Grups d'Estudi de Matemàtica i Tecnologia

Barcelona, July 2009

Edited by

Aureli Alabert (UAB)

Jordi Saludes (UPC)

Joan Solà-Morales (UPC)

Centre de Recerca Matemàtica
Campus de Bellaterra, Edifici C
08193 Bellaterra (Barcelona)

First edition: June 2010

ISBN:
Legal deposit:

Presentation

This booklet collects the problems that were proposed at the *Grups d'Estudi de Matemàtica i Tecnologia* (Study Groups of Mathematics and Technology, GEMT 2009), and the reports on the solutions to these problems. The problems belong to different areas of Mathematics and were proposed by companies and institutions in various ways. The event took place in the premises of the Facultat de Matemàtiques i Estadística of the Universitat Politècnica de Catalunya, Barcelona, from July 7 to 9, 2009.

The participation was free of any cost for companies and institutions and also for participants. We are especially grateful to the two institutional co-organizers, the Facultat de Matemàtiques i Estadística of the Universitat Politècnica de Catalunya (FME) and the Centre de Recerca Matemàtica (CRM). They both contributed by strongly supporting these Study Groups. We also thank Ingenio Mathematica (i-MATH) for its financial support. Additionally, we have worked in coordination with the *Jornadas de Consulta Matemática para Empresas e Instituciones*, organized by CESGA in Santiago de Compostela, an event with the same aims as ours.

Finally, we thank very much all the participants: the companies and institutions that presented the problems and also the researchers who contributed to the discussions and wrote the final reports.

Barcelona, December 2009

Aureli Alabert, Jordi Saludes, and Joan Solà-Morales

Statement of the Problems

Continuous symmetry and shape measures, or how to measure the distance between polyhedra representing molecules

by Pere Alemany, Institut de Química Teòrica i Computacional de la Universitat de Barcelona (IQTCUB), Barcelona, Spain

1.1 Introduction

One of the most deeply rooted ideas in chemistry is that molecules which are similar should behave in a similar way, that is, they should exhibit similar chemical and physical properties. This kind of reasoning is at the origin of a large number of attempts to rationalize chemical observations, including the type theory developed in the first half of the 19th century, the concept of functional groups in organic chemistry, or modern Quantitative Structure Activity Relations (QSAR) methods.

Although it is difficult to define precisely what is understood by similarity in chemistry, it is easy to distinguish two main aspects: chemical similarity arising from the periodic trends in the electronic structure of atoms (alcohols and thiols are similar in this sense, since oxygen and sulphur have similar chemical properties) and structural similarity, referring to molecules in which atoms (which may be chemically similar or not) occupy similar positions in space (methane and the phosphate ion would be similar in this sense, since both have tetrahedral structures). Intuitively, it is easy to conclude that symmetry must play a crucial role in structural similarity, although it is not a trivial one, since symmetry and similarity are usually measured using different approaches: similarity is treated as a continuous (grey) property, allowing the definition of a degree of similarity, while symmetry is described as a discrete (black or white) property which is either present in a molecular

structure or not. This gap has, however, been closed in recent years by the proposal of Avnir et al. to also define symmetry as a continuous property by introducing the formalism of *continuous symmetry measures* (CSM) and the closely related *continuous shape measures* (CShM)[1].

Let us start with the conceptually simpler problem that leads to the definition of continuous shape measures. As an example, we may consider the phosphate ion (PO_4^{3-}) present in a large number of inorganic and organic compounds. In classical structural chemistry, the phosphate ion is said to have a tetrahedral structure, in the sense that the four oxygen atoms are located around the central phosphorus atom occupying the positions of the vertices of a tetrahedron (Figure 1.1).

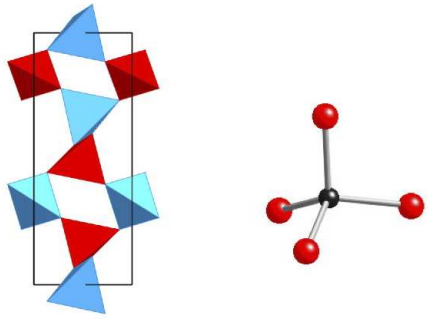


Figure 1.1: **Left:** Crystal structure of Berlinite ($AlPO_4$) showing PO_4 tetrahedra in red and AlO_4 tetrahedra in blue. **Right:** Ball and stick representation of one PO_4 tetrahedron with the central phosphorus atom in black and the four oxygen atoms in red.

In spite of the widespread idea that the phosphate ion is tetrahedral, a detailed analysis of 211 good quality crystal structures of compounds containing PO_4^{3-} ions revealed that none of the phosphorus atoms in these crystals was really located at a position compatible with full tetrahedral symmetry, with all the phosphate ions presenting more or less pronounced distortions from the shape of an ideal tetrahedron.

Careful analysis reveals that situations like this are not an exception in structural chemistry, but that they are the rule. Several tools have been developed in order to quantify the distortions from ideal polyhedral structures, relying on the analysis of geometrical parameters such as bond distances and

[1] H. Zabrodsky, S. Peleg, D. Avnir, *J. Am. Chem. Soc.* **1992**, 114, 7843–7851. This article actually introduces continuous symmetry measures, not continuous shape measures, although in the particular case of the tetrahedron these two measures can be shown to be equivalent.

angles, or on group theoretical techniques applied to a normal mode analysis for a given structure. All these approaches, although useful for the purposes for which they were developed, share a common problem, since they are not easily generalizable and their application is limited in many cases to small distortions for a specific polyhedron.

A general solution to this problem was given by Avnir and his colleagues in 1992 by introducing continuous shape measures. The basic idea behind CShM is to define a distance between the real and the ideal polyhedra and to use this distance to gauge the similarity/dissimilarity between the structures[2].

The mathematical expression of this statement for an arbitrary structure Q defined as a set of N vertices with coordinates $\{q_k\}$ with respect to a reference structure P defined as another set of N vertices with coordinates $\{p_k\}$ is

$$S(Q,P) = 100 \cdot \min \frac{\sum_{k=1}^{N} |q_k - p_k|^2}{\sum_{k=1}^{N} |q_k - q_0|^2} \tag{1.1}$$

where q_0 is the geometric centre of Q. Since the symmetry of an object does not depend on its size, position or orientation, the measure must be minimized for all possible relative positions and orientations between the structures, as well as with respect to their relative size[3]. In other words, if we consider our problem structure Q with vertices $Q = q_k$ and an ideal structure P with vertices located initially at $P^0 = p_k^0$, the coordinates for the ideal structure $P = p_k$ minimizing equation (1.1) can be expressed as:

$$P = A\mathbf{R}P^0 + t \tag{1.2}$$

where A is a scaling factor, $\mathbf{R}$ a matrix associated to a unitary transformation (the 3×3 rotation that determines the optimal spatial orientation of P), and t a translation vector. The problem of minimizing $S(Q,P)$ to find A, $\mathbf{R}$ and t has been solved analytically and an efficient algorithm is thus available. To ensure that $S(Q,P)$ is really a measure of the P-shape contents of Q, one must perform an additional minimization over all possible pairings between the vertices of Q and P. Finally, the values of $S(Q,P)$ are scaled by multiplying them by an arbitrary factor of 100. Adopting this definition for the CShM, an object having the desired P-shape will have $S(Q,P) = 0$, while distortions of this object from the ideal symmetry will lead to higher values of the measure (Figure 1.2).

[2]M. Pinsky, D. Avnir, *Inorg. Chem.* **1998**, 37, 5575–5582.

[3]For a detailed description of the mathematical background of continuous shape measures, see: David Casanova, *Mesures de forma i simetria: algorismes i aplicacions*, Doctoral Thesis, Universitat de Barcelona, **2006**.

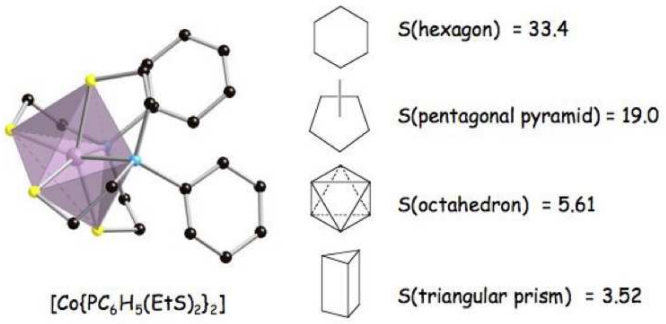


Figure 1.2: Continuous shape measures for the coordination polyhedron around the cobalt atom in $[Co\{PC_6H_5(EtS)_2\}_2]$ showing that the best description for its geometry is that of a triangular prism.

Since minimization with respect to position, orientation, and scaling can be performed analytically, it is easy to see that the computational bottleneck in practical applications is associated to the minimization over all $N!$ possible pairings between the vertices of Q and those of P.

The definition of continuous shape measures is easily extended to the so-called continuous symmetry measures. The question in this case is somewhat different. While in CShM we compare the shape of a problem structure Q defined by N vertices with that of a reference structure P with the same number of vertices, in CSM we search for the closest structure T with N vertices that belongs to a given point group G:

$$S(Q,G) = 100 \cdot \min \frac{\sum_{k=1}^{N} |q_k - t_k|^2}{\sum_{k=1}^{N} |q_k - q_0|^2}. \tag{1.3}$$

Although, for some special structures such as the tetrahedron, CShM and CSM are coincident since the tetrahedron is the only 4-vertex polyhedron with T_d symmetry, in general CShMs are more restrictive than CSMs. If we want to evaluate, for example, the $S(Q, D_{4h})$ value for a given 8-vertex structure Q, we need not only to compare Q with a given square prism P, but we need to find the square prism T that is closest (regardless of position, orientation and size) to the problem structure Q. In this example, this implies an additional minimization of the length/height ratio for the square prisms. However, in the general case, looking for T is not a trivial task. To solve this problem, an efficient numerical algorithm, called the *folding-unfolding*

method, has been developed[4] and, as in the case of CShMs, the computational bottleneck in the calculation of CSMs is the minimization over all $N!$ possible vertex pairings between the problem and reference structures Q and T.

Lately we have found an alternative way to search for the desired structure T with a given symmetry that is based on the calculation of the so-called *symmetry operation measures*[5]. Given a symmetry point group $G = R_i$ with h symmetry operations and a problem structure Q, we can define for each operation R_i a symmetry operation measure $Z(Q, R_i)$ as:

$$Z(Q, R_i) = 100 \cdot \min \frac{\sum_{k=1}^{N} |q_k - R_i q_k|^2}{4 \sum_{k=1}^{N} |q_k - q_0|^2} \tag{1.4}$$

where we now compare the problem structure Q with the image obtained from the symmetry operation acting on it. The minimization procedure is somewhat simplified in this case, since we need not minimize $Z(Q, R_i)$ with respect to translations (if R_i is a symmetry operation belonging to a point group, then it must leave a single point invariant and this point coincides with the geometric centre of Q) and the relative size of both structures (since point group symmetry operations do not change the size of the object). The only geometrical minimization that is needed in this case is with respect to the relative orientation of the problem structure and its image, or equivalently to minimize the measure with respect to the spatial orientation of the symmetry element that is associated with the symmetry operation R_i. As in the other cases, an optimization of all possible pairings between vertices of the problem and reference structures is also needed, although in this case there is an additional restriction to the possible permutations, since only G-symmetry preserving permutations should be considered.

It can be shown that the value of the CSM for a given group can be easily evaluated using the symmetry operation measures for the h operations of the group[6]:

$$S(Q, G) = \frac{1}{h} \sum_{i=1}^{h} Z(Q, R_i), \tag{1.5}$$

where the symbol $Z(Q, R_i)$ is used to indicate that the minimization procedure to find the symmetry operation measures must be performed simultaneously for all operations in the group while imposing the necessary constraints

[4]See ref. 1 for a detailed description of this algorithm.

[5]M. Pinsky, D. Casanova, P. Alemany, S. Álvarez, D. Avnir, C. Dryzun, Z. Kizner, A. Sterkin *J. Comput. Chem.* **2008**, 29, 190–197.

[6]M. Pinsky, C. Dryzun, D. Casanova, P. Alemany, D. Avnir, *J. Comput. Chem.* **2008**, 29, 2712–2721.

to keep the relative orientation of the different operations fixed (for a C_{2h} group, for example, one must perform the minimizations necessary to obtain the symmetry operation measures while keeping the C_2 axis perpendicular to the mirror plane). In this respect, $Z(Q, R_i)$ are not real symmetry operation measures, since the value of $Z(Q, R_i)$ may be lower if these constraints are released.

1.2 Problem

As discussed above, one of the main problems in the application of continuous shape or symmetry measures to structural chemistry is the $N!$ dependence of the algorithms devised for the numerical calculation of CShMs and CSMs. A typical study in structural chemistry starts with a search for a given fragment formed by N atoms in a database such as the *Cambridge Structural Database,* containing information on the crystal structures for over 400,000 chemical compounds. In this type of search, the desired fragment is usually described using the connectivity between atoms, although additional geometrical restrictions on bond distances and/or angles may be imposed to limit the search. As a result, satisfying the conditions specified in the search, the user obtains for each fragment a set of N coordinates, one for each atom in the fragment. CShMs and/or CSMs are calculated afterwards for each fragment and used, for example, to classify the retrieved fragments according to their coordination geometries. The plotting of $S(Q, P)$ vs. $S(Q, P')$, where P and P' are two different reference polyhedra, is called a *shape map* and has been shown to be a very useful tool in detecting structural trends for different families of compounds (Figure 1.3).

With our current programs, this process is feasible for sets of around 10,000 fragments with up to approximately 10 vertices. Although we have been able to calculate CSMs for a small set of structures with up to 15 atoms, the study of structures with more than 20 atoms is impossible nowadays if no additional simplifying assumptions are introduced. It is therefore crucial for the application of CShMs and CSMs in structural chemistry to develop efficient algorithms to deal with the optimum vertex pairing problem.

Although there are some differences in the treatment of the minimization of vertex pairings in the calculation of CShMs and CSMs, the basic problem remains the same:

> *Is there an efficient way to avoid the sweeping across all $N!$ index permutations to find the optimal vertex pairing between the problem and the reference structures?*

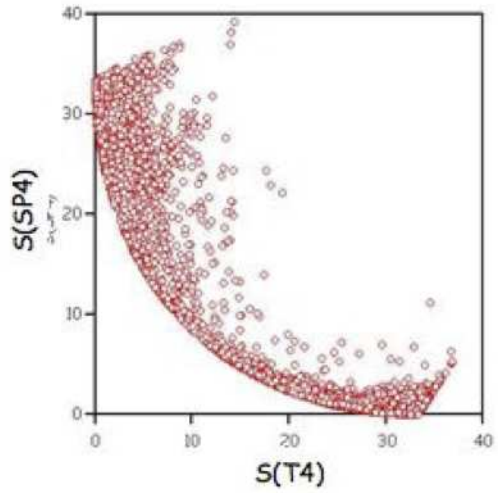


Figure 1.3: Shape map for over 13,000 tetracoordinated ML_4 transition metal complexes classified according to their tetrahedral (T4) and square-planar (SP4) continuous shape measures.

In the following section we will outline the current algorithm implemented in our program as well as some considerations related to the chemical properties of the analyzed structures that may, in some cases, provide additional information useful to avoid a search of all possible permutations.

In order to simplify the procedure, in a first step the size of Q is normalized so that

$$N = \sum_{k=1}^{N} |q_k - q_0|^2. \tag{1.6}$$

Omitting the arbitrary factor of 100 that appears in the equations above, the minimization in Eq. (1.1) is performed in two steps:

$$S(Q,P) = \frac{1}{N} \min_{P_m} \left(\min_{A,\mathbf{R},t} \sum_{k=1}^{N} |q_k - p_k|^2 \right) = \frac{1}{N} \min_{P_m} S_{N,m}(Q,P) \tag{1.7}$$

where we first minimize the sum of the square of the distances between vertices with respect to size, orientation, and translation $(A, \mathbf{R}, t)$ for a given vertex pairing (P_m) to obtain $S_{N,m}(Q,P)$, and afterwards we sweep through all permutations of vertices of the reference shape to find the vertex pairing with the smallest value of $S_{N,m}(Q,P)$, where the subindex N is used to highlight that all N vertices of P and Q are being compared.

In the current algorithm for calculating CShMs, if the number of vertices is large enough (polyhedra with more than 7–8 vertices) the set of N vertices is divided into n_L subsets of $L < N$ vertices, and, if necessary, an additional set with $N - n_L L$ vertices. Given a fixed vertex pairing, it is possible to define n_L functions $S_{L,m}(Q, P)$ analogous to $S_{N,m}(Q, P)$ in which only the vertices of a given subset are compared and the optimal values for A, $\mathbf{R}$, t for each subset are calculated. Since all these functions are positive, for a given vertex pairing (P_m) the following inequality must hold:

$$S_{N,m} \geq \frac{L}{N} \sum_{i=1}^{N} S_{L,mi}. \tag{1.8}$$

This expression shows that, if any of the individual $S_{L,m}$ values is found to be larger than a previously stored value of $S_{N,m}$, then it is not necessary to calculate values of $S_{N,m}$ for all $(N-L)!$ permutations with the same vertices in this partition. Tests using $L = 4$ have shown that, for structures with more that $N = 8$ vertices, the algorithm based on partitioning the set of vertices in different subsets is, in general, more efficient than sweeping all $N!$ permutations, since for most of the analyzed structures a large number of vertex pairings can be discarded. This algorithm has allowed us to reach the calculation of CShMs for structures with up to around 20 vertices without imposing any additional approximation to discard vertex pairings.

Besides this algorithm, the program allows to introduce additional restrictions in order to reduce the number of permutations that must be analyzed. The simplest one is to restrict the analyzed permutations just to a given set that is provided in the input by the user. This is an extremely fast solution that is mostly applicable in cases in which the problem and reference structures are closely related and a visual inspection is able to provide unambiguously the optimum vertex pairing (or a small set of candidates for it). Although extremely efficient, this solution is not applicable to the automatic computation of CSMs for large sets of data, since it requires a previous visual inspection to decide the permutations that will be analyzed, and it is also susceptible to major errors, since it depends on the subjective choice of a limited set of vertex pairings, a task that is not always easy.

Another approach that may be useful is to divide the vertices into a number of subsets and to perform all permutations only within each of these subsets. This approach can be applied if chemical information is available (we can, for example, restrict the number of permutations in the structure of a hydrocarbon C_nH_m molecule to those which interchange only carbon atoms with other carbon atoms and hydrogen atoms with other hydrogen atoms). Although useful, this approach is not very efficient, since the limit of feasible

permutations within each subset is easily reached. Besides, it is often desirable to compare the structures of molecules built from different atoms in order to highlight structural similarities that go beyond the chemical nature of the molecules, a situation where it is difficult to define the different sets of atoms unambiguously. This technique has also been applied to molecular clusters that are built in concentric layers of atoms. In this case, it is sometimes possible to restrict permutations to atoms within each layer (or at least between atoms in a given layer or in the two closest ones).

Related to this approach, it is also often possible to use information related to the graph representing the chemical bonds in a molecule. For the case of CSMs, we have devised an algorithm that classifies the different vertices into subsets according to their connectivity in the molecular graph and restricts the permutations only between atoms that have the same connectivity (Figure 1.4). Although this algorithm saves a large amount of time in some cases, we are aware that it does not exploit the full symmetry of the molecular graph, a direction that is probably worth exploring.

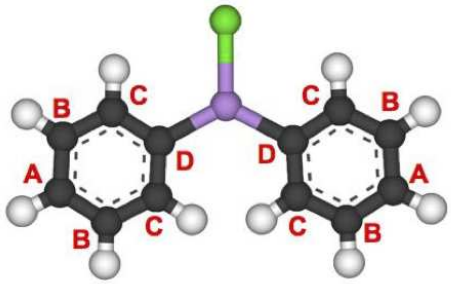


Figure 1.4: Ball and stick representation of diphenyl-chlorarsine showing the partitioning of the 12 carbon atoms into 4 different sets A, B, C, and D according to their connectivity in the molecular graph.

Although molecular graphs may be useful in limiting the number of vertex pairings to be analyzed, their use is restricted for various reasons. Two different questions arise, both related to the possibility of defining the molecular graph unambiguously.

A molecular structure is a graphical representation of the average position of the atomic nuclei in a given molecule. These nuclei are in fact in constant movement around their so-called *equilibrium positions*. If a nucleus is separated from its equilibrium position, the molecular energy is increased. For a polyatomic molecule with N atoms, the potential energy surface (a function that yields the energy of the molecule given the position of the nu-

clei) shows, in general, more than one equilibrium nuclear configuration in which the nuclei are arranged in different positions. Each of these nuclear configurations gives a different molecular graph in which the connections between atoms correspond to chemical bonds. The criteria for deciding whether there is a bond or not between two atoms is not, however, unambiguous. The potential energy surface for a diatomic molecule is a simple function of the internuclear distance that can be conveniently described by the Morse function (Figure 1.5, in blue).

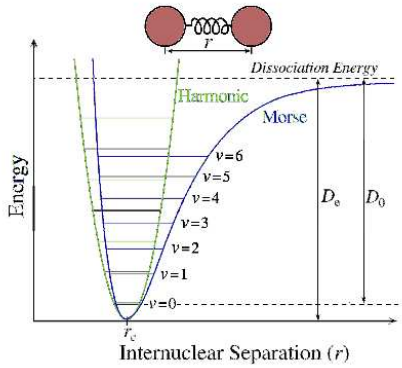


Figure 1.5: Potential energy curves for a diatomic molecule. The blue curve corresponds to the Morse function while the green curve corresponds to the harmonic approximation, used to study small deviations of the nuclei from their equilibrium positions.

The Morse function is characterized by two parameters, the equilibrium distance r_e and the dissociation energy D_e. The first corresponds to the internuclear distance for which the energy is minimal, while D_e indicates the energy that is needed to separate the two atoms (to break the bond). Although in a polyatomic molecule it is not possible to write a simple mathematical expression for the potential energy surface, it is often possible to analyze how the energy changes when a pair of atoms is separated, and these energy changes can also be described by a Morse function. By analyzing different molecular structures, it is easy to find similar r_e and D_e values for a given pair of atoms, and this allows chemists to visualize a molecule as a set of atoms linked by a number of chemical bonds. In hydrocarbons, for example, for C–C pairs typical r_e distances around 1.5 Å are found and programs used to visualize molecular structures will plot a bond between carbon atoms at a distance shorter than approximately 1.6 – 1.7 Å. If the minimum in the

Morse curve between two atoms is small, the energy rises very steeply with distance and there is a small variation in the bond distances between these two atoms in different molecules. This is not, however, always the case. There are situations where there are "weak bonding" interactions between atoms, giving a broad Morse curve for which variations around r_e represent only small energy changes. This is translated in a large dispersion of values in the internuclear distances in different molecules containing this pair of atoms and the difficulty in establishing a clear criterion of what should be considered a bond and, hence, in defining the molecular graph. This situation is found, for example, in metal clusters like Au_{28} (Figure 1.6), where there is a total of 94 Au–Au distances between 2.7 and 3.4 Å with an average Au–Au distance around 2.9 Å.

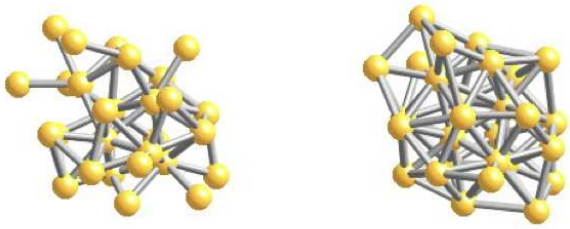


Figure 1.6: Ball and stick representation of an Au_{28} cluster plotting bonds for distances below 2.9 Å (left) or 3.4 Å (right).

This example shows the first limitation of the application of the molecular graph to the computation of CSMs and CShMs. Changing the distance to define an Au–Au bond between 2.7 and 3.4 Å gives different molecular graphs that lead to different values for the symmetry or shape measures when using the graph to limit the number of vertex pairings to be searched, or even to cases where these measures cannot be calculated because the possible number of vertex pairings within each of the vertex subsets is too high.

The other problem is related to the application of molecular graphs to continuous shape measures. While in continuous symmetry measures the problem structure and the reference structure (the image of the problem structure under the application of a symmetry operation) share the same molecular graph, this is not necessarily true for CShMs. Although it is possible to establish a standard molecular graph for the reference structure, this graph will, in general, only be preserved for small structural distortions. For larger distortions, chemical bonds may be broken or formed and the graph

of the problem structure will change as shown in the example in Figure 1.6. When analyzing large sets of fragments retrieved from a structural database, it is very difficult to detect these changes, since the definition of the molecular graph depends, as explained above, on the chemical nature of the atoms and it is quite difficult to develop an algorithm to establish unambiguously the equivalence of a given graph with that of the reference structure.

Figure 1.7: Ball and stick representation of two deformations of a tetrahedral fragment with different molecular graphs.

In conclusion, let us restate our problem:

> *Is there an efficient way to avoid sweeping across all $N!$ index permutations to find the optimal vertex pairing between the problem and the reference structures?*

As we have shown, the problem can be addressed in several ways, depending on the additional information at hand for each case. However, the question remains open for difficult cases like the Au_{28} cluster shown in Figure 1.6, where the molecular graph is of little help and the number of permutations is too high to be treated in currently available computers.

Delay problems in networks

by Jérôme Galtier, Orange/France-Telecom

2.1 Overview

The world of telecommunications is now facing a new challenge on end-to-end delays that are raised by interactive applications. Video communications, interactive games, after voice IP applications such as Skype, force us to rethink the network.

Data networks are subject to congestion, thereby the delay in crossing the network may be long enough to put consumers off continued use of the network. We present the problem of determining routing within a given network to minimise the delay or keep it within certain boundaries.

There are some positive and NP-completeness results for these problems, and also some open problems, limitations and questions worth studying around them.

2.2 Some details

The quality of service, QoS for short, offered by a telecommunication network can be expressed either in terms of delay or in terms of loss. The *delay* is the amount of time spent by an element, typically a packet, of flow communication in order to go across a component of the network. The *loss* is calculated as the percentage of the number of packets committed to the network which are lost during the transmission. It can be said that, if a packet is lost, its transmission delay is infinite. We will study problems related to routing flows in a network with delay constraints. The transmission delay can be expressed through the transmission links of the network; in that case it is the time needed to go through a link. Or we may consider end-to-end delay, in which case it is the total amount of time needed to cross the network. The end-to-end delay may be the sum of the delays of the transmission links which form a route. In that case, the criterion is said to be *additive*.

It should be noted that this is not always the case if, for instance, a packet is split into smaller ones in certain links. However, here we are focusing on additive criteria for QoS, and on the end-to-end delay.

There are two problems to be studied in this context. The first concerns the minimisation of the mean delay over the set of transmission links. This problem has already been studied using methods that assume differentiability. Unfortunately the congestion functions have a vertical asymptote when capacity is reached, and this can cause numerical instability.

The second problem is that of computing minimum cost routing subject to a maximum admissible delay. There are alternative contexts for this problem. The delay may be a fixed number associated to each edge, or the delay of an edge may be a function of the value of the flow circulating on that edge and the capacity of the edge, which is assumed to be fixed (thus, when the flow is approaching capacity, we find a vertical asymptote). In any case, the aim for this second problem is to determine a routing of the demand flow which minimises the routing cost for a given linear function of the flow and such that the total delay incurred in the network in order to go from the source node to the demand node is bounded by a given value.

Voltage drop in on-chip power distribution networks

by Josep Rius Vázquez, Department of Electronic Engineering, Universitat Politècnica de Catalunya, Barcelona, Spain

3.1 Introduction

The increase in complexity and in integration density in present-day digital circuits makes the interaction between the power supply lines and the circuit activity increasingly close. The *Power Distribution Network* (PDN) of an integrated circuit is a distributed system that can be approximately modeled as a large RLCG mesh. As a consequence, the current peaks produced during the circuit activity perturb the PDN, thus changing the effective supply voltage of logic gates. This perturbation, or *Power Supply Noise* (PSN), increases the gate delay or can be the cause of reliability problems, thus reducing the circuit performance. In this way, to guarantee a PSN level within a specified value, the design of a PDN becomes an important concern [1].

There are simulation tools integrated in the design flow specifically dedicated to estimating the PSN level of a digital IC as a function of location and time. Such tools usually require the full layout of the circuit and the detailed specification of its activity to obtain the PDN response. Therefore, they require a large amount of information and long computer times to get accurate results. But worse than this is that, as a result of time to market constraints, it is extremely expensive to re-design the whole or a part of the PDN if the simulated results show an excessive PSN anywhere. So, these tools are extremely useful for validating the final layout of a given PDN, but they are not well suited to re-designing the PDN and recalculating its response interactively.

As a consequence, there is a need for exploratory tools which trade off accuracy for interactivity, thus allowing the exploration of the PDN design space to take the best decisions during the early phases of its design, when little information on the IC is available.

Typical issues that appear during such early phases of PDN design are:

- Number and distribution of supply pads.
- Number, pitch and placement of grid segments and stripes.
- Number and distribution of decaps.
- Calculation of static IR-drop in both value and location.
- Dynamic features of PSN, peak value and width of its pulses.

3.2 Explorer for pre-layout power grid construction

If the power grid is sufficiently dense, the PDN of a digital IC can be approximately described as a rectangular continuum medium with given electrical properties: sheet resistance and inductance, leakage and capacitance per unit area. Voltage and current at any place and time can be described by a non-homogeneous two-dimensional wave equation which, in theory, can be solved if boundary and initial conditions and the non-homogeneous term (current density, $J(x, y, t)$) are given. To date, this problem has been solved only by numerical techniques. However, under the simplifying assumption that the supply voltage and the current density function are not time dependent, the wave equation collapses to the Poisson PDE and the problem of finding the maximum drop voltage (the so-called *maximum static IR-drop voltage*) can be analytically solved for a number of cases of engineering interest.

Furthermore, the boundary conditions depend on the type of chip package. For wire-bond packages, the IC boundaries can be approximated to have constant voltage and the Poisson equation can be solved in closed form if the current density function J is constant.

For flip-chip packages, the problem is much more difficult. Here, the chip boundaries have the normal voltage derivative equal to zero (zero current at the boundaries) and the constant voltage is defined only at the supply pads distributed in the whole IC area. This problem has been criticised for looking for a direct solution of the Poisson equation assuming constant current density in the whole IC, thus obtaining an explicit formula for the maximum static IR-drop voltage under these simplifying assumptions [2].

An outline of another possible approach to solve this simplified problem is the following: in a two-dimensional domain it is possible to find the potential at any observation point $V(x, y)$ when two line current sources inject and draw current at known points. One of such line currents is fixed at one

supply pad and the other may be at any place inside the chip. To fit the boundary conditions (zero current at the chip boundaries), it is necessary to calculate $V(x, y)$ for an infinite two-dimensional array of such line current sources, thus obtaining the solution for $V(x, y)$ as a doubly infinite series. In addition, instead of having the second line current source at a prescribed point, the real IC has a distributed current spread over its whole area, which is defined by a current density function J. So, an additional double integration is necessary to find the potential at any point due to such a distributed current. The final result can be further simplified by using the concept of *geometrical mean distance* (GMD) to eliminate the necessity for double integrals. Another possibility is to partially or totally eliminate the double infinite series by using trigonometric or elliptic functions, thereby reducing the problem to an integration of such functions. In both cases, the fact that the influence of the line current sources on the potential at the observation point exponentially decreases with distance can be used to further simplify the final expressions.

It is expected that this or another similar approach can be adapted to find the maximum static IR-drop voltage of an IC with flip-chip package under realistic conditions (current density function J not constant along the chip). The final goal is to obtain a pre-layout tool implementing the solution which would require as inputs the key PDN parameters, as IC dimensions, the distribution of the current density function J, and basic technological data such as metal sheet resistances, pitches, metal widths, etc.

Bibliography

[1] M. Popovich, A. V. Mezhiba, E. G. Friedman, *Power Distribution Networks with On-Chip Decoupling Capacitors*, Springer, 2008.

[2] K. Shakeri, J. D. Meindl, Compact physical IR-drop models for chip/package co-design of gigascale integration (GSI), *IEEE Transactions on Electron Devices* **52 (6)** (2005), 1087–1096.

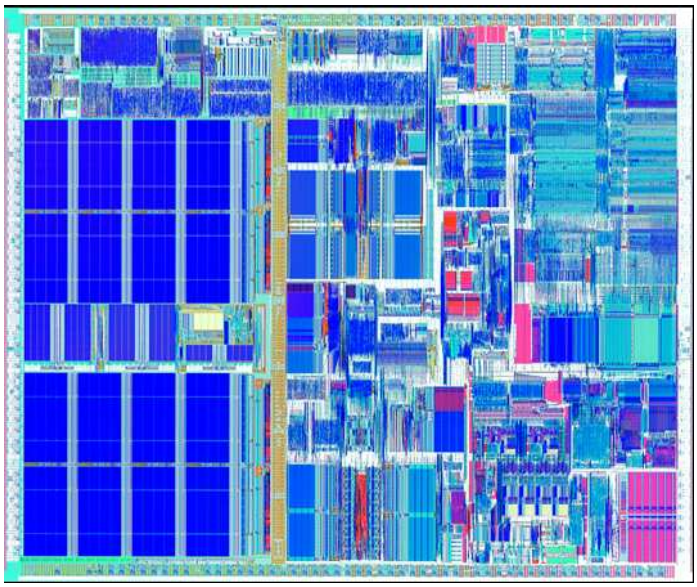


Figure 3.1: Image of a wire-bond chip.

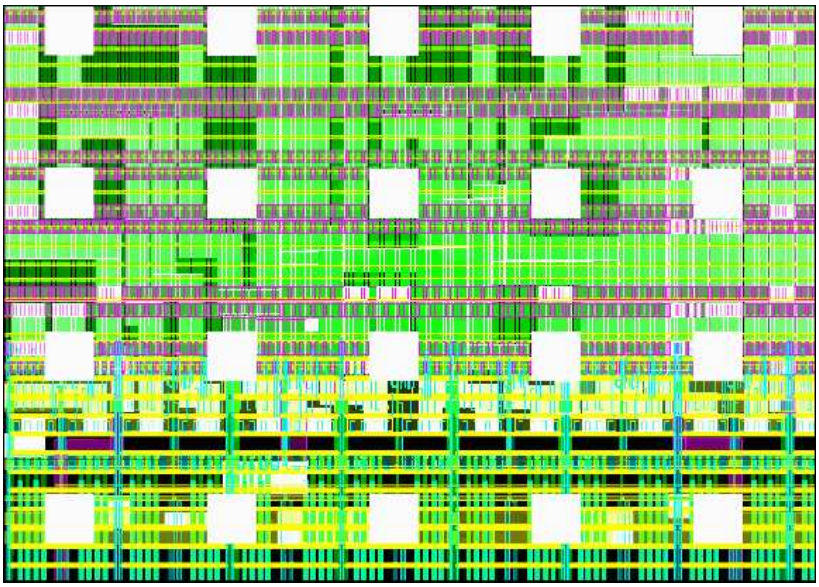


Figure 3.2: Image of a flip-chip.

Answers to the Problems

Continuous symmetry and shape measures

by Guillem Perarnau-Llobet,[1] Julian Pfeifle[2] and Jordi Saludes[2]

1.1 Introduction

A problem in computational chemistry posed during GEMT 2009 at UPC was to find the optimal affine transformation between two point sets $X, Y \subset \mathbb{R}^3$ of n points each that minimizes a certain similarity measure. Given a bijection $\pi\colon X \to Y$, the optimal affine transform sending $\pi(Y)$ to X can be computed efficiently by analytic means [3]. The crucial bottleneck encountered in previous work lies with the combinatorial complexity of having to enumerate all $n!$ permutations of these point sets to find the best affine transform.

In this paper, we present an algorithm that *approximately* matches X and Y using affine transformations, and returns the best correspondence between the transformed sets. From this, the best global affine transform can then be computed analytically.

Our strategy is to first translate X and Y so that their respective barycenters lie at the origin, and then scale each set so that the variation of the set of distances to the origin is the same. The only remaining ingredient is to find a rotation $R \in \mathrm{SO}(3)$ that makes $R(X)$ and Y agree "as far as possible".

1.1.1 Error measures

We will take the sum of squared distances of matching points. Another viable option would be Hausdorff distance [1].

[1]Departament de Matemàtica Aplicada 4, UPC
[2]Departament de Matemàtica Aplicada 2, UPC

1.2 The Cayley chart of SO(3)

The Lie group SO(3) of orthogonal 3×3 matrices with determinant 1 has many different charts. The most important for us is that given by the *Cayley transform*

$$\begin{aligned} \mathcal{C}:\ \mathrm{so}(3) &\longrightarrow \mathrm{SO}(3) \smallsetminus M \\ A &\longmapsto (I-A)(I+A)^{-1}. \end{aligned}$$

It establishes a bijection between skew-symmetric matrices

$$A \;=\; \mathrm{sk}(x,y,z) \;=\; \begin{pmatrix} 0 & x & -y \\ -x & 0 & z \\ y & -z & 0 \end{pmatrix} \;\in\; \mathrm{so}(3)$$

with $x, y, z \in \mathbb{R}$ and the set $\mathrm{SO}(3) \smallsetminus M$, where M denotes the set of rotation matrices that have -1 as an eigenvalue. Specifically, it takes A to

$$\mathcal{C}\big(\,\mathrm{sk}(x,y,z)\big) \;=\; \frac{1}{\Delta}\begin{pmatrix} 1+x^2-y^2-z^2 & 2\,x\,y-2\,z & 2\,(y+x\,z) \\ 2\,(x\,y+z) & 1-x^2+y^2-z^2 & -2\,x+2\,y\,z \\ -2\,y+2\,x\,z & 2\,(x+y\,z) & 1-x^2-y^2+z^2 \end{pmatrix},$$

where $\Delta = 1 + x^2 + y^2 + z^2$.

The inverse map is given by the same expression,

$$\mathcal{C}^{-1}(Q) \;=\; (I-Q)(I+Q)^{-1} \qquad \text{for } Q \in \mathrm{SO}(3) \smallsetminus M.$$

We need to find the set of rotation matrices that map a point with spherical coordinates (θ_1, φ_1) in the 2-dimensional sphere S^2 to the point with spherical coordinates (θ_2, φ_2). Elementary calculations yield the following result:

Proposition 1.2.1. *The inverse image under the composite map $\mathcal{C} \circ \mathrm{sk}$ of the set of rotations that send $u = (\theta_1, \varphi_1)$ to $v = (\theta_2, \varphi_2)$ is the affine line ℓ in $(\mathbb{R}^3, (x, y, z))$ given by*

$$x = \frac{\cos(\varphi_2) - \cos(\varphi_1) + y\,(\cos(\theta_2)\,\sin(\varphi_2) + \cos(\theta_1)\,\sin(\varphi_1))}{\sin(\varphi_2)\,\sin(\theta_2) + \sin(\varphi_1)\,\sin(\theta_1)},$$

$$z = \frac{y\,(\cos(\varphi_2) + \cos(\varphi_1)) - \cos(\theta_2)\,\sin(\varphi_2) + \cos(\theta_1)\,\sin(\varphi_1)}{\sin(\varphi_2)\,\sin(\theta_2) + \sin(\varphi_1)\,\sin(\theta_1)}.$$

If $u = (u_1, u_2, u_3)$ and $v = (v_1, v_2, v_3)$ are the Cartesian coordinates of u, respectively v, then a point p on ℓ and a direction vector a for ℓ are given by

$$p = \left(\frac{-u_3 + v_3}{u_2 + v_2},\, 0,\, \frac{u_1 - v_1}{u_2 + v_2}\right), \qquad a = (u_1 + v_1,\, u_2 + v_2,\, u_3 + v_3).$$

1.2.1 Computing rotations

We will try to match a given triangle in the reference point set to every suitable triangle in the problem set using a rotation. From the above, it is clear that finding the optimal rotation that achieves this corresponds to intersecting the three lines $q_i + t_i v_i$ for $i = 1, 2, 3$ in the Cayley parametrization space. If the triangles in question are congruent, these three lines will meet at a single point, so the problem is overdetermined; if the triangles are not congruent, however, the three lines will not intersect at all. We therefore choose to solve the problem of minimizing the sum of squared distances

$$D \;=\; \sum_{i,j}^{3} \|q_i - q_j + t_i v_i - t_j v_j\|^2.$$

By computing the gradient of D with respect to the unknowns t_i, we obtain the equivalent system of linear equations

$$\sum_{j \neq i} (q_j - q_i) \cdot v_i \;=\; 2t_i \|v_i\|^2 - \sum_{j \neq i} t_j v_i \cdot v_j \qquad \text{for } i = 1, 2, 3,$$

which expressed in matrix form reads as follows:

$$\begin{pmatrix} 2\|v_1\|^2 & -v_1 \cdot v_2 & -v_1 \cdot v_3 \\ -v_2 \cdot v_1 & 2\|v_2\|^2 & -v_2 \cdot v_3 \\ -v_3 \cdot v_1 & -v_3 \cdot v_2 & 2\|v_3\|^2 \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = \begin{pmatrix} \sum_{j \neq 1} (q_j - q_1) \cdot v_1 \\ \sum_{j \neq 2} (q_j - q_2) \cdot v_2 \\ \sum_{j \neq 3} (q_j - q_3) \cdot v_3 \end{pmatrix}.$$

As a candidate for the optimal rotation, we take the one corresponding to the barycenter of the solution points: $\frac{1}{3} \sum_{i=1}^{3} q_i + t_i v_i$. Alternatively, we could consider the minimization problem

$$D' = \sum_{i=1}^{3} \|q_i + t_i v_i - p\|^2$$

with unknowns $t_i \in \mathbb{R}$ and $p \in \mathbb{R}^3$.

1.3 Approximate affine point matching

1.3.1 Overview of the algorithm

The input data are two ordered sets $X = (x_1, \ldots, x_n)$, $Y = (y_1, \ldots, y_n) \subset \mathbb{R}^3$ of n points each. We want to compute a permutation $\pi \in S_n$ such that

the ordered set $Y_\pi = (y_{\pi(1)}, \ldots, y_{\pi(n)})$ approximately minimizes the *shape measure* $S(X,Y) = \min_{\pi \in S_n} S_\pi(X,Y)$, where

$$S_\pi(X,Y) \;=\; \min_{f\,\text{affine}} \frac{\displaystyle\sum_{i=1}^{n} \|x_i - f(y_{\pi(i)})\|^2}{\displaystyle\sum_{i=1}^{n} \|x_i - \beta\|^2}.$$

Here $\beta = \frac{1}{n}\sum_{i=1}^{n} x_i$ is the barycenter of X, and we take the minimum over *all* affine transformations of $\mathbb{R}^n$. We find a permutation that approximates $S_\pi(X,Y)$ using only a finite number of such transformations.

The first steps are to translate the barycenters of X and Y to the origin, and to scale both sets so that the variances of their distances to the origin equal some fixed value. We retain the names X and Y for these translated and scaled sets. After this, we need to optimize over all rotations.

To any rotation $R \in \mathrm{SO}(3)$ we associate the map $\pi = \pi(R) : \{1, \ldots, n\} \to \{1, \ldots, n\}$ that assigns to each point $x_i \in X$ the point $y'_{\pi(i)} \in Y' = R(Y)$ closest to it in the Euclidean norm. In favorable cases, for example when Y is an affine image of a slight perturbation of X, this map π is actually a permutation of $\{1, \ldots, n\}$. The optimal affine transform that maps X to Y_π can then be found by analytical means.

Denote the set of triangles formed by points in X and Y by $\mathcal{T}_X$ and $\mathcal{T}_Y$, respectively. To find a good set of candidate rotations, we first choose a certain (relatively small) subset $\mathcal{T}'_X \subset \mathcal{T}_X$ of the triangles in X. For each such triangle $T_X \in \mathcal{T}'_X$, we iterate over *all* triangles $T_Y \in \mathcal{T}_Y$, and for each pair (T_X, T_Y) we find the rotation $R \in \mathrm{SO}(3)$ that most closely maps T_X onto T_Y using the methods in the preceding section. We then apply R to the entire set X, find the corresponding optimal permutation $\pi(R)$, and calculate the associated shape measure $S_{\pi(R)}(X,Y)$. Finally, we return the permutation corresponding to the best rotation among all those seen throughout the process.

1.3.2 Implementation details

Choosing $\mathcal{T}'$

In general, the centered and scaled sets X and Y will not lie on a sphere. Thus, two points $x_i \in X$ and $y_j \in Y$ will generally have different norms. If this is true, it makes little sense to try to rotate x_i into y_j. Reciprocally, if X and Y are almost affine images of each other, it stands to reason that the distribution of the norms of their elements will be similar.

Algorithm 1 Pseudocode of the matching algorithm

```
1: procedure MATCHING(X, Y)
2:     X: the reference points
3:     Y: the problem points
4:     Set X := scale(center(X))
5:     Set Y := scale(center(Y))
6:     global_error := 0
7:     x_1 := closest_to_origin(X)
8:     x_2 := furthest_to_origin(X)
9:     for all triangles T_X = {x_1, x_2, x} ∈ T_X do
10:        for all examinable triangles T_Y = {y_1, y_2, y_3}, where y_i ∈ Y do
11:            R := optimal_rotation(T_X,T_Y)
12:            X_R := rotate(X,R)
13:            if matching_error(X_R,Y) < global_error then
14:                actualize the matching and its error
15:            end if
16:        end for
17:    end for
18:    Set X̂ := scale(center(specular(X)))
19:    x̂_1 := closest_to_origin(X̂)
20:    x̂_2 := furthest_to_origin(X̂)
21:    for all triangles T_X̂ = {x̂_1, x̂_2, x̂} ∈ T_X̂ do
22:        for all examinable triangles T_Y = {y_1, y_2, y_3}, where y_i ∈ Y do
23:            R := optimal_rotation(T_X̂,T_Y)
24:            X_R := rotate(X,R)
25:            if matching_error(X̂_R,Y) < global_error then
26:                actualize the matching and its error
27:            end if
28:        end for
29:    end for
30: end procedure
```

In the hope of rapidly and accurately capturing the shape of Y, we therefore choose the points $x_{\min}$ and $x_{\max}$ of minimal and maximal norm to always form part of the initial triangle T_X. This leaves us with a linear number of initial triangles:

$$\mathcal{T}_X \;=\; \big\{\, \mathrm{conv}\{x_{\min}, x_{\max}, x\} \;:\; x \neq x_{\min}, x_{\max} \big\}.$$

Sometimes rotation does not suffice because the X and Y have different orientations. Hence we may define $\hat{X} = \sigma(X)$, where σ is some reflection, for example that with respect to the plane $\{x = 0\}$. Analogously, we define $\mathcal{T}_{\hat{X}} = \sigma(\mathcal{T}_X)$. Note that $\hat{x}_{\min} = \sigma(x_{\min})$ and $\hat{x}_{\max} = \sigma(x_{\max})$.

Finding the error given R

To calculate the error induced by a rotation R, we must compute the map $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$. We use a *k-d-tree* built from X to rapidly query the closest corresponding rotated point. This gives us an injective map that is not necessarily exhaustive; however, this has always been the case in the experiments we have conducted. Note that we build the *k-d*-tree on the fixed reference set, so that we only have to execute this preprocessing once.

Sorting pairs of points by difference in norm

Another crucial optimization to find the optimal rotation is the following. We calculate the norms of all points in $\mathcal{T}_X$ and Y, and sort the list $\big(\big|\|x\| - \|y\|\big| : x \in \mathcal{T}_X,\ y \in Y\big)$ of absolute values of their differences by size. We then use the two triangles formed by the first three pairs of points from this sorted list to calculate the first candidate rotation R. Intuitively, this makes sense because we expect these triangles to be quite similar. We then proceed with other candidate triangles from the beginning of the list. One must take a little care to check that each triple of the selected pairs really consists of six distinct points.

Due to this optimization, for each triangle in $\mathcal{T}_X$ we only examine certain triangles in $\mathcal{T}_Y$ (the first according to this sorted list), and this improves the execution time.

1.4 Results

We experimentally evaluate the efficiency of our algorithm in terms of time complexity and quality of the solution found. We test it in the following examples:

- A 7-vertex polyhedron with a central vertex.
- A cluster Au_{28} consisting of 28 gold atoms.
- Instances of a scalable artificial dataset. In order to be able to experiment with large datasets, we have implemented a program that outputs an arbitrarily large point cloud and a perturbation of it, and allows the amount of perturbation to be tuned.

Each instance is accompanied by a perturbed version, which we then try to match.

1.4.1 7-vertex polyhedron with a central vertex

This dataset consists of a polyhedron with 7 vertices on its convex hull, along with another point at the barycenter. Exhaustive enumeration confirms the permutation output by our algorithm to be the optimal one.

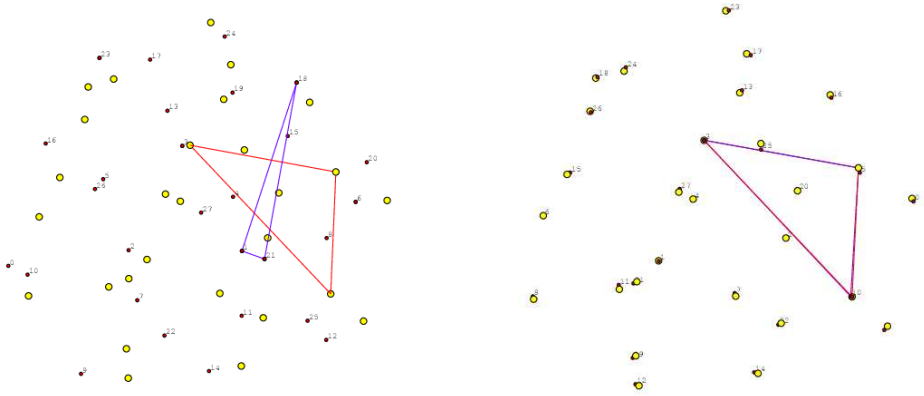
We do not apply an optimal analytical minimization of the distances between matched points; hence, we obtain $S(Q, P) \leq 0.93518$ instead of the optimal $S(Q, P) = 0.47764$. This shows that, despite computing the optimal permutation, we overestimate $S(Q, P)$.

The algorithm spends approximately two seconds on this example.

1.4.2 Au_{28}

For this 28-point instance, it is computationally out of the question to enumerate all $28! \approx 3 \times 10^{29}$ permutations. Other heuristic methods [2] have obtained a permutation of the nodes in Au_{28} that lead to the bound $S(Q, P) \leq 1.69182$. This heuristic consists of finding an optimal plane to apply a specular symmetry, assuming that no rotation is needed.

Here the specular symmetry approach is very important, since the two datasets do not have the same orientation. Our algorithm finds a substantially different permutation, leading to an upper bound for the symmetry measure of $S(Q, P) \leq 0.23426$, which improves the former. This solution is found in 20 seconds.



(a)Au_{28} and ideal polyhedra before rotation. (b)Au_{28} and ideal polyhedra after rotation.

Figure 1.1: The Au_{28} dataset and its perturbed version before and after rotation, but *after* applying the mirror symmetry. In both images, we have marked the triangles that select the optimal rotation.

1.4.3 Scalable artificial dataset

Finally, to test the real time complexity and the quality of the solution when both the size increases and the quantity of perturbation varies, we use our artificial dataset generator. The time spent by the algorithm depending on the size of the point cloud is shown in Figure 1.2. We also show our upper bound on the symmetry measure compared with the real one given a fixed size and varying the perturbation of the points.

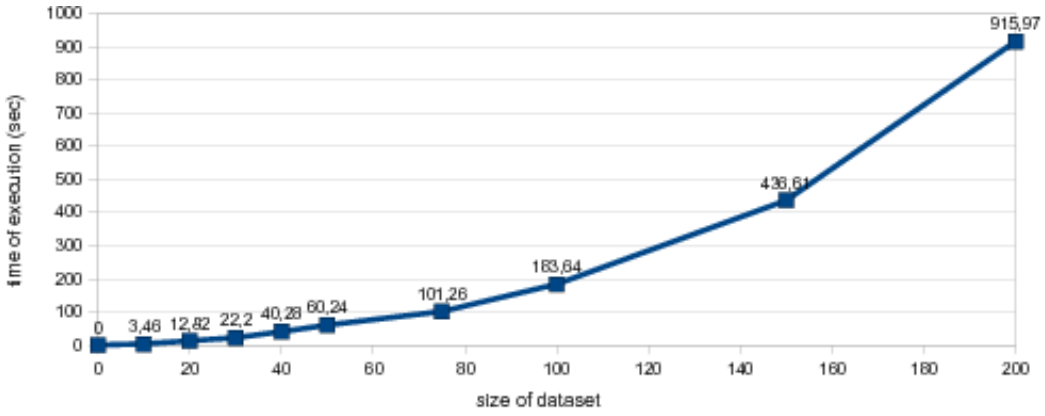


Figure 1.2: Execution time of the matching algorithm in an artificial set of points.

1.5 Other ideas

1.5.1 Local minimization techniques

Another possibility is to define a smooth function such as the one given by an attractive potential between the reference points $P = \{p_i\}$ and the problem points $Q = \{q_i\}$,

$$V(P, Q) = \sum_{i,j} \phi(\|p_i - q_j\|),$$

where $\phi(r) = -r^{-\alpha}$ for $\alpha > 0$. Then, given a rotation R and a local chart containing R with coordinates x, y, z, we will try a steepest descent method on $(x, y, z) \mapsto V(R(x, y, z)P, Q)$. In this way R is changed slightly to get a better match.

To avoid falling into local minima, we should grid the group of rotations and take the elements of the grid as initial values for the steepest descent method.

Bibliography

[1] O. Aichholzer, H. Alt, and G. Rote, *Matching shapes with a reference point.*, Int. J. Comput. Geom. Appl., 7 (1997), pp. 349–363.

[2] P. Alemany, private communication.

[3] M. Pinsky, C. Dryzun, D. Casanova, P. Alemany, and D. Avnir, *Analytical methods for calculating continuous symmetry measures and the chirality measure*, J. Comput. Chem., 29 (2008), pp. 2712–21.

Delay problems in networks

by Aureli Alabert[1] and Xavier Muñoz[2]

2.1 Introduction

The general framework of the problems discussed here is the following: A certain amount of data d is to be moved across a communications network, that we may think is made of a set of nodes connected by edges, from an initial node to a final destination node. The data can be subdivided in small packets that can circulate the network following different paths. Each node admits a waiting line of waiting packets, that are to be processed and resent sequentially.

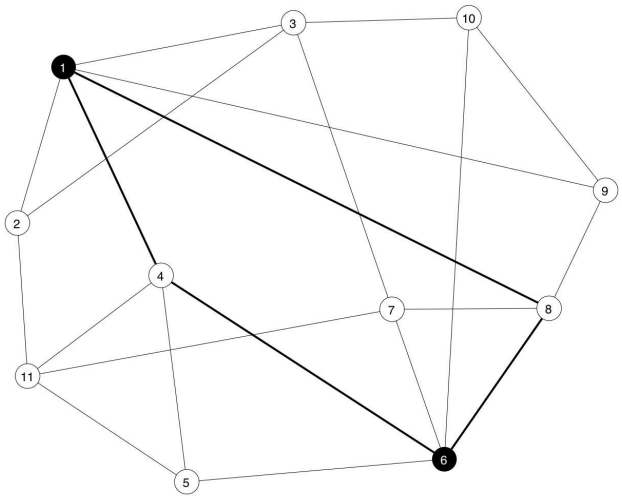


Figure 2.1: An example network.

[1]Departament de Matemàtiques, UAB
[2]Departament de Matemàtica Aplicada 4, UPC

We may thus represent the network as a graph. A simple example is depicted in Figure 2.1. Data has to be transported from node 1 to node 8, and the bold lines are two possible paths for the data packets.

This report is based on discussions held at the Grups d'Estudi *by the authors, with the occasional collaboration of several participants in the event.*

2.2 An *easy* problem

One natural problem to pose in this situation is to minimize the mean delay in the transportation of the whole of the total data, without violating a capacity constraint of each edge (which takes into account the queue capacity of each node).

Denote by E the set of edges of the graph, and by Π the set of all paths from the initial to the final node. The capacity of an edge $e \in E$ will be written as C_e. Denote by t_e the delay through the edge e and by ϕ_p the portion of flow that is transported by the path $p \in \Pi$. The delay can be modelled, if Poisson arrivals can be assumed at a node, by a function

$$x \longmapsto \frac{x}{C - x} \tag{2.1}$$

where C is the capacity.

The minimization of the mean (or total) delay can thus be formulated as

$$\begin{aligned}
\min \quad & \sum_{e \in E} t_e \\
\text{such that} \quad & \sum_{p \in \Pi} \phi_p \geq d, \\
& \phi_e = \sum_{p \ni e} \phi_p \leq C_e \qquad \forall e \in E, \\
& t_e = \frac{\phi_e}{C_e - \phi_e} \qquad \forall e \in E, \\
& \phi_p \geq 0 \qquad \forall p \in \Pi,
\end{aligned}$$

where the restrictions are, respectively, the satisfaction of the demand (the total of the data that has to be transported by the set of paths), the edge capacity constraint, the relation between t_e and C_e given by (2.1), and the positivity of all variables, respectively.

The problem can also be equivalently formulated as

$$\begin{aligned} \min \quad & \sum_{e \in E} t_e \\ \text{such that} \quad & \sum_{p \in \Pi} \phi_p \geq d, \\ & t_e \geq \frac{\phi_e}{C_e - \phi_e} \qquad \forall e \in E, \\ & \phi_p \geq 0 \qquad\qquad \forall p \in \Pi, \end{aligned}$$

since obviously the equalities in the third line of constraints will certainly hold for sure at any optimal point, and the capacity constraints are implied by the denominator $C_e - \phi_e$.

This is an "easy" problem, since it can be tackled by convex programming methods. The problem is even easier if we assume that the values t_e are all constants, and we simply want to find a feasible point that produces a delay below some tolerable threshold τ. We could also introduce some unitary cost k_e of crossing each edge, and thus write the optimisation problem as follows:

$$\begin{aligned} \min \quad & \sum_{e \in E} k_e \sum_{p \ni e} \phi_p \\ \text{such that} \quad & \sum_{p \in \Pi} \phi_p \geq d, \\ & \phi_e = \sum_{p \ni e} \phi_p \leq C_e \qquad \forall e \in E, \\ & \left(\begin{array}{ll} \sum_{e \in p} t_e \leq \tau & \forall p \in \Pi, \end{array} \right) \\ & \phi_p \geq 0 \qquad\qquad \forall p \in \Pi. \end{aligned}$$

The constraints in parentheses are, of course, either always satisfied or make the problem infeasible.

2.3 Duality

A Linear Programming problem can always be represented in the form

$$\begin{aligned} \min \; & c \cdot x \\ \text{such that } & Ax \geq d, \\ & x \geq 0, \end{aligned}$$

where $c, x \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \times m}$, and $d \in \mathbb{R}^m$.

The dual problem is defined as

$$\begin{aligned} &\max d \cdot y \\ \text{such that } &A^T y \leq c, \\ &y \geq 0. \end{aligned}$$

It is always true, and easily verified, that

$$c \cdot x \geq y \cdot A \cdot x \geq y \cdot d \tag{2.2}$$

for all feasible points x and y of the primal and the dual problem, respectively. This implies that every feasible solution of the primal problem bounds from above every solution of the dual. This is useful for solving a linear problem in some variants of the simplex method.

The inequality (2.2) is called *weak duality*. The Strong Duality Theorem in Linear Programming states that there is no gap between the sets of values $c \cdot x$ and $y \cdot d$ when x and y run over the respective feasible sets. This means, in other words, that the optimal values of the primal and the dual problems coincide. Moreover, from the knowledge of one of the optimal points, it is easy to find an optimal point of the other problem. One of the two problems can be faster to solve than the other; hence, this theoretical result has real practical implications. The dual variables have also an interpretation in terms of the original problem (they are the Lagrange multipliers).

The concept of duality can be extended also to Nonlinear Programming problems, but it is not universal. In convex programming, for instance, there is a Strong Duality Theorem, under some condition known as *Slater condition*.

The easy problem of the previous section can be formulated in Positive Semidefinite (PSD) form as:

$$\begin{aligned} \min \quad & \sum_{e \in E} t_e \\ \text{such that} \quad & \sum_{p \in \Pi} \phi_p \geq d, \\ & \begin{pmatrix} t_e + 1 & \sqrt{C_e} \\ \sqrt{C_e} & C_e - \phi_e \end{pmatrix} \succcurlyeq 0 \qquad \forall e \in E, \\ & \phi_p \geq 0 \qquad\qquad\qquad\qquad \forall p \in \Pi. \end{aligned}$$

The Slater condition translates in our formulation as the existence of a path joining the initial and the destination node with a nonzero capacity

slack. In that case, the problem is *approximable*. Roughly speaking, this means that, while we do not know how to obtain the optimum in polynomial time, it is possible to get solutions as close as desired to the optimum in an amount of time that is a polynomial function of the input size. Specifically, one can obtain a solution whose value is within a distance of ε from the optimal solution in a number of steps which is polynomial in the number of vertices $|V|$, the number of edges $|E|$, and ε^{-1}.

2.4 A problem of *medium* difficulty

If in the last problem of Section 2.2 we do not assume that t_e are constants, then the problem is more difficult. The essential difference is not the objective function, but the effective presence of the maximum delay constraints, which apply to all possible paths. Hence, the problem is formally the same as before, but t_e are now considered as variables:

$$\begin{aligned}
\min \quad & \sum_{e\in E} k_e \sum_{p\ni e} \phi_p \\
\text{such that} \quad & \sum_{p\in\Pi} \phi_p \geq d, \\
& \sum_{e\in p} t_e \leq \tau && \forall p \in \Pi, \\
& \begin{pmatrix} t_e + 1 & \sqrt{C_e} \\ \sqrt{C_e} & C_e - \phi_e \end{pmatrix} \succcurlyeq 0 && \forall e \in E, \\
& \phi_p \geq 0 && \forall p \in \Pi.
\end{aligned}$$

This problem is know to be *NP-hard*. This means that, if a polynomial algorithm is found for it, then all other NP-hard problems will be soluble in polynomial time, and in fact all decision problems in the complexity class NP (the problems whose solution can be *verified* in polynomial time) would belong to the class P (the problems whose solution can be effectively *found* in polynomial time). Therefore, this problem is at least as hard as any NP decision problem.

2.5 A *hard* problem

Notice that the last problem above is in fact overconstrained: We are imposing constraints which do not correspond to any real restriction. Indeed, if for

some path p, at one particular feasible solution, one has $\phi_p = 0$, we should not impose the restriction

$$\sum_{e \in p} t_e \leq \tau.$$

For example, take $\tau = 15$ and $d = 13$, with the network of Figure 2.2.

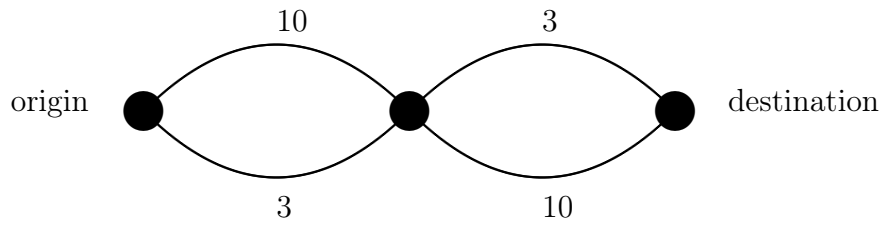


Figure 2.2: An infeasible solution due to an overconstrained model.

In the way the optimisation problem is now formulated, the delays annotated in the edges of the figure are forbidden, since there is a path (solid lines in Figure 2.3) whose total delay of 20 units exceeds τ. However, it is clear that it must represent a feasible solution, since sending 13 units through the upper path and 13 units through the lower path, with a total delay of precisely 13 units, the bound τ is not violated.

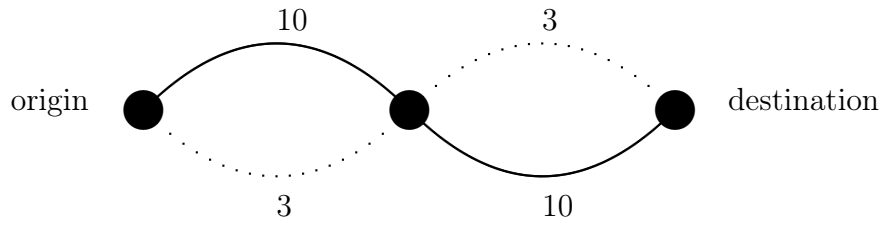


Figure 2.3: In solid lines, the path violating the spurious restriction.

We have several formal options to get rid of the spurious restrictions: If we put

$$\sum_{e \in p} t_e \leq \tau \qquad \forall p \text{ such that } \phi_p > 0,$$

then, apparently, the set of restrictions depends on the specific feasible point under consideration. But we can rewrite it as

$$\Big(\sum_{e\in p} t_e\Big)\cdot u(\phi_p) \le \tau \qquad \forall p \in \Pi,$$

where u is the indicator function of $\{x:\ x>0\}$. We can finally establish a more comfortable formulation if we make use of binary variables:

$$\sum_{e\in p} t_e \le \tau \qquad \forall p \text{ such that } \phi_p > 0$$

is equivalent to

$$\phi_p = 0 \quad \vee \quad \sum_{e\in p} t_e \le \tau \qquad \forall p \in \Pi.$$

We can now enlarge the dimension of the space of variables: For every p, we take $(\phi_p, y_{p,1}, y_{p_2})$ with

$$y_{p,1}, y_{p,2} \in \{0,1\},$$
$$y_{p,1} + y_{p,2} \ge 1.$$

The variable $y_{p,1}$ takes value 1 if $\phi_p = 0$, and the variable $y_{p,2}$ takes value 1 when $\sum_{e\in p} t_e \le \tau$. The requirement $y_{p,1} + y_{p,2} \ge 1$ ensures that one of the alternatives in the disjunction is fulfilled.

This problem is "hard" in the sense that it is not known to which computational complexity class belongs.

Voltage drop in on-chip power distribution networks

by Maria Aguareles,[1] Jordi Blasco,[2] Marta Pellicer[1] and Joan Solà-Morales[2]

3.1 Introduction

The complexity of the different digital circuits with different functions that are becoming integrated in current chips is growing every day. These different components can be found in the market as closed objects, which would be too expensive to re-design in each individual case. These consume electric power at significant rates, which are also quite variable in time. These time variations in some parts of the chip induce undesirable fluctuations in the power supply to the other parts, producing delays and malfunctions in the logic gates. For these reasons, control of maximum voltage drops in the different parts of the chip is one of the most important issues in its design.

Mathematically speaking, we identify the chip with a rectangle $Q = [0, a] \times [0, b]$, and to calculate the voltage drop $u(x, y)$ on it one has to solve the Poisson-like partial differential equation

$$\nabla \cdot \left(\frac{1}{R(x, y)} \nabla u \right) = J(x, y), \tag{3.1}$$

where the sheet resistance function $R(x, y)$ and the current density $J(x, y)$ are known functions. These two functions are usually approximated by functions that are constant across the components of the chip, which are usually of a rectangular form and are usually known as *blocks*. The equation (3.1) has to be solved together with some boundary conditions, which will depend on the type of chip.

[1]Departament d'Informàtica i Matemàtica Aplicada, UdG
[2]Departament de Matemàtica Aplicada 1, UPC

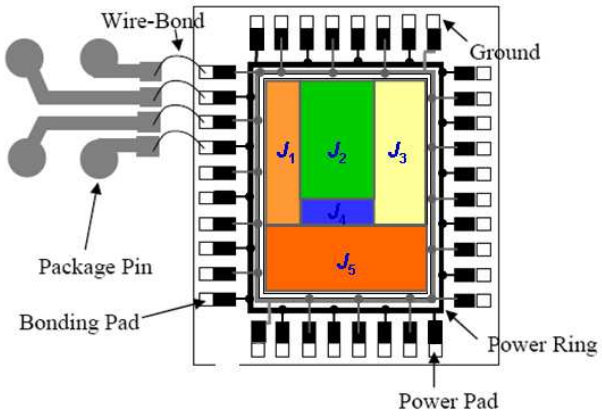


Figure 3.1: Sketch of a wire-bond chip.

In wire-bond chips, the connection to a zero potential conductor is done at the boundaries of Q (see Figure 3.1), so the equation (3.1) has to be solved in Q together with the boundary condition

$$u(x, y) = 0 \quad \text{on the boundary of } Q. \tag{3.2}$$

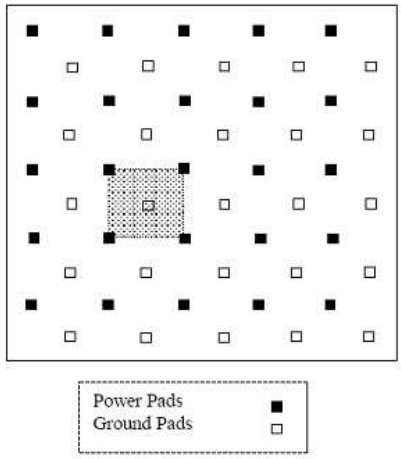


Figure 3.2: Sketch of a flip-chip.

In the flip-chip case, the connection to ground is done through a net of *pads* that are represented by the white small squares in Figure 3.2. In that

case, the boundaries of Q appear as insulating walls, in the sense that there is no external voltage gradient at these points and hence no current flowing through the boundary. If we call P the set of all these pads, we have to solve

$$\begin{cases} \nabla \cdot \left(\dfrac{1}{R(x,y)} \nabla u \right) = J(x,y) & \text{in } Q \setminus P, \\ \dfrac{\partial u}{\partial \mathbf{n}} = 0 & \text{on } \partial Q, \\ u = 0 & \text{on } \partial P. \end{cases} \tag{3.3}$$

The problems (3.1), (3.2) or (3.3) can be quite easily solved with numerical or semi-numerical methods, and we will review these methods later on. However, the designer of chip circuits needs to have online clues about the location of the dangerous zones of large voltage drops before completing the full design. And when these dangerous situations appear, the designer needs to have procedures to decrease these undesirable effects along the way. Completing the full design, realizing that the specifications are not fulfilled and restarting the design again would be a waste of time.

The problem that we have discussed, then, is to find all possible tools that could contribute to the design of a software for *online* support of the design process of these kinds of chips in future work.

This report is based on contributions from the discussions held at the Grups d'Estudi*, which included the following members: Maria Aguareles, Jordi Blasco, Fernando Martínez, Marta Pellicer, Antonio Rodríguez-Ferran and Joan Solà-Morales.*

3.2 Fourier series in the wire-bond chip

Double sine Fourier series are very well adapted to the solution of Poisson equations in rectangles with zero boundary conditions (see [1], for example). Let us suppose for the moment that the function $R(x,y)$ in (3.1) is constant, say $R(x,y) \equiv R_0 > 0$. Then one has to solve

$$\begin{cases} \nabla^2 u = R_0 J(x,y) & \text{in } Q = [0,a] \times [0,b], \\ u = 0 & \text{on } \partial Q. \end{cases} \tag{3.4}$$

This problem has a straightforward solution:

$$u(x,y) = \sum_{m,n \geq 1} E_{m,n} \sin\left(\frac{m\pi}{a} x\right) \sin\left(\frac{n\pi}{b} y\right), \tag{3.5}$$

where

$$E_{m,n} = \frac{4R_0}{\left(\frac{m^2}{a^2} + \frac{n^2}{b^2}\right)\pi^2 ab} \int_0^a \int_0^b J(x,y) \sin\left(\frac{m\pi}{a}x\right) \sin\left(\frac{n\pi}{b}y\right)\, dxdy.$$

Now let us suppose that the function $J(x,y)$ is piecewise constant, with values $J_1, J_2, \ldots, J_p$ on subrectangles $Q_1, Q_2, \ldots, Q_p$ that cover Q (and we still keep $R(x,y) \equiv R_0$ everywhere on Q). Then an observation that could be useful is to note that we can write the solution of (3.4) as

$$u(x,y) = \sum_{k=1}^{p} u^k(x,y),$$

where each u^k solves (3.4) with a right-hand side that is $J_k\varphi^k(x,y)$, where $\varphi^k(x,y)$ is constantly equal to 1 in Q_k and equal to zero on $Q \setminus Q_k$.

One first property of this decomposition process is that one can calculate the functions $u^k(x,y)$ successively, and this means that, along a design process, every time a new component is added the designer can get an idea of the effect of the new component into the previously designed parts. We believe that this property is quite relevant and it also holds, with the natural variations, in the case of variable $R(x,y)$ and also for the flip-chip configurations.

But a second property of this decomposition in the present case is that the Fourier series expression of the partial solutions $u^k(x,y)$ can be explicitly given (see Figure 3.3). If Q_k is the rectangle of dimensions $a_k \times b_k$ centered at the point (x_k, y_k), then the solution $u^k(x,y)$ is given by (3.5), where the values of the $E_{m,n}$ are explicitly

$$E^k_{m,n} = \frac{16R_0J_k}{\left(\frac{m^2}{a^2} + \frac{n^2}{b^2}\right)\pi^4 mn} \sin\left(\frac{m\pi}{a}x_k\right) \sin\left(\frac{a_k m\pi}{2a}\right) \sin\left(\frac{n\pi}{b}y_k\right) \sin\left(\frac{b_k n\pi}{2b}\right).$$

The double Fourier series method can only be applied to (3.1) and (3.2) when the coefficient function $R(x,y)$ is constant. Nevertheless, we present an idea, that should perhaps be further developed in a future work, for the use of the Fourier series when the function $R(x,y)$ is no longer constant throughout the whole chip, but takes different constant values R_k in each Q_k block, which is indeed a much more realistic situation. In that case we could start by making an estimate of a reasonable possible average value R_0 and proceeding as before, but now calculating a correction term $v(x,y)$ that will be small if the sheet resistances at each block are similar, that is, where $\varepsilon_k = R_k - R_0$ are small.

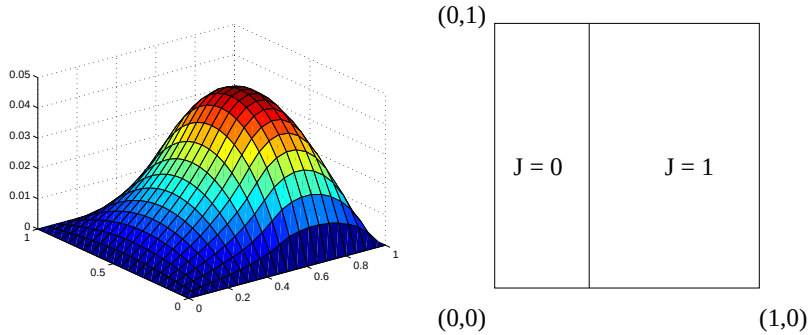


Figure 3.3: Solution to equation (3.4) with two blocks: the left one with $J_1 = 0$ and the second one with $J_1 = 1$.

We first look for an approximation to this correction term of the form $v(x, y) = \sum_{k=1}^{n} \varepsilon_k v^k(x, y)$. If we plug $u + v$ into (3.1) and (3.2), use that u satisfies (3.4), and drop the quadratic terms in the ε_k, we obtain the following equation for v^k:

$$\begin{cases} \nabla^2 v^k = \dfrac{1}{R_0} \nabla \cdot (\varphi^k \nabla u) & \text{in } Q = [0, a] \times [0, b], \\ v^k = 0 & \text{on } \partial Q, \end{cases} \tag{3.6}$$

and this problem can also be solved with double Fourier sine series, as in (3.4) and (3.5).

As we have already pointed out, the double Fourier sine series can only be used if the Laplacian has constant coefficients. However, there is a second way to solve the problem with variable coefficients that also converts the non-constant coefficients problem into infinitely further problems with constant coefficients. The method consists in writing the equation (3.1) in terms of a fixed point equation to be solved by iteration:

$$\nabla \cdot \left(\frac{1}{R_0} \nabla u^{(\ell+1)} \right) + \nabla \cdot \left(\left(\frac{1}{R(x, y)} - \frac{1}{R_0} \right) \nabla u^{(\ell)} \right) = J(x, y),$$

and if $R(x, y)$ is sufficiently close to R_0 it will converge to the actual solution. Each step of this iteration can be done with the method of the Fourier series. Observe that if one starts with $u^{(0)} \equiv 0$, then $u^{(1)}$ will be an approximate solution of the same form as the one in (3.4).

3.3 Asymptotic analysis in the flip-chip

We now tackle the problem of determining the voltage drop in a flip-chip package. This problem was successfully studied by Shakery and Meindl in [5] for the case where both R and J are constant. They considered the two cases represented in Figure 3.4, where the pads are at a distance a from each other and they are either circles of radius ε or squares of sides 2ε. Using the periodicity and symmetry properties of the geometry, one may focus on solving the problems that are represented in Figure 3.5, where one has taken the center of the pad as the origin of coordinates, and then the maximum voltage drop is given by the value $u(a/2, a/2)$.

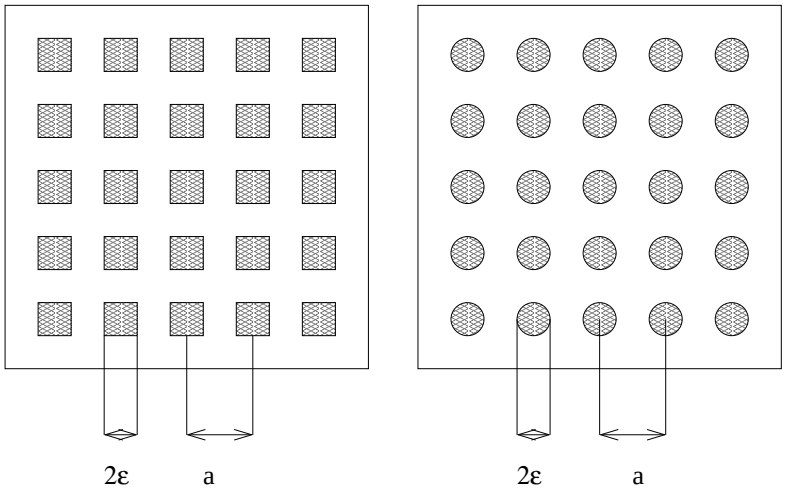


Figure 3.4: Flip-chip with squared or circle pads.

The formula obtained in [5, formula (31)] for the approximate value of $u(a/2, a/2)$ is

$$\frac{RJa^2}{2\pi} \ln \varepsilon - \frac{RJa^2}{2\pi} \ln(0.387a) \tag{3.7}$$

when the pads are circular, and

$$\frac{RJa^2}{2\pi} \ln \left(\frac{0.5903}{0.5} \varepsilon \right) - \frac{RJa^2}{2\pi} \ln(0.387a) \tag{3.8}$$

if the pads are squares. In fact, these formulae appear in [5] with changed sign, because they compute the voltage drop, and not just the voltage at that point.

Although this formula is only an approximation, it gives good results when compared with other calculations. We have also checked this in our

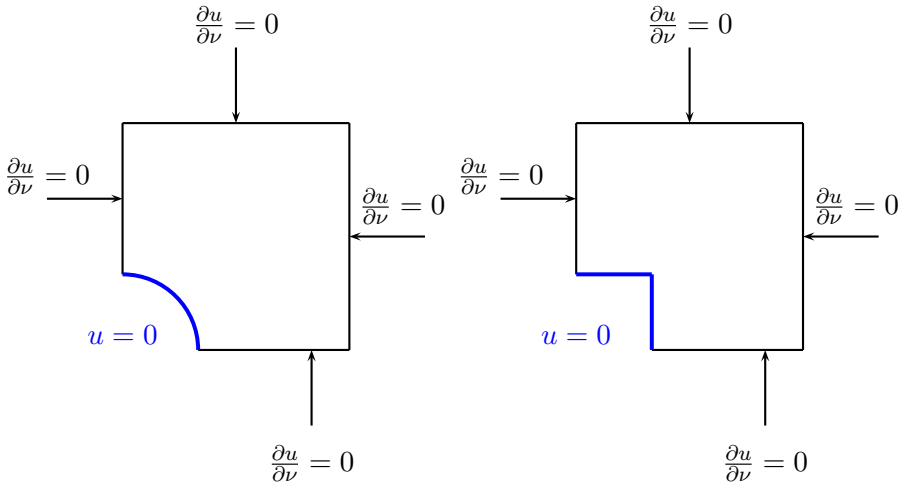


Figure 3.5: Boundary value problems.

finite element simulations, which we present in Section 3.5 below. The formulas are obtained in [5] with ad hoc arguments and reasonable estimates, but we want to point out that these formulas can be obtained systematically with the use of asymptotic analysis methods. We present this approach in the rest of this section.

If we write formula (3.7) when $a = 1$ and $RJ = 1$, we simply obtain

$$u(0.5, 0.5) \simeq \frac{1}{2\pi} \ln \varepsilon + 0.151. \tag{3.9}$$

One can imagine that this is just the beginning of an asymptotic formula in powers of ε like

$$u(0.5, 0.5) \simeq c_{-1} \ln \varepsilon + c_0 + c_1 \varepsilon + c_2 \varepsilon^2 + \cdots . \tag{3.10}$$

We show how the coefficients c_{-1} and c_0 of this formula can be obtained systematically, and will postpone the possibility of calculating the higher order values for future work. These coefficients will also be calculated using finite elements in Section 3.5 below.

In asymptotic analysis ([3] for example), one starts by distinguishing the *inner* and *outer* regions. The outer region will simply be the whole square $[0, 0.5] \times [0, 0.5]$ where we should solve the equation $\nabla^2 u^{\text{out}} = 1$ with the boundary condition $\partial u^{\text{out}} / \partial \mathbf{n} = 0$. This problem is clearly incompatible, and this is because by taking $\varepsilon = 0$ we have forgotten that the wall $\sqrt{x^2 + y^2} = \varepsilon$ acts like a sink that compensates the source term. So the correct outer

equation is $\nabla^2 u^{\text{out}} = 1 - \delta$ with the boundary condition $\partial u^{\text{out}}/\partial \mathbf{n} = 0$, where δ is the delta-function centered at $(0,0)$ and its coefficient has been chosen to make the system compatible.

It is perhaps easier to solve this problem by extending it to the full square $[0,1]\times[0,1]$ with one delta-function on each corner. Then the right-hand side of the equation will be the function $1-\delta_1-\delta_2-\delta_3-\delta_4$, that one can develop in double cosine series, and obtain

$$\sum_{\ell=1}^{\infty} 2\left(\cos(2\ell\pi x)+\cos(2\ell\pi y)\right) + \sum_{n,m=1}^{\infty} 4\cos(2n\pi x)\cos(2m\pi y).$$

So the solution is

$$u^{\text{out}} = C(\varepsilon) + u^{\text{aux}}(x,y),$$

where

$$u^{\text{aux}}(x,y) = \sum_{\ell=1}^{\infty} \frac{-1}{2\pi^2\ell^2}\left(\cos(2\ell\pi x)+\cos(2\ell\pi y)\right) + \\ + \sum_{n,m=1}^{\infty} \frac{-1}{\pi^2(n^2+m^2)}\cos(2n\pi x)\cos(2m\pi y)$$

and $C(\varepsilon)$ is a constant (depending on ε) that has to be determined.

Upon writing $r = \sqrt{x^2+y^2}$, we have evaluated these series numerically and calculated numerically the limit as $(x,y)\to 0$ of $u^{\text{aux}}(x,y) + (\ln r)/(2\pi)$ (remember that $\nabla^2 - \ln r = -2\pi\delta$) and have obtained the approximate value of -0.208. This means that

$$u^{\text{out}}(x,y) = -\frac{1}{2\pi}\ln r - 0.208 + O(r) + C(\varepsilon).$$

We have also calculated numerically $u^{\text{aux}}(0.5,0.5) \simeq -0.0552$. These numerical calculations have also been used to draw Figure 3.6.

On the other hand, the inner problem is now stated in the scaled variables $x' = x/\varepsilon$, $y' = y/\varepsilon$, and, letting $\varepsilon \to 0$, the region becomes

$$\Omega = \{(x')^2 + (y')^2 > 1,\ x > 0,\ y > 0\}.$$

Performing this change of variables in the equation and allowing $\varepsilon \to 0$, one finally has to solve

$$\begin{cases} (\nabla')^2 u^{\text{inn}} = 0 & \text{in } \Omega, \\ u^{\text{inn}} = 0 & \text{on } (x')^2 + (y')^2 = 1, \\ \dfrac{\partial u^{\text{inn}}}{\partial \mathbf{n}} = 0 & \text{on } x' = 0 \text{ and } y' = 0. \end{cases} \tag{3.11}$$

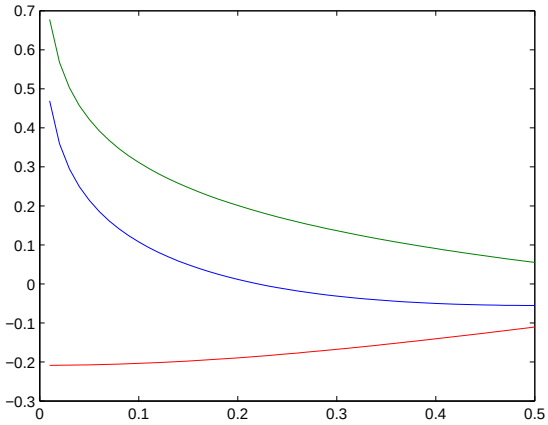


Figure 3.6: From top to bottom: $y = -\ln(\sqrt{2x^2})/(2\pi)$, $y = u^{\mathrm{aux}}(x,x)$, and $y = u^{\mathrm{aux}}(x,x) + \ln\sqrt{2x^2}/(2\pi)$.

The solution is

$$u^{\mathrm{inn}} = C_1 \ln r'.$$

Observe that C_1 does not depend on ε.

If we now write u^{out} also in the scaled variables, we see that one has to match $u^{\mathrm{out}} = -\frac{1}{2\pi}\ln(\varepsilon r') - 0.208 + C(\varepsilon) + O(\varepsilon)$ with $C_1 \ln r'$, and we obviously obtain that $C_1 = -1/(2\pi)$ and $C(\varepsilon) = \frac{1}{2\pi}\ln\varepsilon + 0.208 + O(\varepsilon)$. Finally,

$$u(0.5, 0.5) \simeq -0.0552 + C(\varepsilon) = \frac{1}{2\pi}\ln\varepsilon + 0.153, \tag{3.12}$$

which agrees with the formula (3.9) obtained in [5] up to the second significant digit.

It remains to be explained how we would proceed in the case of squared pads, represented in the left-hand picture of Figure 3.5. The outer solution is obtained as before and it is exactly the same solution. The difference lies in the outer solution, that has to be found in the domain

$$\Omega = \{x > 0,\ y > 0,\ \max(x, y) > 1\}.$$

This problem is more difficult than the previous one, but still solvable. With a Schwarz–Christoffel conformal map, one can transform the domain Ω into the half plane $y > 0$, and then, with (the inverse of) the Joukowsky map, the domain becomes the exterior of the unit disc, where the solution of (3.11) is simply $u^{\mathrm{inn}} = C_1 \ln r'$ again (see Figure 3.7 below).

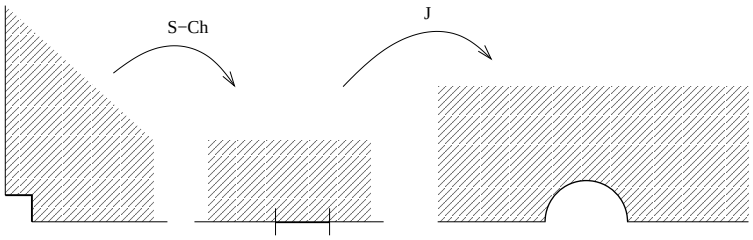


Figure 3.7: Two conformal transformations.

3.4 An averaged model

This section describes only projects that could be undertaken in the direction of building an averaged model. These projects are not very highly elaborated yet, and need further development.

From the mathematical point of view, the role of the pads in the equation of a flip-chip is to act as sinks to compensate the source terms of the main equation (3.1). Instead of solving boundary-value problems where the boundaries of all the individual pads would have to be taken into account, we could also look for an averaged homogenized or continuous model, where the role of the pads would be played by a new term in the equation (3.1), that would have to be solved in the whole chip, and not only outside the pads.

Moreover, we have so far considered only regular distributions of pads, but we could also consider less regular or even very disordered distributions, and there the use of the averaged equation would be even more useful. The same averaging principle could be applied to the sub-rectangles where the functions R and J are constant. If there are a large number of them, one can think that a kind of average would be relevant enough.

We are always thinking in approximating equations to help the designer reach an idea of how far his or her design is from a dangerous situation in which the specifications may not be fulfilled. Of course, once the chip is fully designed, it is reasonable and necessary to perform a full simulation involving all the pads and all the sub-rectangles, with all the details. But this type of simulation is perhaps not very reasonable for carrying it out *online* in the design process.

In order to simplify the problem when there are many pads and many rectangular regions where the function $J(x, y)$ is constant, it is a good idea to forget the existence of the ground pads and the zero boundary condition

on them, and try to represent their effects through a new dissipation term $-Ku$ in the equation, that could become

$$\nabla \cdot \left(\frac{1}{R(x,y)} \nabla u \right) - Ku = J(x,y), \tag{3.13}$$

where K can either be a constant or a space-dependent coefficient.

This kind of process of averaging has quite a large tradition in the mathematical literature, under the name of *homogenization in PDE's*. For a recent publication with some aspects of this approach one can see the monograph [4].

In that approach, a bulk material represented by a set Ω and some *absorbing inclusions* $F_\varepsilon \subset \Omega$ is considered. This is the same as our case, which is different from a situation where there are Neumann boundary conditions at the interior inclusions, the so-called *reflecting inclusions*. Nevertheless, it has to be remarked that when $\varepsilon \to 0$ it is supposed that not only the diameters of the inclusion components tend to zero, but also the distances among these components tend to zero in the correct proportion. In the notation suggested by Figure 3.4, one would have something along the lines of $a = h\varepsilon$, for a fixed h, and then $\varepsilon \to 0$.

Another possibility would be to change the right-hand side of (3.1) instead of the left-hand side. This has more or less been the way to deal with the outer solution in the preceding section. In the calculation of the outer solution we have substituted the effect of the (small) pads by the effect of delta functions. That could be another option to explore. It should also be noted that with these outer solution we have not calculated more than the first two terms in the series expression (3.10). The way in which the subsequent terms are calculated would also give more insight in the development of a more refined averaged model.

3.5 Simulations with finite elements

The type of problems that we have dealt with so far are easy to tackle using some standard numerical methods such as Finite Element Methods (FEM). Indeed, one would start by computing some solutions to both the flip-chip package problem or the wire-bond one to get some insight on the type of solutions to be expected and even to compute an excellent numerical approximation of the maximum IR-drop in the chip. However, as it was already exposed in the Introduction, the idea is to provide the designer with formulae such that with a simple scientific calculator he or she could obtain an approximation to the maximum IR-drop with no need of solving any differential equation by any means. In this sense, the sort of expressions given

in (3.9)–(3.10) are the type of formulae that we are looking for. There are analytic asymptotic method techniques that give the right values of the a-priori unknown coefficients $c_{-1}, c_0, c_1, \ldots$ Nonetheless, and given that these equations are easy to solve numerically, another possibility is to numerically obtain the maximum IR-drop and afterwards match the obtained value with the formula given in (3.10) to obtain the coefficients.

In this section, we are going to use the FEM method to solve the equations and we are going to show how to obtain the coefficients in (3.10) afterwards. Let us focus on the elementary problems stated in Figure 3.5. In particular, let us consider the following problem:

$$\begin{cases} \Delta u = 1 & \text{in } \Omega, \\ \dfrac{\partial u}{\partial \mathbf{n}} = 0 & \text{on } \partial\Omega_1, \\ u = 0 & \text{on } \partial\Omega_2. \end{cases} \tag{3.14}$$

Here we are taking $R(x,y)\,J(x,y) \equiv 1$ in the whole domain. We are considering $\Omega = \Omega_Q \setminus \Omega_\varepsilon$, where $\Omega_Q = (0,\, 0.5) \times (0,\, 0.5)$ and Ω_ε stands for the pad domain. We distinguish two cases here: $\Omega_\varepsilon = \{\max(x,y) \leq \varepsilon\}$ in the case of a square pad of length ε, and $\Omega_\varepsilon = \{x^2 + y^2 \leq \varepsilon^2\}$ in the case of a circular pad of radius ε. The values of ε that we have considered are $\varepsilon = 0.05, 0.1, 0.15, 0.20$. Concerning the boundaries, $\partial\Omega_2$ stands for $\partial\Omega_\varepsilon$ and $\partial\Omega_1 = \partial\Omega \setminus \partial\Omega_2$.

This type of problems can be solved numerically using, for instance, the `pdetool` toolbox from `Matlab`. This is a FEM software which has some standard PDE's implemented that can be numerically solved for different geometries in $\mathbb{R}^2$. We need to follow these steps:

1. Define the geometry of the problem.
2. Define the boundary conditions, that can be different depending on the part of the boundary.
3. Choose the PDE between the ones that are already implemented and give the values of the parameters involved.
4. Initialize the mesh and refine it if necessary.
5. Solve the PDE problem.

In Figure 3.8 we present some screenshots of these previous steps in `Matlab`. Once we have done this, we obtain the numerical solution for the PDE on each node of the mesh. The software also allows us to plot this solution and its contour levels (see Figure 3.9).

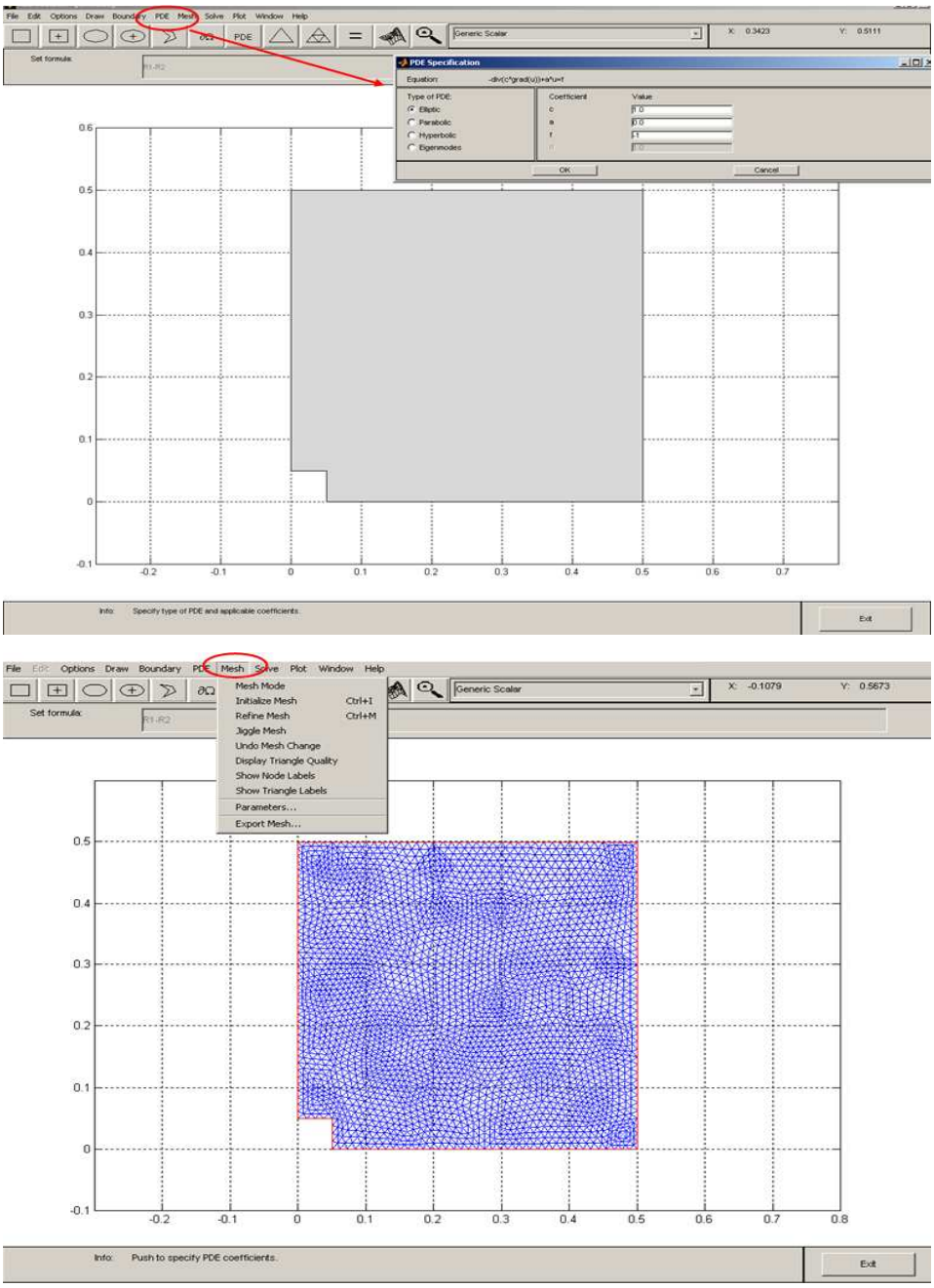


Figure 3.8: Two screenshots showing the aspect of the simulation with `Matlab`. From top to bottom, PDE and mesh definition for a problem with a square pad.

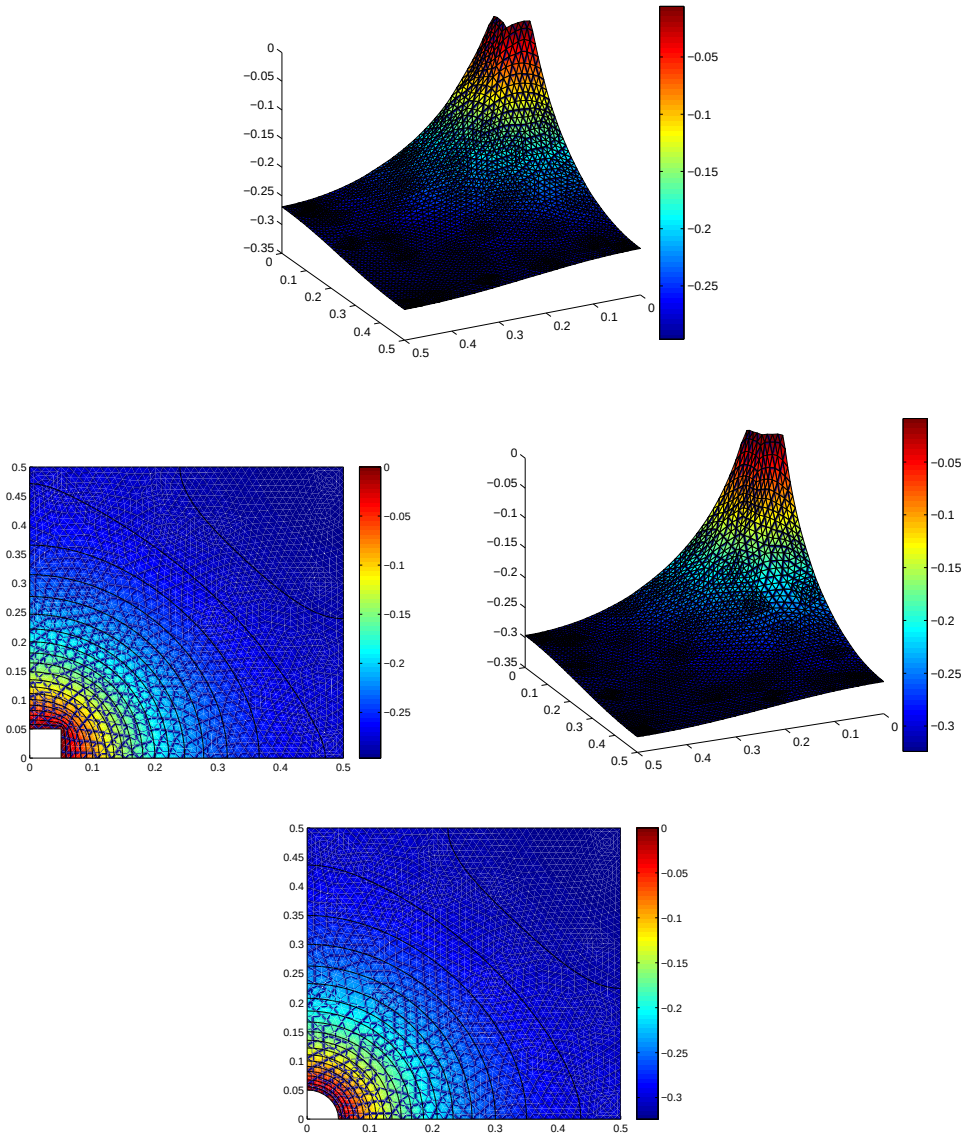


Figure 3.9: Numerical solutions, including contour plot, for a square and a circular pad when $\varepsilon = 0.05$.

It is also interesting to note that, once achieved these numerical values at the nodes of the mesh, one can manipulate them in order, for instance, to calculate the maximum of the absolute value of the solution, which in this case is attained at $u(0.5, 0.5)$ (see Section 3.3). This, together with the asymptotic expansion obtained in (3.10), may be used to compute approximate values for the coefficients c_0, c_1 and c_2 (according to Section 3.3, we will take $c_{-1} = 1/2\pi$, as $R, J, a = 1$ in this problem). To do so, one has to use the numerically obtained values of $u(0.5, 0.5)$ for the different values of ε and find the set of coefficients in (3.10) that best fit them. For instance, in this case, the following system must be solved:

$$\begin{cases} -0.323992 = \frac{1}{2\pi}\ln 0.05 + c_0 + c_1(0.05) + c_2(0.05)^2 \\ -0.215547 = \frac{1}{2\pi}\ln 0.1 + c_0 + c_1(0.1) + c_2(0.1)^2 \\ -0.154141 = \frac{1}{2\pi}\ln 0.15 + c_0 + c_1(0.15) + c_2(0.15)^2 \\ -0.112725 = \frac{1}{2\pi}\ln 0.2 + c_0 + c_1(0.2) + c_2(0.2)^2 \end{cases} \tag{3.15}$$

for a circular pad, and

$$\begin{cases} -0.297656 = \frac{1}{2\pi}\ln 0.05 + c_0 + c_1(0.05) + c_2(0.05)^2 \\ -0.190042 = \frac{1}{2\pi}\ln 0.1 + c_0 + c_1(0.1) + c_2(0.1)^2 \\ -0.129616 = \frac{1}{2\pi}\ln 0.15 + c_0 + c_1(0.15) + c_2(0.15)^2 \\ -0.089079 = \frac{1}{2\pi}\ln 0.2 + c_0 + c_1(0.2) + c_2(0.2)^2 \end{cases} \tag{3.16}$$

for a square pad. As these systems are overdetermined, the idea is to find the coefficients c_0, c_1 and c_2 that are *closer* to the exact solution upon using a linear least squares method (see for instance [2]). Hence, one has to solve the corresponding normal equations, which in this case yield

$$c_0 = 0.153420,\ c_1 = -0.00005163,\ c_2 = -0.249654, \tag{3.17}$$

for a circular pad, and

$$c_0 = 0.180623,\ c_1 = -0.0168976,\ c_2 = -0.254666, \tag{3.18}$$

for a square pad.

These results are in strong agreement with those of [5] included in Section 3.3. For circular pads, formula (3.7) is written for $a = 1$ and $RJ = 1$

in (3.9). For square pads with $a = 1$ and $RJ = 1$, formula (3.8) becomes

$$u(0.5, 0.5) \simeq \frac{1}{2\pi} \ln \varepsilon + 0.178.$$

However, the good point in our calculations in comparison to the method used in [5] is that the method we have used would hold to obtain higher order terms in the expansion, and thus obtain a more precise solution. This is especially interesting since, in real chips, the pads are not that small in comparison with the size of the chip, and hence the value of ε is actually not so small. Thus, the only chance to obtain greater accuracy is to take higher order terms in the asymptotic expansion presented in Section 3.3.

Bibliography

[1] R. Courant, D. Hilbert, *Methods of Mathematical Physics*, New York: John Wiley & Sons, 1989.

[2] R. J. Hanson, *Solving Least Squares Problems*, Upper Saddle River: Prentice-Hall, 1974.

[3] E. J. Hinch, *Perturbation Methods*, Cambridge: Cambridge University Press, 1991.

[4] V. A. Marchenko, E. Y. Khruslov, *Homogenization of partial differential equations*, Translated from the 2005 Russian original, Progress in Mathematical Physics, 46, Birkhäuser Boston, Inc., Boston, MA, 2006.

[5] K. Shakeri, J. D. Meindl, Compact physical IR-drop models for chip/package co-design of gigascale integration (GSI), *IEEE Transactions on Electron Devices* **52 (6)** (2005), 1087–1096.

Contents