos\eta), \end{aligned} \tag{5.29}$$

$$\kappa_1 = \frac{U\tau}{h_1},\ \kappa_2 = \frac{V\tau}{h_2},\quad \kappa_3 = \frac{\nu\tau}{h_1^2},\ \kappa_4 = \frac{\nu\tau}{h_2^2}, \tag{5.30}$$

$\xi = m_1 h_1$, $\eta = m_2 h_2$. The quantities κ_3 and κ_4 are nonnegative by virtue of their physical meaning, therefore, $a \geq 1$, and, hence, matrix A is invertible. Multiplying the both sides of equation (5.28) from the left by A^{-1} we obtain the system

$$\delta\vec{w}_{j,k}^{n+1} = G\,\delta\vec{w}_{j,k}^{n}, \tag{5.31}$$

where matrix $G = A^{-1}B$ is called the amplification matrix of the difference scheme with constant coefficients. But in our case, the coefficients depend on x, y, and t with regard for (5.30). Therefore, we will consider in the following the matrix G in (5.31) for fixed values of x, y, t and will term the corresponding matrix G the Fourier symbol of the difference scheme.

All analytic formulas presented in this section and in the next section can be obtained with the aid of the computer algebra system. In particular,

$$G = A^{-1}B = \begin{pmatrix} \frac{b}{a} & 0 & \frac{c}{a} & 0 \\ 0 & \frac{b}{a} & 0 & \frac{c}{a} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}. \tag{5.32}$$

Denote by $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ the eigenvalues of matrix G. The von Neumann necessary stability conditions then have the form [25]

$$|\lambda_m| \leq 1 + O(\tau), \quad m = 1, \ldots, 4. \tag{5.33}$$

Let $\vec{\kappa} = (\kappa_1, \kappa_2, \kappa_3, \kappa_4)$. We have found the expression for the characteristic polynomial $f(\lambda, \vec{\kappa}, \xi, \eta) = \mathrm{Det}(G - \lambda I)$ of matrix G, where I is the identity matrix, with the aid of the CAS we obtain:

$$f(\lambda, \vec{\kappa}, \xi, \eta) = \frac{(a\lambda^2 - b\lambda - c)^2}{a^2}. \tag{5.34}$$

This equation has two roots λ_1, λ_2, and the multiplicity of each of these roots is equal to two:

$$\lambda_1 = \frac{b - \sqrt{b^2 + 4ac}}{2a}, \quad \lambda_2 = \frac{b + \sqrt{b^2 + 4ac}}{2a}. \tag{5.35}$$

5.1.2 Analytic Investigation of Eigenvalues

We first consider the particular case of creeping fluid flows when $U \approx 0$, $V \approx 0$. Assuming then $\kappa_1 = \kappa_2 = 0$ we obtain the following expression for λ_2: $\lambda_2 = (1 - \sigma)/(1 + \sigma)$, where $\sigma = 2[\kappa_3 \sin^2(\xi/2) + \kappa_4 \sin^2(\eta/2)] \geq 0$. It is easy to be sure of the fact that $|\lambda_2| \leq 1$ for any $\kappa_3, \kappa_4, \xi, \eta$. That is there are no limitations for κ_3 and κ_4. This is not surprising because for $\kappa_1 = \kappa_2 = 0$ scheme (5.16) is implicit, therefore, it is absolutely stable [26].

We now consider the particular case when $\kappa_3 = \kappa_4 = 0$, $\kappa_1 \geq 0$, $\kappa_2 \geq 0$. It is clear that the coefficient c in (5.29) reaches its maximum over ξ, η at $\xi = \eta = \pi/2$. If $\kappa_1 < 0$, $\kappa_2 < 0$,

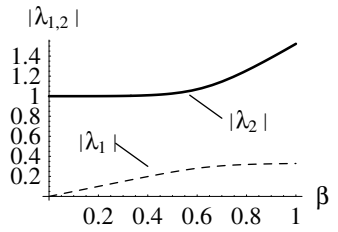


Figure 5.2: The graphs of $|\lambda_{1,2}|$ vs. β

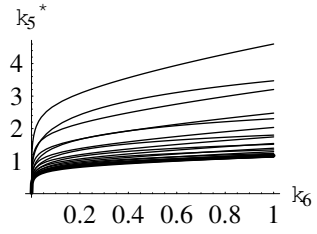


Figure 5.3: The graphs of the root $(\kappa_5^*)_2$ vs. κ_6 for different ξ

then this maximum is reached at $\sin\xi = \operatorname{sgn}\kappa_1$, $\sin\eta = \operatorname{sgn}\kappa_2$. Then in the general case it is obvious that $\max_{\xi,\eta}|c| = (1/2)(|\kappa_1| + |\kappa_2|)$. The graphs of the quantities $|\lambda_1|$, $|\lambda_2|$ are shown in Fig. 5.2 as the functions of the quantity $\beta = |\kappa_1| + |\kappa_2|$. It is seen that $|\lambda_2|$ exceeds unity by a small value in the interval $0 \le \beta < 0.5$. That is scheme (5.16) is weakly unstable in this interval.

It follows from the above consideration of particular cases that the necessary stability condition of scheme (5.16) for values $\kappa_1, \kappa_2, \kappa_3, \kappa_4$ different from zero must have the following form: $|\kappa_1| + |\kappa_2| \le \varphi(\kappa_3, \kappa_4)$, where the function $\varphi(\kappa_3, \kappa_4)$ should satisfy the following properties:

- $\varphi(0,0) = 0$;
- $\varphi(\kappa_3, \kappa_4) > 0$, $|\kappa_3| + |\kappa_4| > 0$.

The property $\varphi(0,0) = 0$ ensures the presence of the above revealed instability of scheme (5.16) for $\kappa_3 = \kappa_4 = 0$.

In the case when $\kappa_1 \ne 0, \kappa_2 \ne 0, \kappa_3 \ne 0, \kappa_4 \ne 0$ the derivation of stability condition in an analytic form from (5.35) is difficult because of the availability of square roots of complex numbers. In this connection, we use in the following the concept of the resultant, to which one can reduce the problem of determining the stability region boundary. The corresponding procedure was described in [25], therefore, we present it only briefly here. Thus, let $f(\lambda, \vec{\kappa}, \xi, \eta)$ be the characteristic polynomial of a difference scheme, and let its degree in λ be equal to m ($m \ge 1$). Following [25] let us perform the Möbius transformation $\lambda = (\omega + 1)/(\omega - 1)$. Then we obtain the polynomial

$$g(\omega, \vec{\kappa}, \xi, \eta) = (\omega - 1)^m f((\omega + 1)/(\omega - 1), \vec{\kappa}, \xi, \eta).$$

Let $\omega_1, \ldots, \omega_m$ be the roots of polynomial g. The condition $\operatorname{Re}\omega_j \le 0$, $j = 1, \ldots, m$, corresponds to condition $|\lambda_j| \le 1$, $j = 1, \ldots, m$. Then at the boundary Γ of the stability region the polynomial g must have at least one purely imaginary zero. Set $\omega = i\sigma$ and consider the polynomial $\psi(\sigma, \vec{\kappa}, \xi, \eta) = g(i\sigma, \vec{\kappa}, \xi, \eta)$. It is clear that the boundary Γ is determined by those values of quantities $\vec{\kappa}$, ξ, η, at which the polynomial ψ has a real zero σ. Zeroes of polynomial ψ are determined by the system of two equations with real

coefficients $\mathrm{Re}\,\psi = 0$, $\mathrm{Im}\,\psi = 0$. This system has the solution if and only if the resultant of equations $\mathrm{Re}\,\psi = 0$, $\mathrm{Im}\,\psi = 0$ equals zero:

$$\mathrm{Res}(\mathrm{Re}\,\psi, \mathrm{Im}\,\psi) = 0. \tag{5.36}$$

As a result, we obtain the following formula for $\mathrm{Res}(\mathrm{Re}\,\psi, \mathrm{Im}\,\psi)$:

$$\begin{aligned} R(\vec{\kappa}, \xi, \eta) &= \mathrm{Res}(\mathrm{Re}\,\psi, \mathrm{Im}\,\psi) = -a^4 + a^2 b_1^2 + a^2 b_2^2 + 4ab_1 b_2 c_1 + 2a^2 c_1^2 \\ &+ b_1^2 c_1^2 + b_2^2 c_1^2 - c_1^4, \end{aligned} \tag{5.37}$$

where in accordance with (5.29)

$$\begin{aligned} a &= 1 + \kappa_3(1 - \cos\xi) + \kappa_4(1 - \cos\eta),\ b_1 = 1 - \kappa_3(1 - \cos\xi) - \kappa_4(1 - \cos\eta), \\ b_2 &= -(3/2)(\kappa_1 \sin\xi + \kappa_2 \sin\eta),\ c_1 = (1/2)(\kappa_1 \sin\xi + \kappa_2 \sin\eta). \end{aligned} \tag{5.38}$$

The substitution of expressions (5.38) in (5.37) leads to a bulky formula, which we do not present here for the sake of brevity.

As we have shown above in this section, in the particular case when $\kappa_3 = \kappa_4 = 0$ the most restrictive stability condition is obtained for $\sin\xi = \sin\eta = 1$. In this connection, we will investigate in the following the case $\xi = \eta$ in more detail. As a result, we obtain a quadratic equation in $z = \kappa_1^2$ to determine the roots of equation $R(\vec{\kappa}, \xi, \xi) = 0$. Using the Maple command `solve` we can obtain the analytic expressions for the both roots. For the sake of brevity we present only the second root z_2. We introduce the notation $\kappa_5 = |\kappa_1| + |\kappa_2|$, $\kappa_6 = \kappa_3 + \kappa_4$, $z = \kappa_5^2$. Denote by κ_5^* the value of quantity κ_5 at the stability region boundary. Then

$$\begin{aligned} z_2 &= (\kappa_5^{*2})_2 = \frac{1}{2}\mathrm{Csc}^4\xi \left(-10\kappa_6 \sin^2\xi - 12\kappa_6^2 \sin^2\xi + 10\kappa_6 \cos\xi \sin^2\xi \right. \\ &+ 24\kappa_6^2 \cos\xi \sin^2\xi - 12\kappa_6^2 \cos^2\xi \sin^2\xi + 2\sqrt{\kappa_6}\sqrt{-1 + \cos\xi} \times \\ &\left. (-1 - 2\kappa_6 + 2\kappa_6 \cos\xi)\sqrt{-8 - 9\kappa_6 + 9\kappa_6 \cos\xi} \sin^2\xi \right). \end{aligned} \tag{5.39}$$

In particular, at $\xi = \eta = \pi/2$ we obtain the following expressions for the both roots $(\kappa_5^*)_1$ and $(\kappa_5^*)_2$:

$$\begin{aligned} (\kappa_5^*)_1 &= (-5\kappa_6 - 6\kappa_6^2 - \sqrt{\kappa_6}(1 + 2\kappa_6)\sqrt{8 + 9\kappa_6})^{1/2}, \\ (\kappa_5^*)_2 &= (-5\kappa_6 - 6\kappa_6^2 + \sqrt{\kappa_6}(1 + 2\kappa_6)\sqrt{8 + 9\kappa_6})^{1/2}. \end{aligned}$$

The radicand in formula for $(\kappa_5^*)_1$ is negative because it is the sum of negative items. Therefore, it is worthwhile considering only the root z_2 given by (5.39). In order to be sure that the values $\xi = \eta = \pi/2$ yield the most restrictive stability condition we have constructed twenty curves of the family $(\kappa_5^*)_2(\xi, \xi)$ with the step $\Delta\xi = 0.045\pi$. These curves are shown in Fig. 5.3, in which the curve for the particular pair $\xi = \eta = \pi/2$ is shown as a thick line. We can see that this line is the lowest one in Fig. 5.3. Thus, we have obtained an approximate form of the necessary stability condition:

$$|\kappa_1| + |\kappa_2| \le (-5\kappa_6 - 6\kappa_6^2 + \sqrt{\kappa_6}(1 + 2\kappa_6)\sqrt{8 + 9\kappa_6})^{1/2}. \tag{5.40}$$

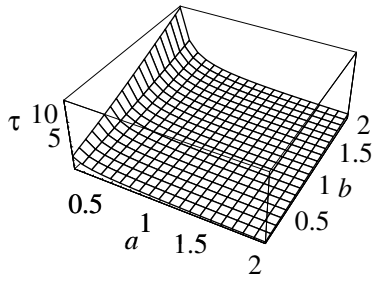


Figure 5.4: The surface $\tau = \tau(a, b)$

For $\xi = \eta = \pi/2$, the expression for the resultant becomes especially simple:

$$R(\vec{\kappa}, \pi/2, \pi/2) = (1/2)z^2 - 4\kappa_6 + 5z\kappa_6 - 8\kappa_6^2 + 6z\kappa_6^2 - 4\kappa_6^3. \tag{5.41}$$

Substituting the expressions for $\kappa_1, \kappa_2, \kappa_3, \kappa_4$ from (5.30) into (5.41) we obtain a fourth-degree polynomial equation for determining the time step τ. Its solution is efficiently found with the aid of the Mathematica function `Solve[...]`, and it turns out that equation $R = 0$ has two real roots and two complex conjugate roots. The real root $\tau = 0$ is of no practical value. The other real root is as follows:

$$\begin{aligned}
\tau &= -\frac{2(5ab - 4b^3)}{3(a^2 + 12ab^2)} - \left(2^{1/3}(-148a^2b^2 - 416ab^4 - 64b^6)\right) / (3(a^2 + 12ab^2) \times \\
&\quad (216a^4b + 1744a^3b^3 + 19776a^2b^5 + 9984ab^7 + 1024b^9 \\
&+ 24\sqrt{3}a^{3/2}b\sqrt{27a + 4b^2}(a^2 + 8ab^2 - 48b^4))^{1/3}) \\
&+ \frac{1}{3 \cdot 2^{1/3}(a^2 + 12ab^2)} \left((216a^4b + 1744a^3b^3 + 19776a^2b^5 + 9984ab^7 \right. \\
&+ \left. 1024b^9 + 24\sqrt{3}a^{3/2}b\sqrt{27a + 4b^2}(a^2 + 8ab^2 - 48b^4)\right)^{1/3},
\end{aligned} \tag{5.42}$$

where

$$a = \left(\frac{|U|}{h_1} + \frac{|V|}{h_2}\right)^2, \quad b = \frac{\nu}{h_1^2} + \frac{\nu}{h_2^2}. \tag{5.43}$$

Note that after the non-dimensionalization of the Navier-Stokes equations, the value ν is usually replaced with $\nu = 1/\mathrm{Re}$.

We show in Fig. 5.4 the surface $\tau = \tau(a, b)$. We can draw the following conclusions from this figure: (i) for sufficiently large values of $|U|$ and $|V|$, such that $a > 0.5$, the time steps are smaller than for $a < 0.5$; (ii) for low Reynolds numbers, when b is sufficiently high, the maximum time step becomes higher and higher with increasing b for sufficiently low a. This may be explained by the well-known fact that with decreasing Re, the dissipative effects become more pronounced, and right these effects are known to stabilize the difference solution.

Let us consider the case when $0 < \kappa_6 \ll 1$ (high Reynolds numbers). Using the Mathematica command `Series[`τ`,b,0,1}]` we find:

$$\tau = \frac{2(a^4)^{1/3}b^{1/3}}{a^2} - \frac{10b}{3a} + O(b^{4/3}). \tag{5.44}$$

If, for example, $\mathrm{Re} = 1/\nu = 10^4$, then $\tau = O(10^{-4/3})$. This consideration explains why the computations by scheme (5.16) are stable also for such high Reynolds numbers.

Note that formula (5.42) for the maximum time step allowed by stability is approximate because for $\xi \neq \eta$ one may expect, in principle, a somewhat more restrictive stability condition. Therefore, it is advisable to compute the time step τ_n in computer code implementing scheme (5.16) from the known difference solution at the nth time level by formula

$$\tau_n = \theta \cdot \min_{j,k} \tau(a_{j,k}, b), \tag{5.45}$$

where τ is computed by (5.42) at each grid cell (j, k), and θ is the user-specified safety factor, $0 < \theta \leq 1$ (for example, $\theta = 0.98$).

On the other hand, although the stability condition (5.42) is approximate, it has a correct analytic form obtained from the von Neumann stability condition with the aid of the algebra of resultants. This enables the obtaining of information on the stability properties of a numerical method under the variation of such important physical parameters as the Reynolds number and the gas velocity.

A shortcoming of symbolic-numerical methods for stability investigation consists of the fact that although it is possible to obtain with their aid a finite set of the stability region boundary points these methods do not give information about the structure of the analytic form of the stability region boundary. Although one can obtain the analytic approximation for the maximum time step τ with the aid of the method of least squares the resulting analytic formulas have a shortcoming that they specify this analytic form in a user-predefined class of forms, which may be far from the true analytic dependence.

5.1.3 Verification of Stability Conditions

The Taylor–Green Vortex

The Taylor–Green vortex is one of few analytical solutions of the two-dimensional Navier–Stokes equations. The solution, with $\nu = 1$ and $\rho = 1$, is given by formulas [37]

$$u = -e^{-2t} \cos x \sin y, \;\; v = e^{-2t} \sin x \cos y, \;\; p = -e^{-4t}(\cos 2x + \cos 2y)/4. \tag{5.46}$$

The flow is represented by periodic counter-rotating vortices that decay in time. The computational domain is over $\pi/2 \leq x, y \leq 5\pi/2$, which corresponds to homogeneous Dirichlet boundary conditions for the velocity component normal to the boundary and homogeneous Neumann boundary condition for the velocity component tangent to the boundary. The pressure boundary condition is homogeneous Neumann everywhere.

We have carried several computations of this test problem using formula (5.42) for the time step τ. It turns out that this formula gives the τ values, which are by factors from 2 to 6 higher than those obtained from formula (5.21). These factors varied depending on the grid step sizes h_1 and h_2 and the number of executed time steps, that is on the local values of the velocity components. The computation by the above described difference method nevertheless remained stable when using (5.42) with safety factor $\theta = 0.5$.

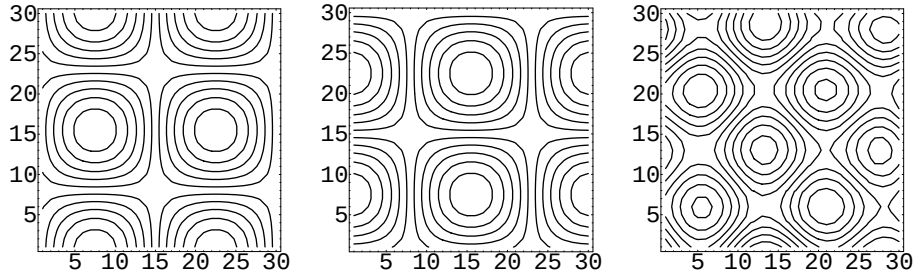


Figure 5.5: The contours of u (a), v (b), and p (c) for $n = 20$ ($t = 0.96157$)

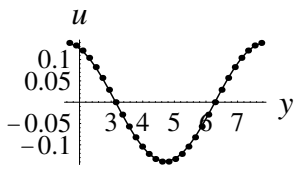


Figure 5.6: The profile of $u = u(x_0, y)$, $x_0 = 3.45575$ ($\approx 1.1\pi$); solid line is the exact solution, dotted line is the numerical solution for $n = 20$ ($t = 0.96157$),

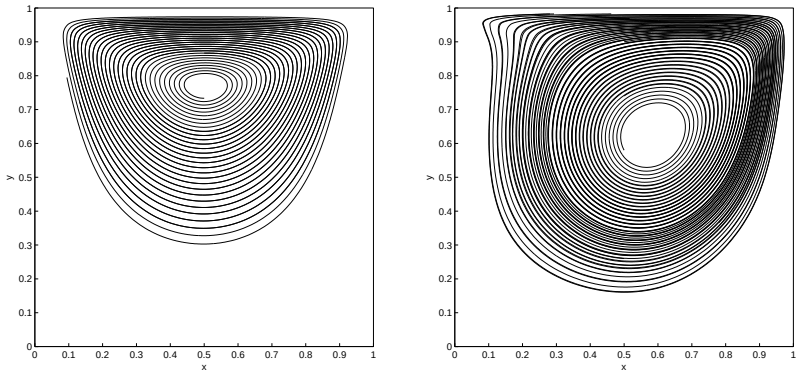
Since the amplitudes of the velocity components decay exponentially with time in the given task, we present in Figures 5.5 and 5.6 the results for the case of using formula (5.21) with $C_{conv} = 0.5$, $C_{diff} = 0$ and the 30×30 grid to show that our computer code works correctly also after executing several dozens of time steps.

Lid-Driven Cavity Problem

This problem is frequently used as a test of numerical methods for the incompressible Navier–Stokes equations, although it has no known exact analytic solution. In this problem the no-slip boundary conditions are imposed on the left, bottom, and right walls of the cavity, and the x-component U_0 of the velocity is specified at the upper boundary (the moving "lid"). Let B be the horizontal cavity size. Then the dimensional lengths are non-dimensionalized with respect to B, and the Reynolds number Re has the form $\mathrm{Re} = U_0 B\rho/\mu$. The dimensionless velocity component $u = 1$ at the lid. The pressure boundary condition is homogeneous Neumann everywhere.

We have done numerous computations by the difference method of Section 2 for the purpose of elucidating the validity of formula (5.42) for the maximum time step allowed by stability. We at first consider the case when the Reynolds number $\mathrm{Re} = 1$. It turns out that the computation remains stable even if the actual time step exceeds the value given by (5.42) by a factor of three, that is $\theta = 3$ in (5.45). But, on the other hand, for $\theta > 1$ the convergence to the stationary solution of the lid-driven cavity problem slows down with increasing θ.

Another interesting fact revealed by our computations in the low Reynolds number case is that the actual time step computed with the aid of (5.45) was by factors from 33 to 58 higher than in the case of using the known empirical formula (5.21), in which we specified the values $C_{conv} = 0.5$, $C_{diff} = 0$. This result means that in the case of numerical solution



(a) (b)

Figure 5.7: Streamlines in the lid-driven cavity problem: (a) Re = 1; (b) Re = 400

of more complex stationary flow problems with low Reynolds numbers it is possible to have very significant savings in CPU times (by a factor of up to 58).

And the final observation, which we have drawn from our numerical experiments involving (5.42) is that it ensures the fastest convergence to the stationary solution in the case of Re = 1 when the value on the right-hand side of (5.42) is multiplied by a safety factor of about 0.6.

In the case of a higher Reynolds number, namely Re = 400, the computation using (5.45) with $\theta = 1$ proves to be unstable. In order to ensure the stability for Re = 400, one must take the value $\theta < 0.1$ in (5.45). But even in this case, the actual "stable" time step exceeded the value given by (5.21) by a factor of about five.

Although the computation using (5.45) may remain stable also for $\theta > 1$, in the case of large time steps one should ensure the needed accuracy of the results. For this purpose, one can use the known test problems for which the exact analytic solutions are available.

We show in Fig. 5.7 some numerical results obtained with the use of formula (5.42), in which the right-hand side was multiplied by the safety factor $\theta = 0.6$ in the case of Re = 1. One can verify that these two figures are very similar to Figs. 4, (a) and (b) from [37]. Figure 5.7 was obtained on a mesh of 30×30 cells.

Bibliography

[1] J. Angeles: Fundamentals of Robotic Mechanical Systems, Springer Verlag New-York, 2003

[2] S.W. Armfield, R. Street: The fractional-step method for the Navier–Stokes equations on staggered grids: the accuracy of three variations, J. Comp. Phys., 153:660–665, 1999

[3] S. Basu, R. Pollack, M.-F. Roy : Computing Roadmaps of Semi-algebraic Sets on a Variety. In Foundations of Computational Mathematics, F. Cucker and M. Shub (eds.), 1-15, Springer-Verlag, 1997

[4] A. Ben-Israel, T. N. E. Greville: Generalized Inverses: Theory and Applications, John Wiley and Sons, 1974

[5] A. Bjerhamar: A Generalized Matrix Algebra, Kungl. Tekn. Hoegsk. Handl, 49, 1951

[6] B. J. Boersma, G. Brethower, F. T. M. Nieuwstadt: Numerical investigtion on the effect of the inflow conditions on the self-similar region of a round jet, Phys. Fluids, 10:899–909, 1998

[7] L. C. Groove, C. T. Benson: Finite Reflection Groups, Springer-Verlag, 1985

[8] J. W. Bruce, P. J. Giblin: Curves and Singularities, Cambridge University Press, 1984

[9] H.-J. Bungartz: Dünne Gitter und deren Anwendung bei der adaptiven Lösung der dreidimensionalen Poisson-Gleichung, Dissertation, Institut für Informatik, Technische Universitait München, 1992

[10] J. Canny: The Complexity of Robot Motion Planning, MIT Press, 1987

[11] J. Canny, A. Rege, and J. Reif: An exact algorithm for kinodynamic planning in the plane, Discrete and Computational Geometry, 6:461-484, 1991

[12] B. F. Caviness , J. R. Johnson (eds.): Quantifier Elimination and Cylindrical Algebraic Decomposition, Springer-Verlag, 1998

[13] Y.-C. Chen: Solving Robot Trajectory Planning Problems with Uniform Cubic B-Splines, Opt. Contr. Appl. and Meth., 12:247–262, 1991

[14] D. Chibisov, V. Ganzha, E. W. Mayr, E. V. Vorozhtsov: Generation of orthogonal grids on curvilinear trimmed regions in constant time. In Proc. CASC'2005, V. G. Ganzha, E.W. Mayr, E. V. Vorozhtsov (eds.), LNCS 3718, Springer-Verlag, Berlin, Heidelberg, 2005, 105–114

[15] D. Chibisov , V. Ganzha, E. W. Mayr, E. V. Vorozhtsov: Stability Investigation of a Difference Scheme for Incompressible Navier-Stockes Equations in Proc. CASC'2007, V. G. Ganzha, E.W. Mayr, E. V. Vorozhtsov (eds.), LNCS 4770, Springer-Verlag, Berlin, Heidelberg, 2005, 102–117

[16] D. Chibisov , V. G. Ganzha, E.V. Vorozhtsov: Hierarchical Advancing Front Triangulation Using Symmetry Properties, In Proc. CASC'2004, V. G. Ganzha, E.W. Mayr, E. V. Vorozhtsov (eds.), TUM Press, 2004

[17] D. Chibisov and E. W. Mayr: Computing Minimum-Time Motion for 6R Robots with Application to Industrial Welding, In L. Gadomski, M. Jakubiak, and A. N. Prokopenya (eds.), CASTR'07, Computer Algebra Systems in Teaching and Research, 36–46, Wydavnictwo Akademii Podlaskiej Siedlce, Poland, 2007

[18] D. Chibisov, E. W. Mayr: Motion Planning for 6R Robots: Multiple Tasks with Constrained Velocity and Orientation of the End-Effector, in Proc. of the 2007 International Workshop on Symbolic-Numeric Computation, July 27-29, 2007, London, Ontario, Canada, ACM Press, New York

[19] D. Chibisov, E.W. Mayr, S. Pankratov: Spatial Planning and Geometric Optimization: Combining Configuration Space and Energy Methods . In Proc. ADG'2004, H. Hong, D. Wang (eds.), LNAI 3763, Springer-Verlag, 2006

[20] G. E. Collins: Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition, Lect. Notes in Comp. Sci. 33, 515-532, Springer-Verlag, 1975

[21] D.A. Cox, J.B. Little, D. O'Shea: Ideals, Varieties, and Algorithms, Springer-Verlag, Berlin, 1996

[22] Croft T.H., Falconer K. J., Guy R. K.: Unsolved Problems in Geometry, Springer-Verlag, 1991

[23] J. C. Culberson: Sokoban is PSPACE-complete. In Proceedings International Conference on Fun with Algorithms (FUN98), 6576, Waterloo, Ontario, Canada, June 1998, Carleton Scientific

[24] J. Denavit and R. S. Hartenberg: A Kinematic Notation for Lower-Pair Mechanisms Based Upon Matrices. J. App. Mechanics, 77:215–221, 1955

[25] V. G. Ganzha, E.V. Vorozhtsov.: Computer-Aided Analysis of Difference Schemes for Partial Differential Equations, Wiley-Interscience, New York, 1996

[26] V. Ganzha, E. V. Vorozhtsov: Numerical Solution for Partial Differential Equations: Problem Solving Using Mathematica, CRC Press, Boca Raton, Ann Arbor, 1996

[27] V. Ganzha, E.V. Vorozhtsov: Symbolic-numerical computation of the stability regions for Jameson's schemes. Mathematics and Computers in Simulation, 42:607-615, 1996

[28] P.L. George: Automatic Mesh Generation. Application to Finite Element Method, John Wiley & Sons, New York, 1991.

[29] F.H. Harlow, J.E. Welch: Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. Phys. Fluids, 8:2182–2189, 1965

[30] Chr. Hoffmann: Implicit curves and surfaces in CAGD, IEEE Computer Graphics and Applications, 1993

[31] W. Hoffmann and T. Sauer: A Spline Optimization Problem from Robotics, Rediconti di Mathematica, 26:221–230, 2006

[32] J. Hopcroft, D. Joseph, and S. Whitesides: Movement problems for 2- dimensional linkages. In J .T .Schwartz, M. Sharir, and J. Hopcroft (eds.), Planning, Geometry, and Complexity of Robot Motion, 282329, Ablex, Norwood, NJ, 1987

[33] J. E. Hopcroft, J. T. Schwartz, and M. Sharir: On the complexity of motion planning for multiple independent objects: PSPACE-hardness of the warehousemans problem. International Journal of Robotics Research, 3(4):7688, 1984

[34] D. A. Joseph and W. H. Plantiga: On the complexity of reachability and motion planning questions. In Proceedings ACM Symposium on Computational Geometry, 6266, 1985

[35] B.B. Ilyushin, D. V. Krasinsky: Large eddy simulation of the turbulent round jet dynamics. Thermophysics and Aeromechanics, 13(1):43–54, 2006

[36] O. Khatib: Real time obstacle avoidance for manipulators and mobile robots, Internat. J. Robotics, 5:90-99, 1986

[37] J. Kim, P. Moin: Application of a fractional-step method to incompressible Navier–Stokes equations. J. Comp. Phys., 59:308–323, 1985

[38] M. P. Kirkpatrick, S.W. Armfield, J.H. Kent: A representtion of curved boundaries for the solution of the Navier–Stokes equations on a staggered three-dimensional Cartesian grid. J. Comp. Phys., 184:1–36, 2003

[39] P. Knupp, , S. Steinberg: Fundamentals of Grid Generation, CRC Press, Boca Raton, Ann Arbor, 1986

[40] KUKA Roboter GmbH: Specification of KR 60 HA, http://www.kuka.com/germany/en/products/industrial_robots/medium/kr60_ha/

[41] J.-C. Latombe: Robot Motion Planning, Kluwer Academic Publishers, 1991

[42] C.S. Lin, P.-R. Chang, J.Y.S. Luh: Formulation and Optimization of Cubic Polynomial Joint Trajectories for Industrial Robots, IEEE Trans. on Automatic Control, 28:1066–1074, 1983

[43] T. Lozano-Perez: Spatial Planning: A Configuration Space Approach, IEEE Transactions on Computers, C-32 (2), 108-120, 1983

[44] S. M. LaValle: Planning Algorithms, Cambridge University Press, 2006

[45] A. Y. Lee: Solving Constrained Minimum-Time Robot Problems using the Sequential Gradient Restoration Algorithm, Opt. Contr. Appl. and Meth., 13:145–154, 1992

[46] S. Marella, S. Krishnan, H. Liu, H.S. Udaykumar: Sharp interface Cartesian grid method I: An easily implemented technique for 3D moving boundary computations. J. Comp. Phys., 210:1–31, 2005

[47] M.L. Minion: On the stability of Godunov-projection methods for incompressible flow. J. Comp. Phys.,123:435–449, 1996

[48] C. O'Dunlaing: Motion planning with inertial constraints, Algorithmica, 2(4):431-475, 1987

[49] A. Pasko, O. Okunev, V. Savchenko: Minkowski sum of point sets defined by inequalities, Computers and Mathematics with Applications, 45(10/11):1479-1487, 2003

[50] D. W. Peaceman, H.H. Rachford: The numerical solution of parabolic and elliptic differential equations, J. of SIAM, 3:28-41, 1955

[51] R. Penrose: A generalized Inverse for Matricesm, In Proc. Cambridge Philos. Soc., 51:406–413, 1955.

[52] J. H. Reif : Complexity of the Generalized Mover's Problem; In Planing, Geometry and Complexity of Robot Motion, T. Schwartz, M. Sharir, J. Hopcroft (eds.), 267-281, Ablex Publishing Corporation, 1987

[53] A. Requicha: Representations for Rigid Solids: Theory, Methods, and Systems, ACM Computing Surveys (CSUR) Archive, 12(4):437-464, ACM Press, 1980

[54] E. Rimon, D. E. Koditschek: The Construction of Analytic Diffeomorphisms for Exact Robot Navigation on Star Worlds, Transactions of the AMS, 327(1):71-116, 1991

[55] P. J. Roache: Computational Fluid Dynamics. Hermosa, Albuquerque, New Mexico, 1976

[56] V. L. Rvachov: Methods of Logic Algebra in Mathematical Physics, Naukova Dumka, Kiev, 1974 (in Russian)

[57] H. Samet: The Design and Analysis of Spatial Data Structures, Addison-Wesley, 1990

[58] T.W. Sederberg: Implicit and Parametric Curves and Surfaces for Computera-Aided Geometric Desgin, PhD Thesis, Purdue University,1983

[59] M. Schlemmer and O. von Stryk, Optimal Control of the Industrial Robot Manutec r3, In D. Kraft R. Bulirsch (eds.), Computational Optimal Control, volume 115 of International Series of Numerical Mathematics, 367–382, 1994.

[60] J. Schwartz, M. Sharir: On the Piano Movers Problem II. General Techniques to Computing Topological Properties of Real Algebraic Manifolds, Advances in Applied Mathematics, 4:298-351, 1983

[61] J. Schwartz, C.K. Yap: Advances in Robotics , Lawrence Erlbaum associates, Hillside New Jersey, 1986

[62] V. Shapiro: Theory and Applications of R-Functions: A primer, Technical Report, Cornel University, 1991

[63] V. Shapiro, I. Tsukanov: Implicit Functions With Guaranteed Differential Properties; In Proceedings of the Fifth Symposium On Solid Modeling "SOLID MODELING'99", 258-269, Ann Abor, Michigan, ACM Press, 1999

[64] H. Schlichting, E. Truckenbrodt: Aerodynamics of the Airplane, McGraw-Hill, New York, 1979

[65] M. Uhlmann: An immersed boundary method with direct forcing for the simulation of particulate flows. J. Comp. Phys., 209:448–476, 2005

[66] H. A. van der Vorst: Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, SIAM J. Sci. Statist. Comput., 13:631–644, 1992

[67] V. Weispfenning: Quantifier elimination for real algebra - the cubic case. In Proc. ISAAC 1994, Oxford, England UK

[68] J. Yang, E. Balaras: An embedded-boundary formulation for large-eddy simulation of turbulent flows interacting with moving boundaries, J. Comp. Phys., 215:12–40, 2006

[69] C. Zenger: Sparse grids. In: Parallel Algorithms for Partial Differential Equations, Proc. Sixth GAMM-Seminar, Kiel, 1990, Hackbusch, W., editor, Vol. 31 of Notes on Num. Fluid Mech. Vieweg-Verlag, Braunschweig/ Wiesbaden, 241–251, 1990