Reduced-Order Equivalent-Circuit Models Of Thermal Systems Including Thermal Radiation

Serap Karagol

North Carolina Agricultural and Technical State University

REDUCED-ORDER EQUIVALENT-CIRCUIT MODELS OF THERMAL SYSTEMS INCLUDING THERMAL RADIATION

by

Serap Karagol

A dissertation submitted to the graduate faculty
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY

Department: Electrical Engineering
Major: Electrical Engineering
Major Professor: Dr. Marwan Bikdash

North Carolina A&T State University
Greensboro, North Carolina
2011

ABSTRACT

Karagol, Serap. REDUCED–ORDER EQUIVALENT-CIRCUIT MODELS OF THERMAL SYSTEMS INCLUDING THERMAL RADIATION (**Major Advisor: Marwan Bikdash**), North Carolina Agricultural and Technical State University.

We established a general, automatic, and versatile procedure to derive an equivalent circuit for a thermal system using temperature data obtained from FE simulations. The EC topology was deduced from the FE mesh using a robust and general graph-partitioning algorithm. The method was shown to yield models that are independent of the boundary conditions for complicated 3D thermal systems such as an electronic chip. The results are strongly correlated with the geometry, and the EC can be extended to yield variable medium-order models. Moreover, a variety of heat sources and boundary conditions can be accommodated, and the EC models are inherently modular. A reliable method to compute thermal resistors connecting different regions was developed. It appropriately averages several estimates of a thermal resistance where each estimate is obtained using data obtained under different boundary or heating conditions. The concept of fictitious heat sources was used to increase the number of simulation datasets.

The method was shown to yield models that are independent of the BCs for complicated 2-D thermal systems such as a 2D cavity. A reliable method to compute thermal resistors connecting different regions was developed. In general, the number of regions required for getting an accurate reduced-order model depends on the complexity of the system to be modeled. We have extended the reduced-order modeling procedure to

include a view-factor based thermal radiation heat transfer model by including voltage-controlled current sources in the equivalent circuit

School of Graduate Studies
North Carolina Agricultural and Technical State University

This is to certify that the Doctoral Dissertation of

Serap Karagol

has met the dissertation requirements of
North Carolina Agricultural and Technical State University

Greensboro, North Carolina
2011

Approved by:

Dr. Marwan Bikdash Major Professor	Dr. Numan S. Dogan Committee Member
Dr. Clinton B. Lee Committee Member	Dr. Abdollah Homaifar Committee Member
Dr. Robert Li Committee Member	Dr. John Kelly Department Chairperson

Dr. Sanjiv Sarin
Dean of Graduate Studies

DEDICATION

This work is dedicated to my father and mother.

BIOGRAPHICAL SKETCH

Serap Karagol was born on May 10, 1979 in Istanbul, Turkey. She received her Bachelor of Science degree in Computer Engineering from Karadeniz Technical University in 2002. She received the Master of Science degree from the Department of Electrical and Computer Engineering at the North Carolina Agricultural and Technical State University in 2006. She is a candidate for a Ph.D. in the Department of Electrical and Computer Engineering at the North Carolina Agricultural and Technical State University.

ACKNOWLEDGEMENTS

When I asked myself who encouraged me to pursue the Ph.D. degree, I always got the same answer — my parents, Mr. Numan Karagol and Mrs. Gul Karagol.

It is a great time to express many thanks to my advisor, Dr. Marwan Bikdash for his support and enthusiastic advice. The past five years studying at North Carolina Agricultural and Technical State University have been some of the happiest times in my life. I also would like to thank Dr. Numan S. Dogan, Dr. Robert Li, Dr. Clinton B. Lee, and Dr. Abdollah Homaifar serving as my committee members and providing valuable advice.

I wish to thank Dr. Prabhat Ranjan Pathak, Dr. Satish Tadiparthi, Meng Vang and Shahina Masood for all their help when I need it.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

Electrothermal analysis is important for several electronics industries such as power electronic devices [1-4], voltage regulators [5], microprocessors [6], motor drives [7], and space communication systems [8]. A major difficulty is that heat generation mechanisms are often tightly coupled with the electrical operation of the circuit. For instance, heat generation depends on electrical parameters that depend in turn in transistor junction temperatures. Hence there is a great interest in transistor junction temperatures. Moreover, detailed thermal models are difficult to use in system design due to the presence of thousands of components. A compact thermal model, if reasonably accurate and easily interpretable, alleviates both the mixed electrothermal simulation problem and the design accessibility problem. A key capability is the ability to predict temperatures at specific points in the geometry, as well as to capture the overall steady-state and transient thermal behavior.

This dissertation contributes to bridging the gap between small-order thermal models and very large and accurate thermal models generated by methods such as the finite-element, the finite-difference or boundary-element methods. These models can be very detailed, accurate, large-scale ($100,000$ equation or more), and computationally expensive. Unfortunately, they often require the use of commercial and expensive software that cannot be easily (or not at all) coupled with other simulation software such as that required for electrical-circuit simulations. In addition to its high expense, this approach may require familiarity with several potentially disparate software products.

Several approaches to generate small-order models have been reported. The thermal resistor network [9-10], especially that consisting of two resistors [11-12], and the DELPHI approach [13] are amongst the most popular instances of such compact thermal models. When the geometry is simple, few resistors are needed and can be determined from test or simulation data [14]. The DELPHI approach [13] contains several thermal resistors that link a junction node to several surface nodes and improves the accuracy for a simple geometry to about 5%. The asymptotic waveform evaluation provides a generalized approach to waveform estimation for RLC circuits with initial conditions [15]. Arnoldi-based techniques [16-17] can generate guaranteed stable and passive reduced-order models but they are not as accurate as a Padé-based model [18] of the same size. Another passive reduced-order model uses projection on block Krylov subspaces [19]. The above methods work best when the thermal behavior is linear.

A geometry-based approach constructs compact models by decomposing the geometry into smaller regions; (e.g. standard shapes such as disks, tubes, etc.) and then developing simplified semi-empirical formulas of the thermal resistance and capacitance of such shapes [20]. Decomposition can be applied to mathematical models that represent any large-scale system whether a chemical plant, electrical network, river basin, bridge structure or the like. One of the distinct characteristics of a large system, as opposed to a subsystem, is that it represents a complex network of interacting elements. In treating such large systems in the physical and social sciences, as well as in engineering, one is concerned with methods of decomposing or breaking up a large problem into a set subproblem of lower dimensionality the union of which is equivalent to the original problem. These subproblems can be treated independently for the purposes of optimization, design, control, etc. Libraries of models relevant to power electronics are often available, for instance, in Saber [21] and MAST [22]. As the level of component integration is increased, these models become less useful as the simplifying assumptions used in deriving them are violated, and because the decomposition into standard simple shapes becomes increasingly unwieldy, and detailed finite-element models become the main hope for sufficiently accurate simulation models. Celo et al. [23] used a parametric model reduction (PMR) technique to create a parameterized reduced thermal model, and subsequently derived a large system of equations. They reported to produce errors from 2% to 10% on few junction temperatures over a large number of boundary conditions (BCs), but usually assuming a reasonably simple geometry [23]. They argued that such performance can be extended to many nodes in the thermal model.

Other parameterized model-order reduction techniques are based on moment matching [24-25], truncated balanced realization [26], and on quasi-convex optimization [27]. Unfortunately, all existing PMR techniques apply only to linear systems [28]. Tsai et al. [29] used ARWE (Adaptive Regionwise Exponential approximation) and Prony's method which are time-domain approximation techniques to obtain from impulse-response-type simulation data some simplified reduced-order models. They claimed a 2x to 5x improvement in accuracy.

Linton and Agonafer [30] presented coarse and detailed CFD modeling of a finned heat sink using Phoenics, a CFD package from CHAM Limited. They concluded that the empirical data agreed well with the detailed CFD heat sink. To reduce the processing time and cost, a coarse-mesh approximation was recommended for a finned heat sink.

Narasimhan and Majdalani [31] also compared detailed and compact models of heat sinks and found close agreement. A thermal model of heat sink in the form of an RC thermal equivalent network was used by Drofenik and Kolar [32]. Their experimental results for two different heat sinks showed temperature errors below 10%. Obinelo [33] presented a generalized method to represent heat sinks at the system level. The purpose was to reduce the detailed flow and thermal behavior of the heat sink to a few parameters that could easily be incorporated into a system level model.

We advocate the use of medium-order models whose design is guided by the available finite-element models and simulation results. These models are further justified by the complexity of the construction of the thermal systems of interest (such as electronic chips) and the sophistication needed in understanding their electrothermal behavior and meeting stringent performance requirements. At this level of complexity, user-performed geometry decomposition becomes unwieldy and unreliable. Hence, we advocate in this dissertation the use and development of medium-order Equivalent Circuit (EC) models that are strongly correlated with the component-level description of the electronic circuit. This approach leads to a natural and reliable method of electro-thermal simulations. Moreover the method intuitively reflects the physical understanding of thermal behavior, and is extendible to nonlinear effects such as radiation. To achieve this goal, one must be able to extract automatically (in a systematic and reliable way) the compact model from large finite-element models. The success of small-order equivalent models in several simpler simulation situations shows that a medium-order equivalent model can be useful for complex geometries.

Radiation is a complicated heat transfer phenomenon. Some electronics packaging engineers choose to ignore radiation heat transfer because it is often insignificant compared with convection and conduction, and because building a radiation model is time consuming [34] and nonlinear. Moreover, conservative designs can be used to side-step the effect of radiation. The biggest challenge in reduced-order modeling is in coupling nonlinear thermal models with nonlinear electrical models. A traditional method of limited applicability is to first linearize the system, then to perform model-order reduction on the linear system. The Ritz vector approach [35] is theoretically suitable for heterogeneous structures and exhibits good accuracy (See [36] for a review). Calculation of view factors plays an important role in estimating the radiation heat transfer rate. View factors can be calculated by direct integration [37-38], the Monte Carlo method [38], crossed-string method [37-38], ray tracing method [39], and stochastic methods [40].

One of the most common methods is ray-tracing. In an arbitrary geometry with obstructions, a check must first be performed to verify that two surfaces are in a direct line of sight prior to calculation of the view-factor. A ray is emitted from the center of surface element i and is directed towards the center of surface element j. If the ray fails to reach surface element j because it intersects another boundary face k first, the view-factor, F_{ij}, particular $i - j$ pair is set to zero, and the view-factor of the $i - k$ pair is instead calculated [41]. In two-dimensional (2D) planar geometry, the view-factors are most efficiently calculated using the cross-string method [42]. Since the coordinates of the vertices of each surface element are known from the mesh generator, the method is rather straightforward.

For arbitrary three-dimensional (3D) geometry, a simple general-purpose method to compute the view-factors is not available. One approach that has been employed to compute view-factors in 3D is the Monte Carlo method which is based on probability and statistics [43]. The other way is further discretizing the differential area into small subelements called pixels and projecting each face onto the pixel grid [44-45]. This method is mainly used in graphics where a high level of detail is necessary to construct a visually realistic scene. In 1977, Jason [46] used a visibility graph built from the corners of the obstacles.

1.1 Synopsis

This dissertation is organized as follows. Chapter 2 presents a general method to derive an equivalent circuit (EC). The principle of the electrical-thermal analogy is introduced. We briefly review the analogy used to estimate resistances, capacitances, voltage and current sources. In Chapter 3, METIS mesh partitioning software is explained. The created interface between METIS and MATLAB is explained in detail. In Chapter 4, three FE models, a thin 3D rectangular slab, a motor pole, and an electronic chip model, are used to validate the proposed algorithm for steady-state and transient analysis for conduction and convection boundary conditions. Several datasets are created to obtain a boundary condition independent EC. The PSPICE interfaces are written for both a steady-state analysis and transient analysis. Chapter 5 presents a view factor formulation and calculation between two surfaces. In Chapter 6, edge-to-edge visibility determination techniques are explained and an edge-to-edge visibility graph algorithm is discussed. An equivalent circuit is computed for a simple geometry. Finally, conclusions are stated in Chapter 7.

CHAPTER 2

GENERAL METHOD TO DERIVE EQUIVALENT CIRCUIT FOR THERMAL SYSTEMS

2.1 Introduction

The modeling procedure adopted for this research is bases on the analogy between across and through variables in different disciplines. Figure 1 shows a diagrammatic representation of the electrical-thermal analogy.

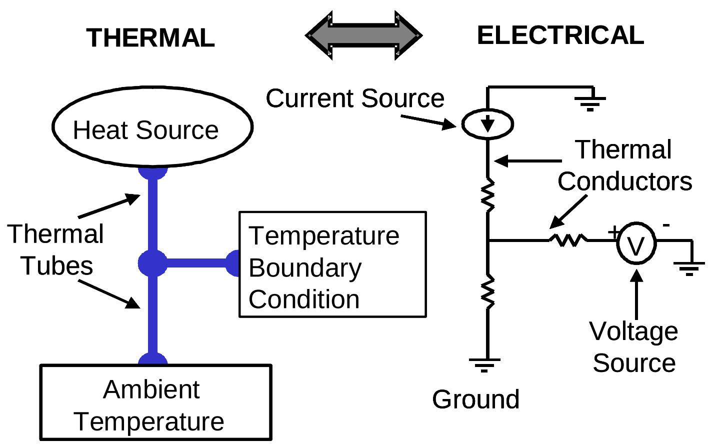


Figure 1. Diagrammatic representation of the electro-thermal analogy

Here the electrical current (through variable) is analogous to the heat flow rate through a specified surface and the voltage drop (across variable) is analogous to a temperature difference. Heat is assumed to flow through "thermal tubes" delineated by virtual adiabatic surfaces. One such thermal tube is then represented by a resistor in the EC, and the current flowing in one resistor in the EC is analogous to the heat flow rate due to conduction or sometimes convection between two adjacent regions. To complete the analogy, a voltage source represents a constant temperature on an isothermal boundary, and a current source represents an injected heat flow rate (See Table 1).

Table 1. A list of analogous quantities between thermal and electrical systems

Electrical		Thermal	
voltage (volts)	V	temperature (^{o}C or ^{o}K)	T
current (amps)	I	heat flow rate (W)	$\dot{Q}$
current density (A/m^2)	J	heat flux rate (W/m^2)	$\dot{q}$
electrical resistance (Ω)	R	thermal resistance (^{o}K/W)	R
electrical capacitance	C	thermal capacitance	C
(farads)		(J/^{o}K)	

2.2 Interpretation Based on Thermal Tubes

The definition of a given virtual thermal tube depends heavily on the set of boundary conditions used, and if those conditions are changed, the supposedly adiabatic surface will no longer remain adiabatic. Hence the virtual thermal tube is not a robust concept. Note however, that a network of such tubes is a more robust concept. In this case, the voltages at the nodes of the EC are best interpreted as representing the temperatures at some actual points in the solid, or at some specified nodes of the FE mesh. For thermal systems where the geometry and the boundary conditions are complex, the geometry cannot be "generated" or "carved" easily or reliably as to generate thermal resistors. For instance, a thermal resistance can be interpreted as a virtual thermal tube connecting two regions as illustrated in Figure 2(a). The two temperatures at the two ends of the tube represent the temperatures of the regions connected through the thermal tube, and the thermal tube can be thought of as being delimited by a "virtual adiabatic surface" consisting of heat flow paths. The postulated thermal tube can be further abstracted, as shown in Figure 2(b), as a virtual thermal tube with an equivalent cross-section and an equivalent thermal conductivity.

Jeong [47] also used the concept of flux tube to represent a magnetic equivalent circuit method. Each element of a squirrel cage induction motor was represented by a flux tube. Davoudi and Chapman [48] created a detailed equivalent model of linear magnetic devices using flux tubes.

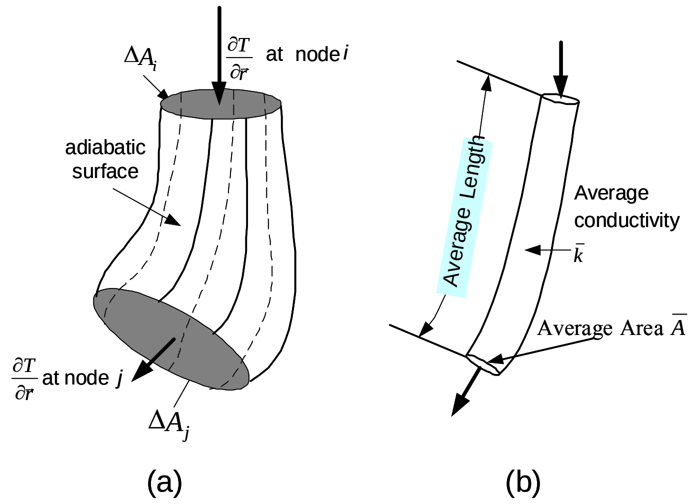


Figure 2. The concept of thermal tube

2.3 Interpretation Based on Regions

The topology of the EC, which we will adopt here, is based on decomposing a given contiguous homogeneous material domain into contiguous reasonably small regions, and representing every region by a node in the EC. In Figure 3(a), there are 7 regions R_1-R_7 shown, and each will be represented by a node in the EC whose voltage will represent the average temperature in the corresponding region.

A constant-temperature T_B boundary condition acts on a surface S_B of a region either directly or through a fluid-solid heat transfer coefficient. If T_B represents the ambient temperature reservoir the heat flow, this boundary will be represented by thermal resistor that is grounded. If the boundary is at temperature that is different that the ambient (or reference) temperature, a voltage source will be needed whose value represents the deviation from the reference temperature. The heat flow rate from region R_1 to region R_2 is denoted Φ_{12} through the surface S_{12}. S_1^{1B}, S_1^{2B}, S_1^{5B}, and S_2^{5B} are the surfaces with radiation boundary condition. Subscripts of the surface refer to surfaces and superscripts refer to regions.

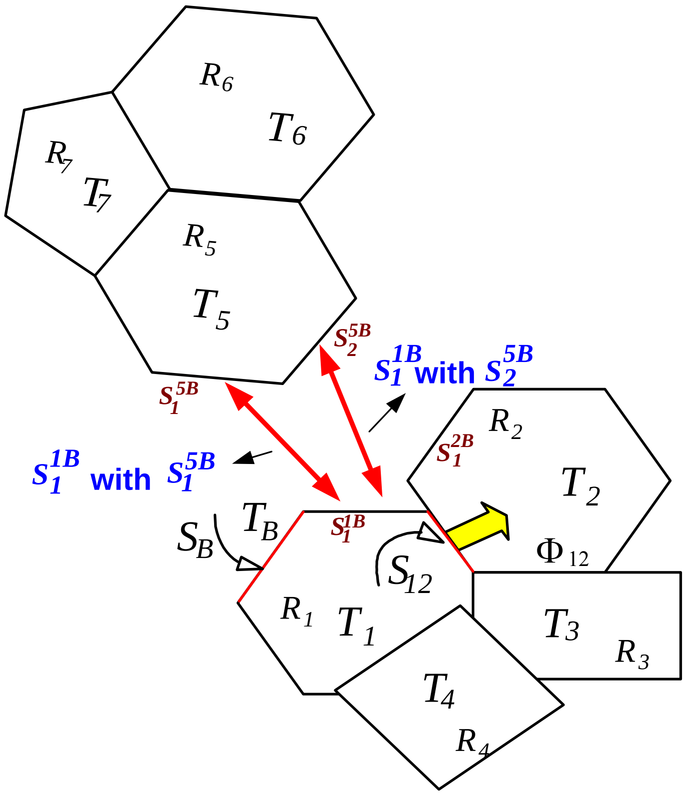
The part of the equivalent circuit corresponding to region R_1 is shown in Figure 3(b). Here the voltage at one node of the equivalent circuit represents a temperature distribution over a region in the geometry decomposition. Current flowing in one resistor of the equivalent circuit represents the heat flow rate or local convection between two adjacent regions. A constant temperature boundary condition is represented by a voltage source and heat sources in the region represented by independent current sources. Capacitance represents the thermal capacitance of the region and is directly proportional to the volume of the region.

2.4 The General Form of the Equivalent Circuit

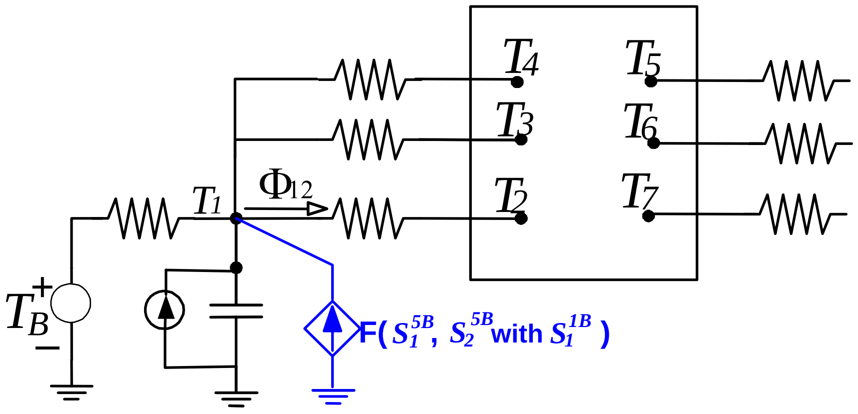
When all the ECs corresponding to all regions are assembled, one obtains an EC similar to that in Figure 4. The most difficult part to model is the so-indicated purely resistive network, which has all the resistors representing the surfaces through which heat flows between adjacent regions. This is also the most controversial part of the topology because it is dependent on how the geometry is carved.

The capacitors at the bottom of Figure 4 represent the thermal capacitance of every region and hence there are as many capacitors as there are regions. The current sources correspond to regions that have internal heat generation. The voltage-controlled current sources represent the radiation exchange between regions. The resistors outside the purely resistive network correspond to those regions having external boundaries. There are four different set of combinations:

- Using a capacitor with a current source and VCCS,
- Using a capacitance with only current source,
- Using a capacitance with a VCCS,
- Using a capacitance by itself.



(a)



(b)

Figure 3. Application of the electro-thermal analogy based on the concept of regions (a) A schematic of the geometry decomposition (b) The part of the EC that corresponds to the region R$_1$

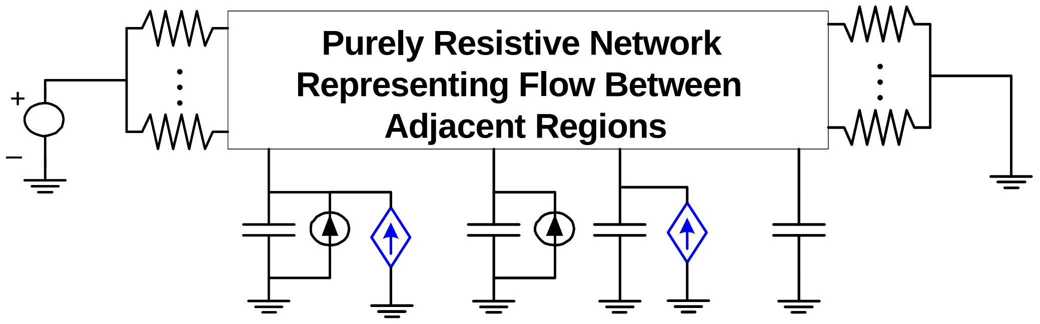

Purely Resistive Network Representing Flow Between Adjacent Regions
+
−

Figure 4. Topology of the EC including effect of radiation. The independent current sources represent internal (volume-based) heat generation and voltage-controlled current sources represent radiative heat exchange between regions

2.5 Mathematical Thermal Relations

In this section, we review the mathematical thermal relations that are to be captured by the EC. Thermal conduction in a region (say Region R_i) of a homogeneous solid material with heat conductivity k_i and three-dimensional coordinate r is governed by the heat diffusion equation which can be written as

$$\frac{\partial T(r,t)}{\partial t} = k_i \nabla^2 T(r,t), \tag{1}$$

where $T(r,t)$ is the temperature at the point r and at time t. An adiabatic boundary condition at r_0 is described by $\frac{\partial T}{\partial n}\big|_{r_0} = 0$ where n represents the outward normal at the boundary. The continuity of temperature across a boundary S_{ij} can be written as $T_i = T_j$ at S_{ij} where T_i denotes the temperature in Region i. The convection heat flow rate between Region i and a fluid at temperature T_F touching Region i over a wet surface A_{iF} is given by

$$\dot{Q}_{iF} = hA_{iF}(T_F - T_i), \tag{2}$$

where h_{iF} is the corresponding convection heat transfer coefficient. For forced convection, one can use the empirical formula

$$h_{iF} \approx 11\sqrt{1+4v_F} \quad \text{(SI Units)} \tag{3}$$

where v_F is the fluid relative speed [49]. A boundary condition combining heat flows due to conduction and convection is given by

$$0 = hA(T_F - T_1) + kA\frac{\partial T_1}{\partial n}. \tag{4}$$

Additional boundary conditions include the heat flux rate due to radiation exchange [50] between two boundary elements (dA_i at temperature T_i and dA_j at temperature T_j), given by

$$\dot{q}_{ij}^{\text{rad}} = \varepsilon\sigma(T_i^4 - T_j^4)g\left(dA_i, dA_j\right) \tag{5}$$

where $g\left(dA_i, dA_j\right)$ is a factor that depends on the geometrical relationship between the dA_i and dA_j, σ is the Stefan-Boltzmann constant ($5.670 \times 10^{-8}\text{Wm}^{-2}\text{K}^{-4}$), and ε is the surface emissivity. Also to be considered is heat flow due to friction heat generation (essentially a fixed heat flow source) and contact resistance (between two solids). A detailed discription will be included in Chapter 5 and 6.

2.6 Overview of Methodology

Figure 5 illustrates the main steps in the procedure of developing the EC model. The first step consists of defining the physical system such as its geometry, its material properties (density, thermal conductivity, specific heat, etc.), and the heating and cooling scenarios (dimensions, locations, dependence on electrical power if needed, and boundary conditions such as those due to forced convection cooling). The thermal system is then modeled using any available FE-based software, such as COMSOL, which makes available the FE mesh and the simulation results.

Fictitious heat sources can be added to the geometry as needed. A fictitious heat source is a new domain which has exactly the same material properties as the original domain into which it is fully embedded, but which is assumed to generate heat internally. In the absence of the assumed generated heat, the addition of a fictitious region will have theoretically no effect, but in practice, it will modify the FE mesh and may have a small numerical effect. The fictitious heat sources will be turned on, one at a time, and in each case a new FE simulation is conducted. In this manner, a richer set of "virtual measurement" data is obtained. The main objective of using these richer datasets is to estimate the inter-region conductances more accurately.

After describing the physical model and meshing its geometry, the mesh is optimally decomposed using a standard graph decomposition algorithm, such as METIS. The decomposed mesh is then used to generate the topology of the EC according to the guidelines discussed in Section 2.3.

The parameters of the EC are then estimated using some (usually half) of the measurement data, and the unused measurements are used to validate the EC model by comparing its predictions against the measurements. This cross-validation approach is standard in statistics and the theory of artificial neural networks [51]. The EC model is tested by comparing its steady-state and transient behaviors with those of the full FE simulation.

The above procedure is iterated until satisfactory agreement with the full-order model is achieved. Parameters that can be varied between iterations include the number of regions into which the geometry is decomposed, the number of fictitious regions added, and the number of simulation datasets generated under different heating and boundary conditions. Increasing the first two numbers increases the complexity of the EC, while increasing the last increases the available training data. The acceptable level of accuracy is usually determined by the application engineer.

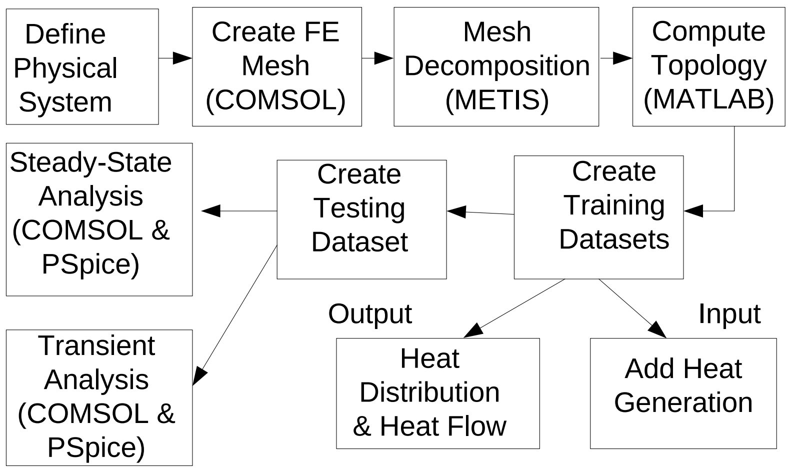

Define Physical System
Create FE Mesh (COMSOL)
Mesh Decomposition (METIS)
Compute Topology (MATLAB)
Steady-State Analysis (COMSOL & PSpice)
Create Testing Dataset
Create Training Datasets
Output
Input
Transient Analysis (COMSOL & PSpice)
Heat Distribution & Heat Flow
Add Heat Generation

Figure 5. Overview of methodology

CHAPTER 3
EQUIVALENT CIRCUIT FOR CONDUCTION AND FORCED CONVECTION

3.1 Geometry Decomposition

The essential graph partitioning problem is that of dividing the vertices of a graph into sets of specified sizes such that few edges cross between sets by partitioning a connected graph into smaller components through the removal of a set of edges or nodes. One can first solve the subproblems associated with the smaller subgraphs. The solutions to the subproblems are then combined to give the overall solution to the original graph. Graph partitioning is a fundamental problem in many scientific contexts. Algorithms that find a good partitioning of highly unstructured graphs are critical for developing efficient solutions for a wide range of problems.

For instance an electric circuit comprises elements such as transistors, resistors, and capacitors that are intricately wired together. To improve the efficiency and reliability of wiring, the circuit can be decomposed into subcircuits, and this decomposition can be attempted through graph partitioning [52-53]. By modeling the circuits in the form of graphs (components as nodes and wires as edges), one can view the layout problem as finding constructions to map graphs into layouts in the most area-efficient way [54-55].

Similarly, a computer network consists of interconnected sites that send, forward, and receive messages of various types. One is interested not just in knowing that it is possible to get the message from every site to every other site, but also in maintaining this connectivity for all pairs of sites as the network changes.

A graph partitioning problem can be used to identify areas which are minimally connected. As a final example, one can consider compiles which build graphs to represent the call structure of a large software system. The items are the various functions or modules that comprise the system; connections are associated either with the possibility that one function might call another or with actual calls while the system is in operation.

Other applications include circuit partitioning [56], efficient storage of large databases on disks and for parallel computing [57-61], data mining [62], and topology design of industrial networks [63].

There are many powerful graph partitioning algorithms such as k-way partitioning, feature-space analysis [64], and principal component analysis (PCA) [65]. Feature-space analysis can be used to partition the mesh into meaningful parts [64], but has the disadvantage of not being able to control the size or shape of the resulting clusters. Principal component analysis applies eigenvalue decomposition to matrices representing the graph adjacency relationship, and does not require pre-defined thresholds [65], but the resulting regions need not be well contiguous or of compact shape.

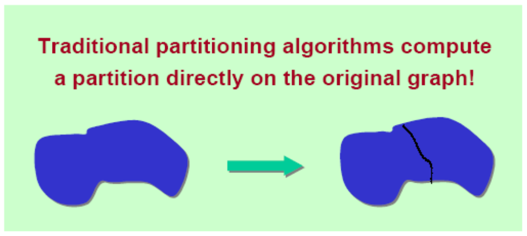
The k-way partitioning of a graph minimizes a measure of the how many edges are cut subject to various balancing constraints associated with the vertices [66-67]. A popular k-way partitioning algorithm is based on a recursive bisection paradigm that reduces the problem of computing a k-way partitioning to that of performing a sequence of bisections [68-69].

Recently, multilevel algorithms have been developed to solve problems of very large size. These multilevel algorithms approximate the original graph by a sequence of increasingly smaller graphs. The smallest graph is then partitioned using an efficient technique and this partition is propagated back through the sequence of graphs and refined. Multilevel techniques have been proposed in [70-73]. Walshaw [74] makes a case for the use of multilevel refinement as a metaheuristic for the solution of combinatorial problems. A fast method to partition a graph into R almost equally sized sets with a small cut set is presented by Karypis and Kumar [75]. This approach is adopted in this dissertation, because it produces high-quality partitions in the sense that it minimizes the number of edgecuts while keeping the sizes of the partitions almost equal. The total connection volume is defined as the number of vertices whose edges are cut by the partition. This methodology is captured by the METIS package which allows several alternative formulations, including multi-constraint partitioning, minimizing the total connection volume, minimizing the maximum connectivity of the subdomains, and reducing the number of non-contiguous subdomains. METIS also provides utilities that can partition meshes or graphs, or convert meshes to graphs.

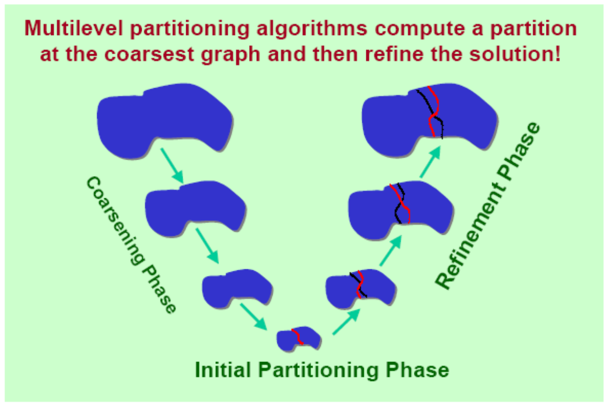
3.1.1 METIS: A Software Package For Partitioning

METIS is a software package for partitioning large irregular graphs, partitioning large meshes, and computing fill-reducing ordering of sparse matrices. The algorithms in METIS are based on multilevel graph partitioning. Traditional graph partitioning algorithms compute a partition of a graph by operating directly on the original graph as illustrated in Figure 6(a). These algorithms are often too slow and/or produce poor quality partitions.

Multilevel partitioning algorithms, on the other hand, take a completely different approach. These algorithms, as illustrated in Figure 6(b), reduce the size of the graph by collapsing vertices and edges, partition the smaller graph, and then uncoarsen it to construct a partition for the original graph. METIS uses novel approaches to successively reduce the size of the graph as well as to further refine the partition during the uncoarsening phase. During coarsening, METIS employs algorithms that make it easier to find a high-quality partition at the coarsest graph. During refinement, METIS focuses primarily on the portion of the graph that is close to the partition boundary. These highly tuned algorithms allow METIS to quickly produce high-quality partitions for a large variety of graphs.


Traditional partitioning algorithms compute a partition directly on the original graph!

(a)


Multilevel partitioning algorithms compute a partition at the coarsest graph and then refine the solution!
Coarsening Phase
Refinement Phase
Initial Partitioning Phase

(b)

Figure 6. (a) Traditional partitioning algorithms (b) Multilevel partitioning algorithms

3.1.2 Partitioning Objectives

Minimizing the Edge-Cut: Consider a graph $G = (V, E)$, and let P be a vector of size $|V|$ such that $P[i]$ stores the number of the partition that vertex i belongs to. The edgecut of this partitioning is defined as the number of edges that straddle partitions; that is, the number of edges (v, u) for which $P[v] \neq P[u]$.

Minimizing the Total Communication Volume (TCV): Let $V_b \subset V$ be the subset of interface (or border) vertices. That is, each vertex $v \in V_b$ is connected to at least one vertex that belongs to a different partition. For each vertex $v \in V_b$, let $N_{adj}[v]$ be the number of domains other than $P[v]$ that the vertices adjacent to v belong to. The TCV of this partitioning is defined as:

$$V_{TC} = \sum_{v \epsilon V_b} N_{adj}[v]. \tag{6}$$

Equation (6) corresponds to the total communication volume incurred by the partitioning because each interface vertex v needs to be sent to all of its $N_{adj}[v]$ partitions.

3.1.3 METIS Mesh Partitioning Programs

METIS provides two programs `partnmesh` and `partdmesh` for partitioning a mesh into *k* equal-size parts. These programs take as input the element node array of the mesh and compute a partitioning for both its elements and its nodes. METIS currently supports four different types of mesh elements which are triangles, tetrahedra, hexahedra (bricks), and quadrilaterals. These programs first convert the mesh into a graph, and then use `kmetis` to partition this graph. The difference between these two programs is that `partnmesh` converts the mesh into a nodal graph (i.e., each node of the mesh becomes a vertex of the graph), whereas `partdmesh` converts the mesh into a dual graph (i.e., each element becomes a vertex of the graph). In the case of `partnmesh`, the partitioning of the nodal graph is used to derive a partitioning of the elements. In the case of `partdmesh`, the partitioning of the dual graph is used to derive a partitioning of the nodes. Both of these programs produce partitioning of comparable quality, with `partnmesh` being considerably faster than `partdmesh`. However, in some cases, `partnmesh` may produce partitions that have higher load imbalance than `partdmesh`.

The dual graph format of an unstructured grid may list the connectivity of each vertex or each cell (element), depending upon which option is chosen. Hence, for an unstructured grid one may calculate the dual graph of either the vertices or the elements. Let us consider the trivial triangular finite element mesh in Figure 7(a). Here there are 3 linear triangular finite elements and a total of 5 vertices which make up the mesh. The corresponding dual graph with respect to the cells (or elements) is shown in Figure 7(b) or in tabular format as in Figure 7(c).

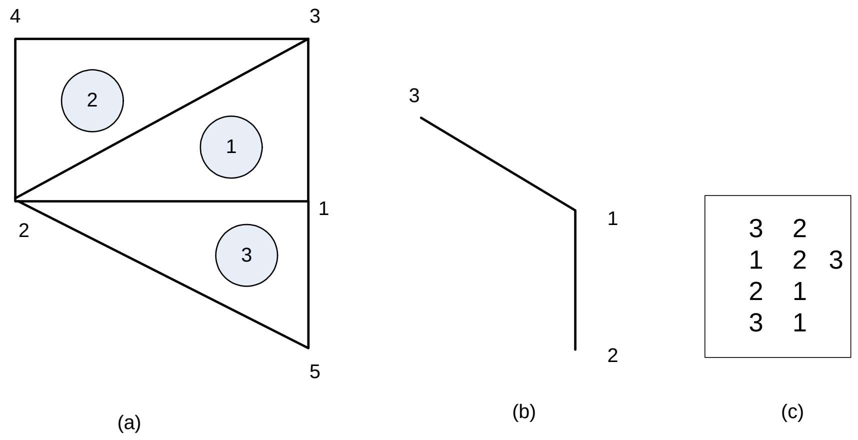


Figure 7. (a) Triangular Mesh (b) Dual graph showing the adjacency of the elements (c) Tabular format used in METIS

Figure 8(a) and (b) show nodal and dual graph respectively for a complex geometry. It is clear that region boundaries in dual graph are more smooth than in the nodal graph.

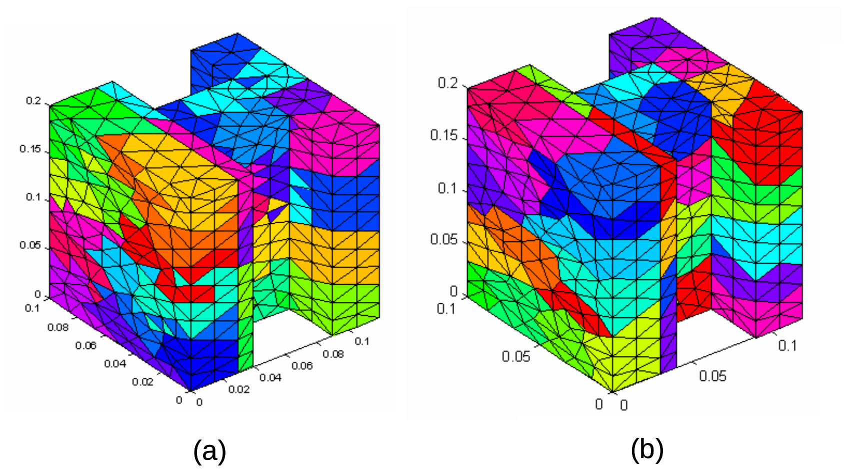
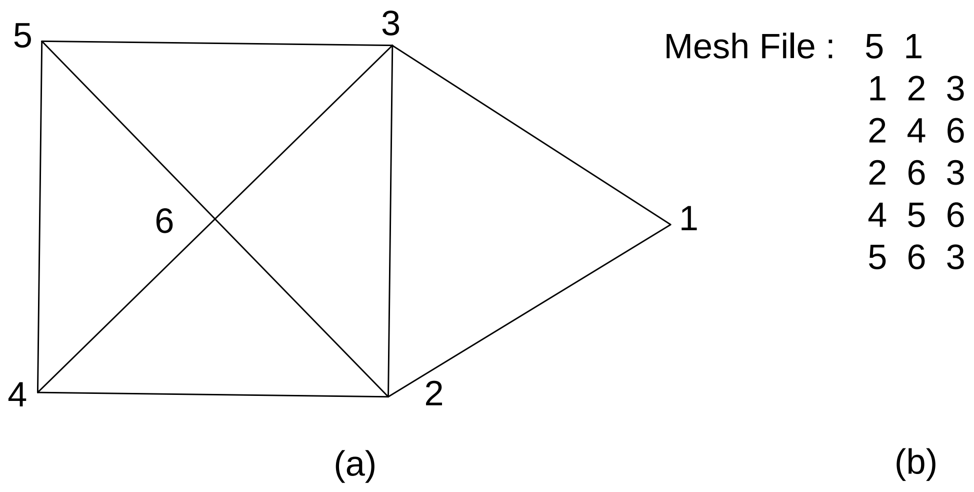


Figure 8. (a) Decomposition based on the nodal graph (b) Decomposition based on the dual graph

3.2 Interface Functions

Figure 9(a) illustrates a small mesh with triangular elements. The primary input of the mesh partitioning programs in METIS is the mesh to be partitioned. A mesh with n elements is stored in a plain text file that contains $n + 1$ lines in the form of the element node array. The first line contains information about the size and the type of the mesh, while the remaining n lines contain the nodes that make up each element. The first line contains two integers as seen in Figure 9(b).



```
Mesh File :  5 1
             1 2 3
             2 4 6
             2 6 3
             4 5 6
             5 6 3
```

Figure 9. Sample Mesh File

The first integer is the number of elements in the mesh. The second integer is used to denote the type of elements that the mesh is made of. It can take the values of 1, 2, 3, or 4, indicating that the mesh consists of either triangle, tetrahedra, hexahedra or quadrilaterals respectively. After the first line, the remaining n lines store the element node array. In particular for element i, line $i + 1$ stores the nodes that this element is made of. Depending on element type, each line can either have three integers (in the case of triangles), four integer (in the case of tetrahedra and quadrilaterals), or eight integers (in the case of hexahedra). Note that the element type field of the mesh file is set to 1 indicating that the mesh consists of triangular elements.

An interface between MATLAB and METIS is developed for this dissertation, which can for instance, take a description of the FE mesh as an input and return a text file that can be read by METIS. Another function reads the decomposition returned by

METIS and constructs in MATLAB the resulting EC topology whose parameters are to be estimated.

1- The following function creates METIS Mesh file from COMSOL mesh data. The output of this function is a simple text file.

   `CreateMetisMeshFile(FileName, Elements, Nodes, Etype):`

   - `Filename`: Name of the file that will store the mesh;
   - `Elements`: The elements belonging to the domain;
   - `Nodes`: The nodes belonging to the domain;
   - `Etype`: The type of elements that the mesh is made of.

2- The following function partition the created mesh file to regions.

   `MeshDecompose(MeshFile, option, Nparts)`

   - `MeshFile`: Name of the file that stores the mesh. This file is created by the `CreateMetisMeshFile.m` function;
   - `option`: If `option = "dual"`, use `partdmesh` for dual graph, if `option= "nodal"`, use `partnmesh` for nodal graph;
   - `Nparts`: The number of partitions that is desired.

     The graph decomposition is returned in two files named: `MeshFile.npart.Nparts` which stores the partitioning of the nodes, and `MeshFile.epart.Nparts` which stores the partitioning of the elements. The partition file of a graph with n vertices consists of n lines with a single number per line. The i^{th} line of the file contains the partition number that the i^{th} vertex belongs to. Partition numbers starts from 0 up to the number of partitions minus one.

3- The function `Mesh2Graph` converts a mesh to a graph representing regions. METIS provides two programs `mesh2nodal` and `mesh2dual` for converting a mesh into the graph format used by METIS. `mesh2nodal` converts the element node array of a mesh into a nodal graph. `mesh2dual` converts the element node array of a mesh into a dual graph. This following function partition the created mesh file to the number of regions.

   `Mesh2Graph(MeshFile,option)`

   - `MeshFile`: Name of the file that stores the mesh. This file created by `CreateMetisMeshFile.m` function;
   - `option`: If `option = "dual"`, use `partdmesh` for dual graph, if `option= "nodal"`, use `partnmesh` for nodal graph.

Actual graph is returned in a file named: `MeshFile.ngraph` in the case of `mesh2nodal`, and `MeshFile.dgraph` in the case of `mesh2dual.`

3.3 Equivalent Circuit Parameter Estimation

Once the FE mesh is decomposed, the topology of the EC is then determined, and FE simulation data is used to estimate the component values of the EC. The following approach is adopted:

Resistors: If two regions have at least one common face, they are deemed to be adjacent and hence connected by a thermal resistance whose value is to be estimated. From FE simulation data, the flow rate across any face separating the two regions is estimated, and the volume, average temperature and amount of heat generation in each region are computed. In simple and well-behaved examples, a formula as simple as

$$R^{kl} = \frac{\text{average } \Delta T \text{ between adjacent regions } k \text{ and } l}{\text{Heat flow } \Phi \text{ between adjacent regions } k \text{ and } l} \tag{7}$$

can be satisfactory. The conductance between two non-adjacent regions is set to zero.

Note however that Equation 7 is not a robust estimator of thermal resistances in many cases, especially when the heat flow paths are more or less tangential to the surface separating the two adjacent regions. In this case, ΔT^{kl} and Φ^{kl} can be both theoretically zero, but numerically corrupted with round-off error, and the above ratio can have a very large variance.

To overcome this problem, several datasets must be used to estimate R^{kl}. To this end, a factor μ_i^{kl} is used to emphasize the estimate of R_i^{kl} from dataset i where the flow is most perpendicular to the face. Hence one computes

$$\mu_i^{kl} = \sum_{m}^{\text{faces for } R^{kl}} \frac{|\overrightarrow{dA^m} \cdot \overrightarrow{F_{av}^m}|}{\left|\overrightarrow{dA^m}\right| . \left|\overrightarrow{F_{av}^m}\right|} \tag{8}$$

where $\overrightarrow{dA^m}$ is a vector whose magnitude is the area of the m^{th} face separating regions R_k and R_l, and F_{av}^m is the average flux through that surface. Hence every term in the above sum represents a cosine. The resistance R^{kl} which is between regions k and l can be calculated as the weighted normalized sum

$$R_{avg}^{kl} = \frac{\sum\limits_i R_i^{kl} . \mu_i^{kl}}{\sum\limits_i \mu_i^{kl}}. \tag{9}$$

Thermal resistances between a region and a heat flux boundary can be calculated from the material properties and the area of the external surface. By using the convective heat transfer coefficient, the convective thermal resistors from a surface to a heat flux boundary can be calculated using

$$R_{convective} = \frac{1}{hA} \tag{10}$$

where h is the heat transfer coefficient and A is the area of the wet surface.

The steps used to estimate the resistances can be summarized as follow:

1- Find the average temperature for each region, $T_{avg}(R_i)$.

2- Compute R_i^{kl} according to Equation 7 for dataset i.

3- Compute μ_i^{kl} using Equation 8.

4- Compute R_{avg}^{kl} using Equation 9.

5- Compute $R_{convective}$ using Equation 10.

Capacitors: The thermal capacitance corresponding to a region is directly proportional to its volume

$$C_i = c_p \rho \text{Vol}(R_i) \tag{11}$$

where Vol(R_i) is volume of the region i, c_p is its specific heat, and ρ is the density of the material.

Current Sources: These are computed directly from the FE by simply adding all the heat generation rates in that region.

Voltage Sources: Conductive and convective boundary conditions are represented by voltage sources in series with resistors. For convective boundary conditions, a voltage source represents the average temperature of the cooling fluid flowing over the region boundary. In this work, that temperature is assumed given or measurable. For isothermal boundary boundaries, a voltage source is specified by the boundary condition and the voltage is attached through a very small resistor that accounts for the difference between the boundary temperature and the average temperature in a region.

Voltage-Controlled Current Source (VCCS): Figure 10 shows voltage-controlled dependent current source. The dependent source is a two-port network, with the current at one port (terminals denoted as n+ and n-) controlled by the voltage at the other port (terminals denoted as nc+ and nc-). The current equals to g times V_1 and flows from node "$n+$" through the source and out "$n-$". "$n+$" and "$n-$" are the positive and negative nodes, respectively. g is called the transconductance and has the dimension of siemens (inverse ohms). The element shown is linear, but nonlinear elements can also be used in theory and practice.

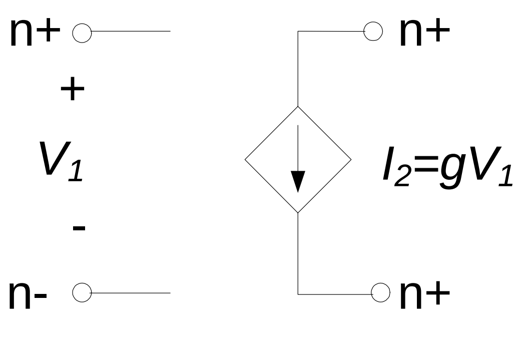


Figure 10. Voltage-controlled current source

The PSPICE representation of the voltage-controlled current source is :

$$\texttt{G} < \texttt{name} > < (+) \ < \texttt{node} > < (-)\, \texttt{node} > \ \texttt{VALUE} \ = \ \{< \texttt{expression} >\}$$

where the (+) and (-) nodes are the output nodes. Positive current flows from the (+) node through the source to the (-) node.

The presence of nonreciprocal elements, which are modeled by controlled sources, affects the analysis of the circuit. Simple circuits may be analyzed through the direct application of Kirchhoff's laws to branch circuit variables. Controlled sources enter this process similar to the constitutive relations defining R, L, and C, i.e., in defining relationships between branch circuit variables. Thus, control sources add no complexity to this basic technique.

Once the electric network model has been set up for the heat conduction equation, the numerical treatment of the analog electric circuit equation can be easily done with the computer code PSPICE. We have developed a MATLAB interface to PSPICE that can, for instance, generate the PSPICE model from an EC description in MATLAB. In other words, it writes the PSPICE source file (filetype ".CIR") describing the EC, (e.g. resistors, capacitors and other elements, connections, the models of the elements and the type of analysis to be conducted).

For coupled electro-thermal simulation, the PSPICE description of the electrical circuit is appended to the PSPICE description of the EC representing the thermal behavior. For instance, a typical diode PSPICE model has the form

$$\texttt{DXX}\, \texttt{N}+\ \texttt{N}-\ \texttt{ModelName} \ldots\ < \texttt{TEMP} = \texttt{T} >$$

where `N+` and `N-` are the electrical nodes the diode is connected to, and the temperature of the diode junction `TEMP` must be made equal to the appropriate node voltage in the EC part. Careful bookkeeping is required. Assume the diode is located in Region 25 represented by node 37 in the EC whose voltage is denoted `VEC37`. Then `DXX N+ N- ModelName ... TEMP=VEC37`. Similarly, let `PowerDiode` represent the electrical power dissipated across the diode. Then a heat source represented by a current source must be added to the EC Region 25 whose value is given by `PowerDiode.`

3.4 Generating Rich Training Datasets

In any given simulation, the heat flow streamlines may be too sparse in one part of the geometry as to allow accurate estimation of the thermal resistances in that part. Also, the streamlines may be too parallel to allow reliable estimation of many thermal resistances, especially those which are physically oriented perpendicularly to the streamlines. In Figure 11, one heat source is placed in one of the subdomains, and it can be seen that thermal resistors located far from the heat source and from the streamlines cannot be estimated accurately.

For complex geometries decomposed into a large number of regions, one needs a systematic procedure to generate measurement datasets that can be used to estimate all thermal resistances accurately. We therefore propose a method in which fictitious heat sources are introduced and used for the training stage only and later removed in the testing stage. A fictitious heat source is built by defining small fictitious regions (FR) inside a homogeneous domain. A fictitious region has exactly the same material as that of the domain it belongs to, but it can be assigned a heat source. This approach provides the flexibility to place heat sources correctly and to calculate heat flows easily under different heating scenarios. Every fictitious region will be considered a new domain. The FRs are very small and are not further decomposed. Since every domain is decomposed separately, every FR will contribute at least one node to the EC.

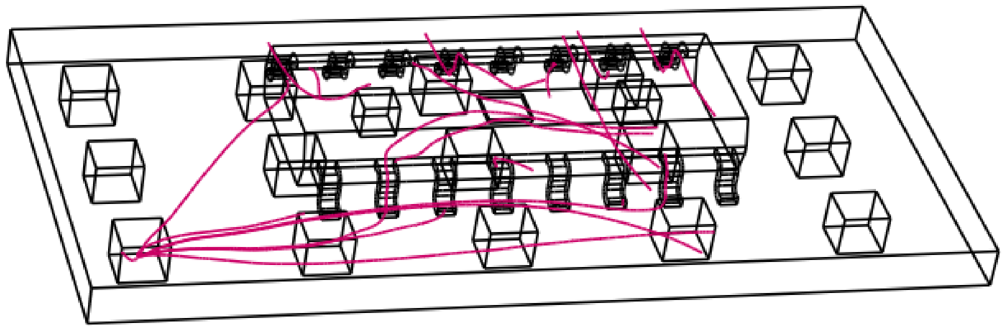


Figure 11. Heat flux streamlines (in gray) for one heat source placed in a fictitious sub domain

CHAPTER 4
VALIDATION OF THE REDUCED-ORDER CONDUCTION MODEL

4.1 Validation Using A Thin 3D Rectangular Slab

The proposed methods are here applied to a rectangular slab to obtain the EC for its thermal behavior. Constant temperature boundary conditions and heating scenarios were simulated to identify training and testing sets. The steady-state and transient analysis were considered for both scenarios.

4.1.1 Geometry and Mesh Decomposition

Figure 12 shows the geometry of the rectangular thin slab with 6 fictitious domains in 3D. The dimensions of the model and fictitious domains are 10 x 21 x 1 and 1 x 1 x 1 (in centimeters) respectively.

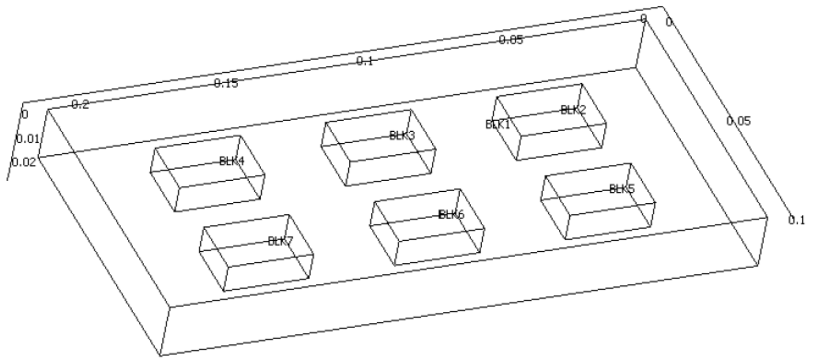


Figure 12. Rectangular slab with 6 fictitious domains

For this model, COMSOL generated 471 nodes, $1,282$ tetrahedral elements, 938 triangular elements on the boundaries and 104 elements on the edges. There are 3032 distinct surfaces. The material used is Aluminum whose properties are shown in Table 2.

Table 2. Material properties for a thin 3D rectangular slab

Material Properties	**Value**	**Unit**
Density	2700	[kg/m^3]
Thermal Conductivity	160	[W/m.K]
Heat Capacity	900	[J/kg.K]

4.1.2 Training Data and Obtaining EC

A constant-temperature boundary condition and several heating scenarios were simulated to obtain training and testing datasets. The main objective in using these datasets is to achieve a large-scale EC which is independent of boundary condition. Usually more than one dataset is need to obtain good estimates for the unknown thermal conductances. Six different training datasets were generated. Each training dataset corresponds to one fictitious heat source set to nonzero. Figure 13 shows the rectangular slab with a heat source in one of the fictitious domain. The modal was simulated assuming a uniform constant temperature 275K on the upper side boundary and 303K on the lower side boundary.

4.1.3 Steady-State Performance of EC

For testing purposes two heat sources as in Figure 14, are placed in two fictitious domains. Table 3 shows the comparison between the FE and EC steady-state simulations. The average percentage error in steady-state temperature average over a region was 0.2375%.

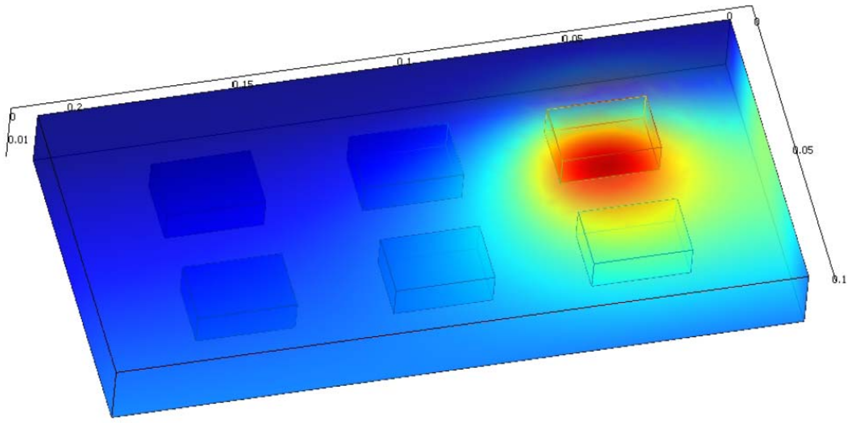


Figure 13. Rectangular slab with a fictitious heat source

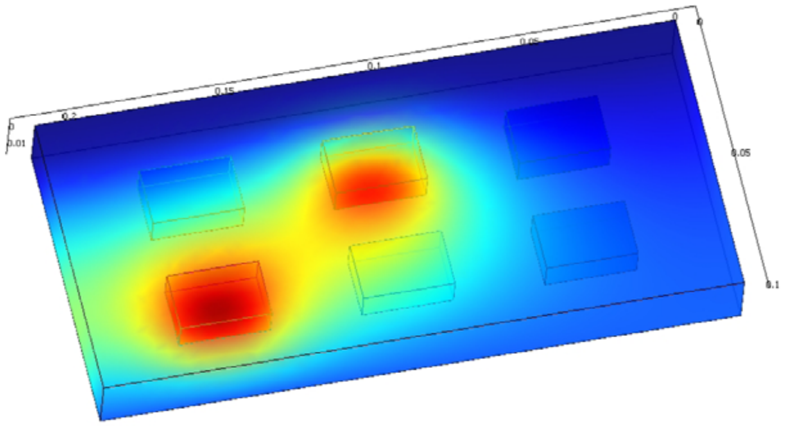


Figure 14. Heat generation placed in two fictitious domains

Table 3. Comparison of a full-model with a reduced-model at Steady-State average temperature

Region No	FE Result	EC Result
1	302.65	303.65
2	286.05	284.92
3	284.22	285.43
4	305.62	306.00
5	311.11	319.05
6	303.70	305.69
...	...	...

4.1.4 Transient Performance of EC

Figure 15 shows a transient analysis comparison for four different regions. The COMSOL curves represent the average temperature in the corresponding regions as obtained by FE simulations (all capacitors in the EC are initially discharged). The agreement of simulation and measurement is almost perfect.

4.2 Validation Using Motor Pole Example

The proposed methods are here applied to a rmotor pole to obtain the EC for its thermal behavior.

4.2.1 Geometry and Mesh Decomposition

The second model considered is a part of a motor pole similar to that considered in [76]. This model is shown in Figure 16(a) along with 15 FRs. Figure 16(b) shows the top view of the motor pole where the boundary conditions (BC) are shown. The FE mesh used consisted of $1,121$ nodes, $4,559$ tetrahedral elements and $2,047$ triangular elements on the boundaries.

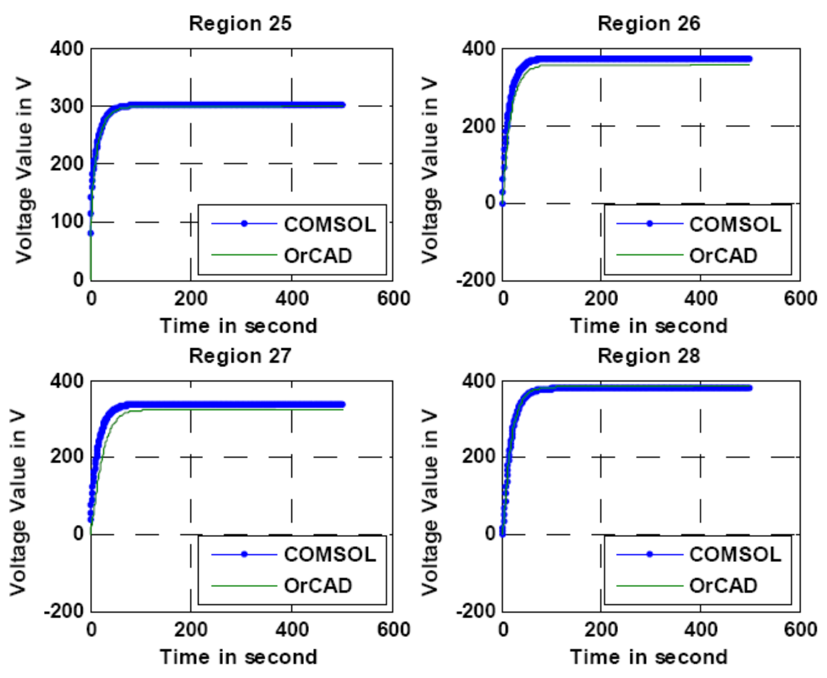


Figure 15. Comparison of transient behaviors obtained by FE models (denoted COMSOL) and EC models (denoted OrCAD)

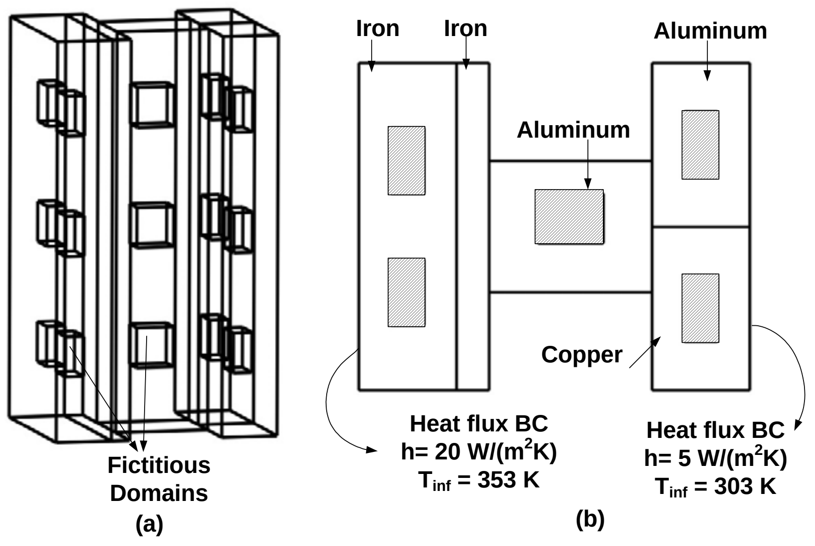


Figure 16. (a) Motor pole with fictitious subdomains (b) Top view of the motor pole with materials. All boundaries are insulated except for the leftmost and rightmost boundaries

The heat transfer coefficient h is 20W/m^{2}K and external temperature is 353K for the leftmost boundary. The heat transfer coefficient h is 5W/m^{2}K and external temperature is 303K for the rightmost boundaries. The other sides are thermally insulated. The material properties are shown in Table 4.

Table 4. Material properties for part of a motor pole

Material Property	Iron	Aluminum	Copper
Density	7650 [kg/m^3]	2705 [kg/m^3]	7360 [kg/m^3]
Thermal Conductivity	42 [W/m-K]	230 [W/m-K]	360 [W/m-K]
Specific Heat	447 [J/kg-K]	900 [J/kg-K]	385 [J/kg-K]

We introduced 15 fictitious subdomains as shown in Figure 16(a). The temperatures across the boundaries of the fictitious regions are required to be continuous. The FE mesh of the motor pole is decomposed by METIS into 302 regions as shown in Figure 17, where some of the regions are the preassigned fictitious subdomains. We used 15 training datasets with each training dataset corresponding to one fictitious heat source being nonzero. There are only 1279 nontrivial adjacencies of the total possible $\frac{302\times301}{2} = 45,451$, and hence the purely resistive subnetwork in Figure 4 will have 1279 resistors. In general, the number of regions required for getting an accurate reduced-order model depends on the complexity of the system to be modeled, as is typically determined iteratively. Complexity is measured here by the number of different domains, and their materials, aspect ratios, shape complexity, and number of adjacent domains, and this complexity is generally well reflected by the required storage or computation requirement in finite-element codes.

4.2.2 *Steady-State Performance of EC*

For testing purposes, two heat sources are placed in two fictitious domains as shown in Figure 18. The comparison between the steady-state behaviors of the FE model (using COMSOL) and the EC (performed in PSPICE) is shown in Table 5 for five regions/nodes. The average relative error in steady-state temperature rise for the 302 regions was 0.2%.

4.2.3 *Transient Performance of EC*

We simulated the step response for 500 seconds. The FE simulation was too slow and barely reached its steady-state. Figure 19 compares the average temperatures in the regions as computed by the FE model to that obtained as voltages in the EC. The agreement is excellent.

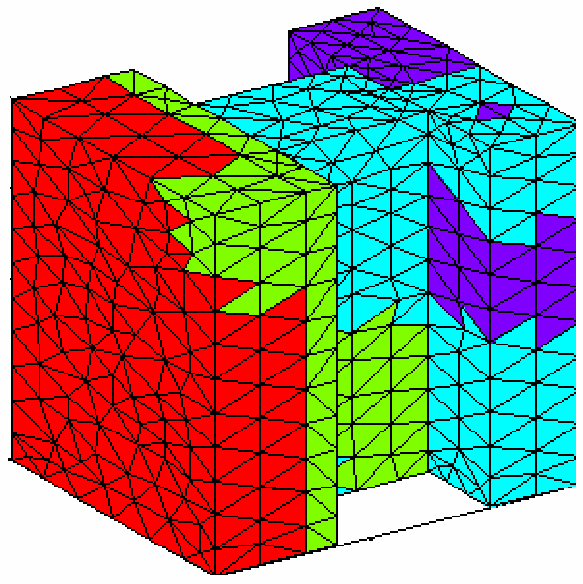


Figure 17. The view of 302 regions obtained through METIS

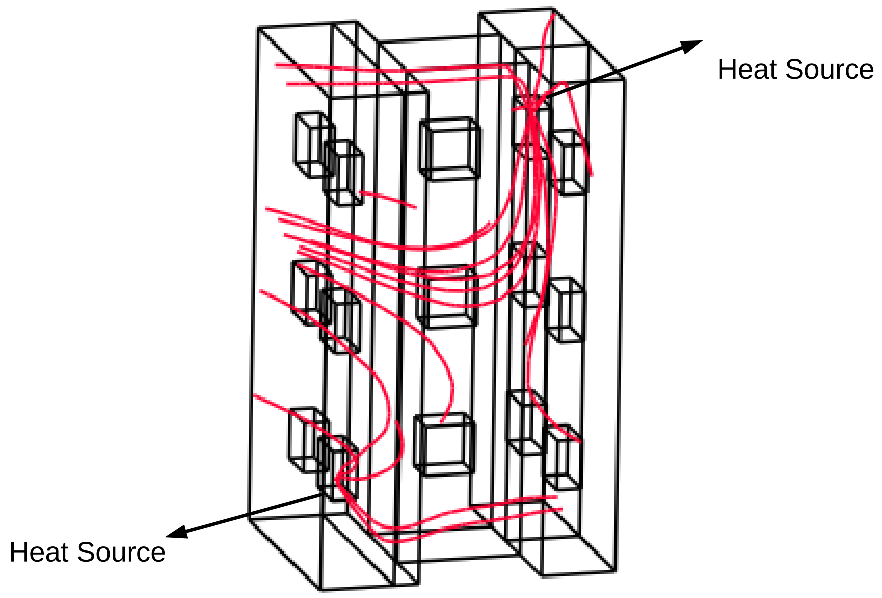


Figure 18. Heat flow in the Motor Pole

Table 5. Comparison of a full-model with a reduced-model at Steady-State average temperature

Region No	**FE Result**	**EC Result**
1	662.6935	661.84
2	663.6000	662.85
3	661.6125	660.82
4	661.8932	661.48
5	662.6563	662.10
...	...	...

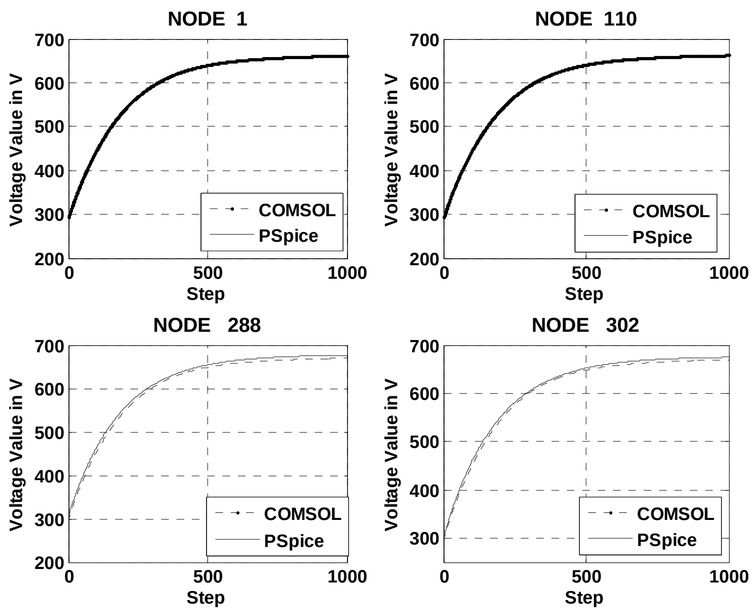


Figure 19. Comparison of transient thermal behaviors obtained by FE models (denoted COMSOL) and EC models (denoted PSpice)

4.3 Validation Using Electronic Chip Model

The proposed methods are here applied to a chip model to obtain the EC for its thermal behavior.

4.3.1 Geometry and Mesh Decomposition

A silicon chip is quite complicated and contains four main solid components which are a silicon chip, aluminum pins, a plastic package and a FR4 board as shown in Figure 20. The material properties are shown in Table 6. All the boundaries of the model are convective and maintained at an external temperature $T_{inf} = 303.15\,\mathrm{K}$ with a heat transfer coefficient $h = 50\,\mathrm{W/(\,m^2\,K)}$. The finite element (FE) model was discretized with unstructured tetrahedral elements. The model consists of $14,830$ nodes, $68,101$ tetrahedral elements and $11,514$ triangular elements on the boundaries. There are $141,112$ distinct surfaces belonging to 364 boundaries (both internal and external). We used 17 fictitious heat sources on the board and package (shown in Figure 20). The METIS software was made to decompose the board and package into 400 regions where some of the regions were simply a single pin (the pin constituted a whole region and a whole domain). We also placed a fictitious heat source inside each pin and one in the chip in addition to 17 fictitious heat sources in the package and board, thus leading to a total of 34 fictitious sources.

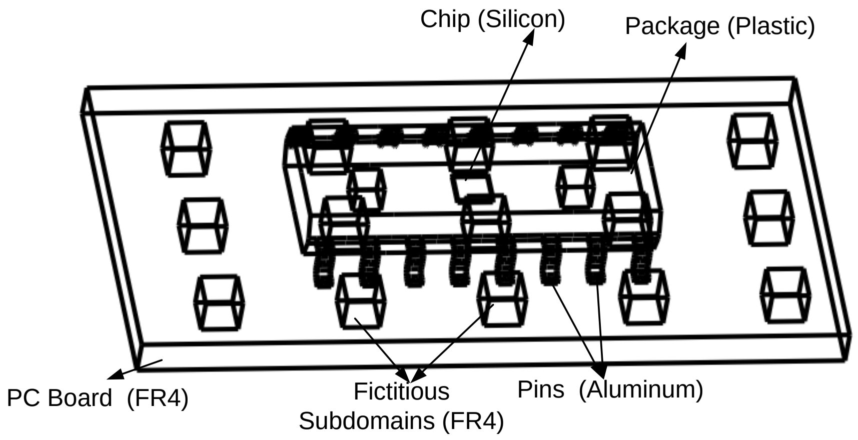


Figure 20. Electronic chip with many components

Table 6. Material properties for Silicon Clip

Material Property	**PCB (FR4)**	**Aluminum**	**Plastic**	**Silicon**
Density	2700 [kg/m^3]	2700 [kg/m^3]	2700 [kg/m^3]	2330 [kg/m^3]
Thermal Conductivity	0.3 [W/m-K]	160 [W/m-K]	0.2 [W/m-K]	163 [W/m-K]
Specific Heat	900 [J/kg-K]	900 [J/kg-K]	385 [J/kg-K]	703 [J/kg-K]

4.3.2 Steady-State Performance of EC

In Figure 21, the full-order temperature profile is shown for the electronic chip in steady-state. Table 7 compares the reduced-order model and the full-order model for few regions. The average relative error in the temperature rise is less than 7%. The histogram of these errors is shown in Figure 22. Out of 400 regions/nodes, 300 of them gave less than 5% error. One can reduce the error further by increasing the order of the equivalent model.

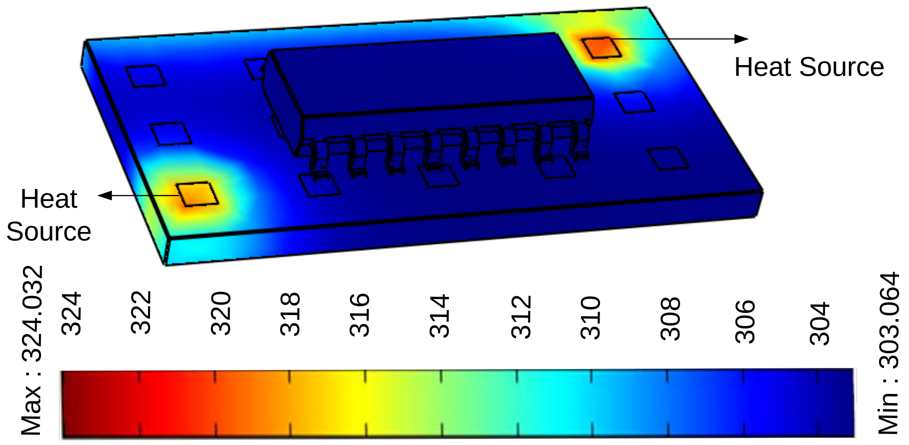


Figure 21. The temperature distribution on the surface

Table 7. Comparions of a full-model with a reduced-model at Steady-State average temperature

Region No	**FE Result**	**EC Result**
1	303.7500	304.8458
2	303.6800	304.6761
3	303.4600	303.9782
4	303.5600	304.3772
5	303.4000	303.9944
⋮	⋮	⋮

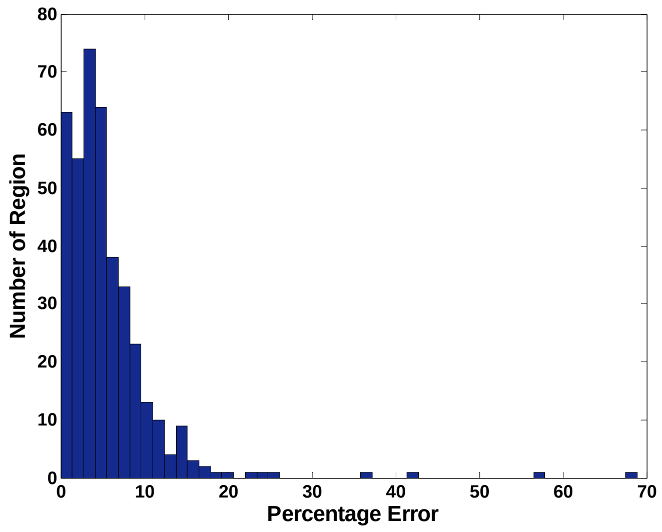


Figure 22. Histogram of relative error in the temperature rise

4.3.3 Transient Performance of EC

In Figure 23, the transient solution and error between the full model and the reduced-order model in case of constant heating power of 300MW/m^3 (according to Figure 21) are shown. The end time of the analysis was fixed at 500 seconds. In general, acceptable accuracy was obtained. As expected, the temperature rises very fast at the beginning and saturates after 100 seconds. The EC simulation is 11 times faster than the FE simulation.

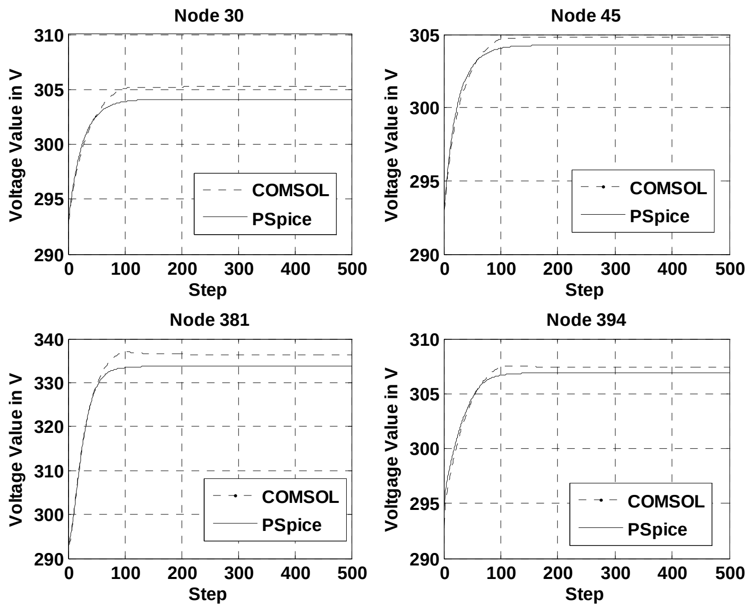


Figure 23. Comparison of transient thermal behaviors obtained by FE models (denoted COMSOL) and EC models (denoted PSpice)

CHAPTER 5
RADIATION

Both conduction and convection require the presence of a medium for the transfer of energy. Thermal radiation, on the other hand, is transferred by electromagnetic waves, or photons, which may travel over a long distance without interacting with a medium. The fact that thermal radiation does not require a medium for its transfer makes it of great importance in vacuum and space applications as well as to many industrial applications such as heating, cooling and drying processes, high-temperature film furnaces, rapid thermal processing, electronic thermal control, building interior thermal environments, solar energy usage and, spacecraft thermal control.

Radiation heat transfer is concerned with the exchange of thermal radiation energy between two or more bodies. In cases where all the objects are at low temperature (around room temperature), radiative heat exchange is so small that it can be neglected. The heat transferred into or out of an object by thermal radiation is a function of several properties. These include surface reflectivity, emissivity, surface area, absolute temperature, and geometric orientation with respect to other thermally participating objects.

5.1 Geometry

Radiation formulae are usually stated in terms of solid angles subtended by areas in 3D. The solid angle ω subtended by a surface S is defined as *the area A of the projection of S on the unit sphere.* It is measured in steradian. Recall that the surface of a sphere of radius r is $4\pi r^2$. Hence if S totally encloses the observation point, then the complete sphere subtends a solid angle of 4π.

For a general surface S, one can use the expression

$$\omega = \iint\limits_{S} \frac{\vec{n} \cdot \mathrm{d}\vec{S}}{r^2} \tag{12}$$

where $\vec{n}$ is a unit vector towards $\mathrm{d}S$, and the r^{-2} term is used to represent that the unit sphere is being projected on. It follows that if $\mathrm{d}S$ is small (w.r.t. r^2) and $\mathrm{d}S$ is orthogonal to r, then

$$\mathrm{d}\omega \approx \mathrm{d}S/r^2 \tag{13}$$

If on the other hand, $\mathrm{d}\vec{S}$ makes an angle θ with $\vec{r}$, then

$$\mathrm{d}\omega \approx \frac{\mathrm{d}S\cos\theta}{r^2}. \tag{14}$$

Figure 24 shows that the solid angle intercepted by the differential element at (r, ϕ, θ) in spherial coordinates is

$$d\omega = \frac{r^2 \sin\theta \mathrm{d}\theta \mathrm{d}\phi}{r^2} = \sin\theta \mathrm{d}\theta \mathrm{d}\phi. \tag{15}$$

Note that the intersection of a plane at distance $r\cos\theta$ with a sphere of radius r is a circle of radius $r\sin\theta$ and perimeter $2\pi r\sin\theta$.

5.2 Radiant Energy Terminology

The radiant energy Q has units of joules and generally has a complex dependence on several variables, including time, wavelength, spatial coordinates, temperature, surface area and relative orientation for the source [77].

The radiant flux or radiant power is the partial derivative with respect to time of radiant energy. It is given the symbol $\Phi = \frac{\mathrm{d}Q}{\mathrm{d}t}$ and has the units of watts (W).

The radiant intensity I, is defined as the radiant flux per unit solid angle radiated in a given direction from a point source. It is measured in watts per steradian (W/sr). Figure 25 provides a geometric interpretation of I for a point source radiating into a solid angle $\mathrm{d}\omega$ [77]. The defining equation is

$$I = \frac{\mathrm{d}\Phi}{\mathrm{d}\omega} = \frac{\mathrm{d}^2 Q}{\mathrm{d}t\mathrm{d}\omega} \quad \text{(W/sr)} \tag{16}$$

As a result, one can write

$$\mathrm{d}\Phi = I\mathrm{d}\omega \tag{17}$$

which expresses the radiation power intercepted in a solid angle $\mathrm{d}\omega$.

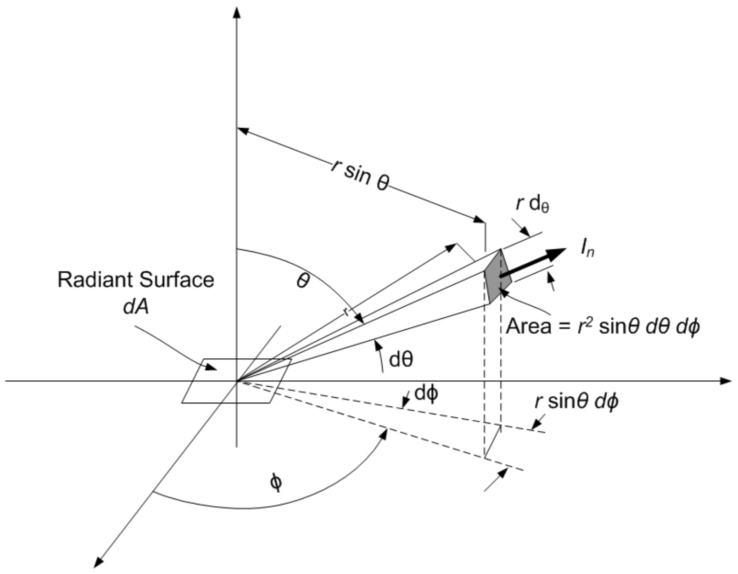


Figure 24. Radiant flux into a hemisphere

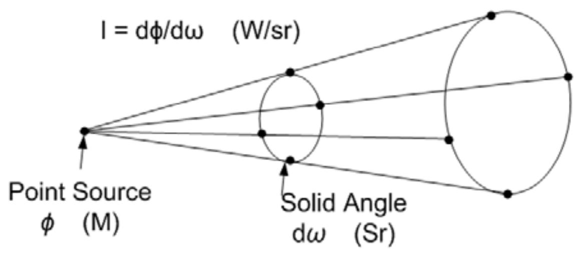


Figure 25. Radiant intensity

5.3 Surface-Related Radiant Quantities and Lambert's Law

The radiant exitance, M, emitted per unit area of the emitting surface has units of watts per square meter (W/m^2), and is illustrated in Figure 26:

$$M = \frac{\mathrm{d}\Phi}{\mathrm{d}A} = \frac{\mathrm{d}^2 Q}{\mathrm{d}t\mathrm{d}A} \quad (\mathrm{W/m^2}) \tag{18}$$

As a result, the power radiated by an area dA can be written

$$\mathrm{d}\Phi = M\mathrm{d}A. \tag{19}$$

The radiance, L, is the radiant flux in a given direction from a surface that is normalized with respect to both surface area and unit solid angle. For a viewing angle normal to the emitting surface as shown in Figure 27, the radiance is given by

$$L = \frac{\mathrm{d}I}{\mathrm{d}A} = \frac{\mathrm{d}^2\Phi}{\mathrm{d}\omega\mathrm{d}A} = \frac{\mathrm{d}^3 Q}{\mathrm{d}t\mathrm{d}\omega\mathrm{d}A} \quad (\mathrm{W/m^2\text{-}sr}). \tag{20}$$

Hence, the power radiated from an area dA and intercepted by a solid angle dω is

$$\mathrm{d}\Phi = L\mathrm{d}\omega\mathrm{d}A. \tag{21}$$

For directions other than normal to the emitting surface, consideration must be made of the fact that the apparent surface area decreases as cosθ for θ as shown in Figure 28 [77].

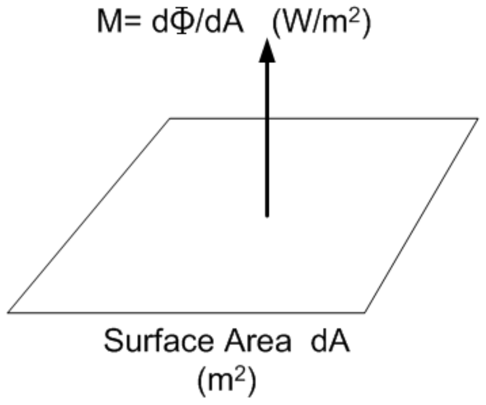


Figure 26. Radiant exitance

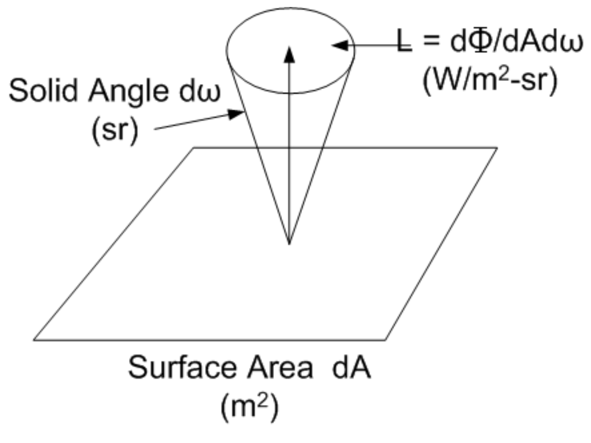


Figure 27. Radiance

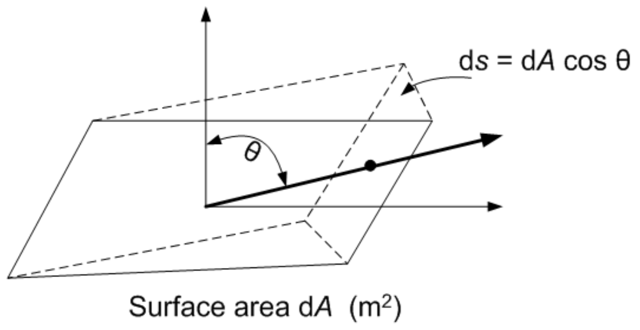


Figure 28. Radiance at angle θ

5.3.1 Lambert's Cosine Law

For radiant energy emitted by a planar surface, the radiant intensity $I = \frac{d\Phi}{d\omega}$ in W/sr varies as the cosine of the angle between the viewing direction and the emitting surface normal. Surfaces for which this relationship is valid are called Lambertian (or diffuse) surfaces. Lambert's cosine law is given by

$$I_\theta = I_n \cos\theta \qquad \text{(W/sr)} \tag{22}$$

Referring to Figure 28, the corresponding radiance in the viewing angle θ is given by [77]

$$L_\theta = \frac{\mathrm{d}I}{\mathrm{d}A} = \frac{I_\theta}{\mathrm{d}S} = \frac{I_n \cos\theta}{\mathrm{d}A\cos\theta} = L \qquad \text{(W/sr)} \tag{23}$$

Thus the radiance from a Lambertian surface is independent of the viewing angle. This relationship has been verified experimentally [77].

Then from the definition for radiant intensity I as $\frac{d\Phi}{d\omega}$ and from Figure 24,

$$\mathrm{d}\Phi = I_\theta \mathrm{d}\omega = I_n \cos\theta \sin\theta \mathrm{d}\theta \mathrm{d}\Phi. \tag{24}$$

Integrating the above relation gives the total radiant flux into a hemisphere

$$\Phi = \int_0^{\pi/2} \mathrm{d}\theta \int_0^{2\pi} \mathrm{d}\Phi I_n \cos\theta \sin\theta \tag{25}$$

or

$$\Phi = \pi I_n. \tag{26}$$

Furthermore, since radiance was shown to be independent of direction for a Lambertian surface, then

$$L = \frac{I_n}{\mathrm{d}A} = \frac{\Phi}{\pi \mathrm{d}A} = \frac{M}{\pi} \quad (\mathrm{W/m^2 \cdot sr}). \tag{27}$$

This result gives simple relationships among radiance L, radiant flux Φ, radiant exitance M, and radiant intensity I for a Lambertian source.

5.4 Gray Bodies

When radiation is incident on a surface, a portion of the total energy is absorbed in the material, a portion is reflected from the surface, and the remainder is transmitted through the body. The fraction absorbed is described by the absorptivity, or absorption coefficient α. The fraction reflected is described by the reflectivity or reflection coefficient ρ, and the fraction transmitted is descended by the transmissivity or transmission coefficient τ, as shown in Figure 29. This decomposition can be expressed by the relative fractions [78],

$$\rho + \alpha + \tau = 1. \tag{28}$$

The relative values of these coefficients depend on the material of the body and the state of its surface. In an opaque body, an incident beam can penetrate only a very short distance into the body. This distance is on the order of a micron for metals. The absorbed energy increases the temperature of the zone immediately adjacent to the surface, and then heat penetrates into the body by conduction. One may think, then, that all the process takes place at the body surface, and then, for an opaque solid, it can be considered that

$$\rho + \alpha = 1. \tag{29}$$

A body with $\alpha = \rho = 0$ is called a white body (then $\tau = 1$). A material that behaves roughly as a white body is glass. Glass is not a perfect white body. A body with $\rho = 1$ is a mirror. The fact that it heats up when exposed to solar radiation demonstrates that its absorption coefficient is not nil. A black body is one that absorbs all the incident radiation. Then, for a black body, $\alpha = 1$. For all real bodies, $\alpha < 1$. This indicates that the body reflects part of the incident radiation.

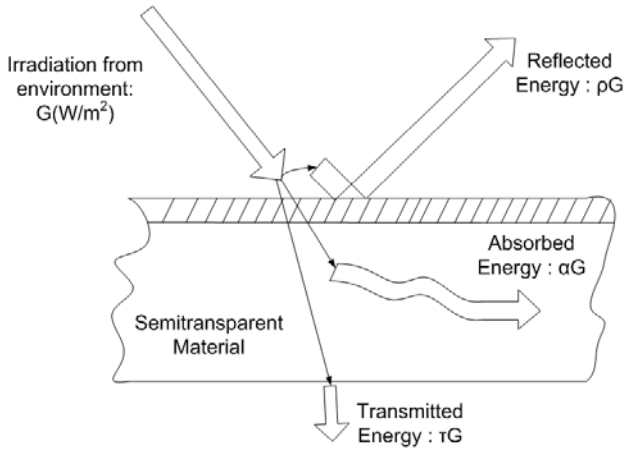


Figure 29. Schematic of a surface receiving irradiation and splitting it into absorbed, reflected and transmitted portions

To extend the simplicity of the Stefan-Boltzmann Law, the radiant power for a body with some absorption and reflection can be approximated as

$$\Phi = \varepsilon\sigma T^4 A \tag{30}$$

where

ε = emissivity of the object ($\varepsilon = 1$ for a black body and $\varepsilon < 1$ for a 'gray body').

If a hot object is radiating energy to its cooler surroundings, the net radiation heat loss rate can be expressed as

$$\Phi = \varepsilon\sigma(T_h^4 - T_c^4)A_c, \tag{31}$$

where

T_h = hot body absolute temperature (K),

T_c = cold surroundings absolute temperature (K),

A_c = area of the object (m^2).

The form of Equation (30) will be justified below for a black body ($\varepsilon = 1$).

5.5 Planck's and Stephan-Boltzmann's Laws

The most fundamental relation describing black-body radiation is given by Planck's Law [79]

$$I\left(\nu, T\right) = \frac{2h\nu^3}{c^2}\frac{1}{\exp\left(\frac{h\nu}{kT}\right) - 1} \tag{32}$$

where I is the *specific radiation intensity*, or emitted power per unit area of emitting surface in the normal direction, per unit solid angle, per unit frequency; h is Planck's constant, k

is Boltzmann's constant, ν is the frequency of the emitted electromagnetic radiation, T is absolute temperature, and c is the speed of light.

From Planck's Law, and under the Lambertian assumption, one can derive the Stefan-Boltzmann Law that the total energy radiated per unit surface area of a black body per unit time (here called the black-body irradiance),

$$M = \sigma T^4 \tag{33}$$

where

$$\sigma = \frac{2\pi^5 k^4}{15c^2h^3} = 5.6676 \times 10^{-8} \quad (\mathrm{Js^{-1}m^{-2}K^{-4}}) \tag{34}$$

is the Stefan-Boltzmann constant. The proof is tedious but simple and involves the integration of the power intercepted by the hemisphere and the integration over all frequencies. In other words, we must show that

$$M = \int_0^\infty I(\nu, T)\,\mathrm{d}\nu \int \cos\theta \mathrm{d}\omega \tag{35}$$

where $\mathrm{d}\omega$ is the solid angle subtended by the ring obtained on the unit sphere between the zenith angles θ and $\theta+\mathrm{d}\theta$. The area of that ring is $2\pi \sin\theta \mathrm{d}\theta$. In short,

$$\int_0^{\pi/2} 2\pi \cos\theta \sin\theta \mathrm{d}\theta = \pi \tag{36}$$

Now using the change of variable $u = \frac{h\nu}{kT}$ we obtain

$$\begin{aligned} M &= \frac{2\pi h}{c^2} \int_0^\infty \frac{\nu^3}{\exp\left(\frac{h\nu}{kT}\right) - 1} \mathrm{d}\nu \\ &= \frac{2\pi h}{c^2} \left(\frac{kT}{h}\right)^4 \int_0^\infty \frac{u^3}{e^u - 1} \mathrm{d}u \end{aligned} \tag{37}$$

where the integral can be shown to evaluate to $\pi^4/15$.

Hence

$$L = \frac{M}{\pi} = \frac{\sigma T^4}{\pi} \tag{38}$$

If the object is Lambertian, then the radiant intensity in a given direction is

$$I_\theta = I_n \cos\theta = \frac{\Phi}{\pi}\cos\theta = \frac{M\mathrm{d}A\cos\theta}{\pi}. \tag{39}$$

The dependence on θ is most easily incorporated into the term for area; that is, the projected area of dA in viewing direction θ is given by dS =dAcosθ.

5.6 View Factor

The total radiation power leaving a surface dA_1 can be written as

$$d\Phi_1 = M_1\mathrm{d}A_1 = \sigma T_1^4\mathrm{d}A_1. \tag{40}$$

As shown in Figure 30, the power intercepted by an area dA_2 at a distance r from dA_1 and at a zenith θ_1 as seen from dA_1 but which is normal to r is therefore, from Equation (21),

$$\mathrm{d}\Phi_2 = L_1\mathrm{d}A_1\mathrm{d}\omega_2 = \frac{M_1}{\pi}\mathrm{d}A_1\cos\theta_1\mathrm{d}\omega_2 \tag{41}$$

where dω_2 is the solid angle subtended by dA_2 and is simply dA_2/r^2. If dA_2 makes an angle θ_2 with r, then the intercepted power is

$$\mathrm{d}\Phi_2 = \frac{M}{\pi}\mathrm{d}A_1\cos\theta_1\frac{\mathrm{d}A_2}{r^2}\cos\theta_2. \tag{42}$$

The ratio

$$\mathrm{d}F_{1-2} = \frac{\text{power leaving } \mathrm{d}A_1 \text{ and intercepted by } \mathrm{d}A_2}{\text{total power leaving } \mathrm{d}A_1} \tag{43}$$

is called the view factor from $\mathrm{d}A_1$ to $\mathrm{d}A_2$. For this generic example, it is given by

$$\begin{aligned} \mathrm{d}F_{1-2} &= \frac{L\mathrm{d}A_1 \cos\theta_1 \frac{\mathrm{d}A_2}{r^2} \cos\theta_2}{M\mathrm{d}A_1} & (44) \\ &= \cos\theta_2 \cos\theta_1 \frac{\mathrm{d}A_2}{\pi r^2}, & (45) \end{aligned}$$

which is dimensionless. Equation (41) can also be written as

$$\mathrm{d}F_{1-2} = \frac{M_1 \mathrm{d}A_1 \cos\theta_1}{\pi M_1 \mathrm{d}A_1} d\omega_2 = \frac{\cos\theta_1 \mathrm{d}\omega_2}{\pi}. \tag{46}$$

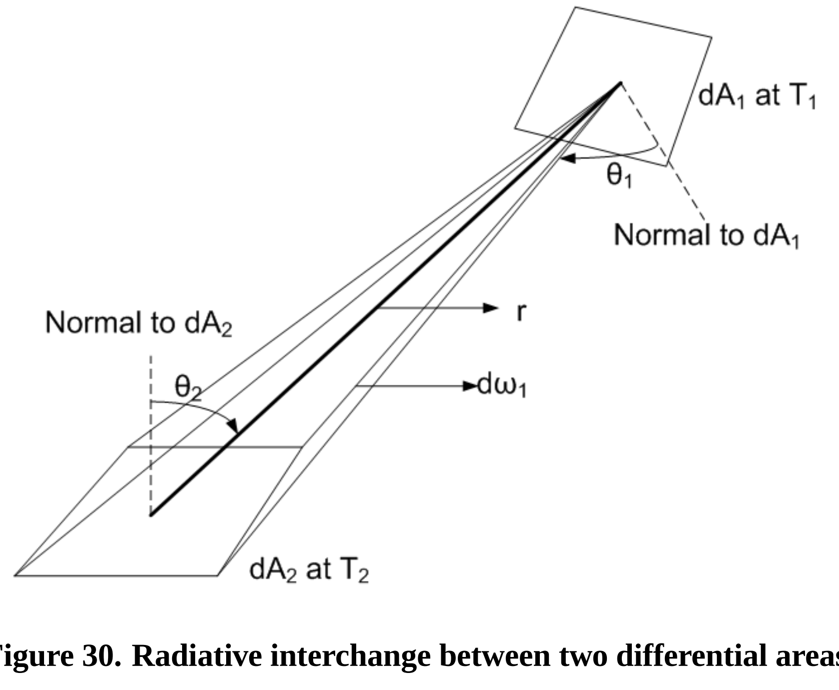


Figure 30. Radiative interchange between two differential areas

By symmetry:

$$\mathrm{d}F_{2-1} = \frac{\mathrm{d}A_1}{\pi r^2} \cos\theta_2 \cos\theta_1, \tag{47}$$

and the power emitted by $\mathrm{d}A_2$ and intercepted by $\mathrm{d}A_1$ is

$$\begin{aligned} \mathrm{d}\Phi_{2-1} &= \sigma T_2^4 \mathrm{d}A_2 \mathrm{d}F_{2-1} \\ &= \sigma T_2^4 \mathrm{d}A_2 \cos\theta_2 \cos\theta_1 \frac{\mathrm{d}A_1}{\pi r^2} \\ &= \sigma T_2^4 \mathrm{d}A_2 \mathrm{d}A_1 \frac{\cos\theta_2 \cos\theta_1}{\pi r^2}. \end{aligned} \tag{48}$$

Similarly, the power emitted by $\mathrm{d}A_1$ and intercepted by $\mathrm{d}A_2$ is

$$\mathrm{d}\Phi_{1-2} = \sigma T_1^4 \mathrm{d}A_1 \mathrm{d}A_2 \frac{\cos\theta_1 \cos\theta_2}{\pi r^2}. \tag{49}$$

The net power exchanged by the two surfaces is

$$\mathrm{d}\Phi_{12} = \sigma \left(T_1^4 - T_2^4\right) \mathrm{d}A_1 \mathrm{d}A_2 \frac{\cos\theta_1 \cos\theta_2}{\pi r^2}, \tag{50}$$

which can be written in any of the following alternative forms:

$$\begin{aligned} \mathrm{d}\Phi_{12} &= \sigma \left(T_1^4 - T_2^4\right) \mathrm{d}A_1 \mathrm{d}F_{1-2} \\ &= \sigma \left(T_1^4 - T_2^4\right) \mathrm{d}A_2 \mathrm{d}F_{2-1} \end{aligned} \tag{51}$$

from which one deduces that

$$\mathrm{d}A_1 \mathrm{d}F_{1-2} = \mathrm{d}A_2 \mathrm{d}F_{2-1}, \tag{52}$$

a property known as the reciprocity relation.

Given two finite surfaces S_1 and S_2, the total net heat rate exchanged by blackbody radiation is given by

$$q_{12} = \int_{S_1} \int_{S_2} \sigma \left(T_i^4 - T_j^4 \right) \mathrm{d}A_i \mathrm{d}A_j \frac{\cos\theta_i \cos\theta_j}{\pi r_{ij}^2} \tag{53}$$

where $\mathrm{d}A_i$ is assumed to belong to surface S_1 and $\mathrm{d}A_j$ is assumed to belong to surface S_2, T_i and T_j are the corresponding absolute temperatures, and

$$\cos\theta_i = \frac{\vec{n}_i \cdot \vec{r}_{ij}}{||\vec{n}_i|| \,.\, ||\vec{r}_{ij}||} \quad \text{and} \quad \cos\theta_j = \frac{\vec{n}_j \cdot \vec{r}_{ij}}{||\vec{n}_j|| \,.\, ||\vec{r}_{ij}||} \tag{54}$$

The total power emitted by S_1 can be expressed as

$$q_1 = \int_{S_1} \sigma T_i^4 \mathrm{d}A_i \tag{55}$$

while the total power emitted by S_1 and intercepted by S_2 is

$$q_{1-2} = \int_{S_1} \int_{S_2} \sigma T_i^4 \mathrm{d}A_i \mathrm{d}A_j \frac{\cos\theta_i \cos\theta_j}{\pi r_{ij}^2} \tag{56}$$

and their ratio is the view factor $F_{1-2} = q_{1-2}/q_1$ which depends on temperature in general. To ensure that the view factor is a geometrical concept and is independent of the temperature, one typically assumes that the temperature is uniform, in which case:

$$F_{1-2} = \frac{1}{A_1} \int_{S_1} \int_{S_2} \mathrm{d}A_i \mathrm{d}A_j \frac{\cos\theta_i \cos\theta_j}{\pi r_{ij}^2}. \tag{57}$$

5.7 View Factor for Two Infinite Strips

Following [38] we derive the view factor corresponding to two infinite strips that are essentially parallel. The two elemental areas in Figure 31 are located on strips that have parallel generating lines [38]. The elements $\mathrm{d}A_1$ and $\mathrm{d}A_2$ are at temperature T_1 and T_2, are arbitrary oriented, and have their normals at angles θ_1 and θ_2 to the line of length S joining them.

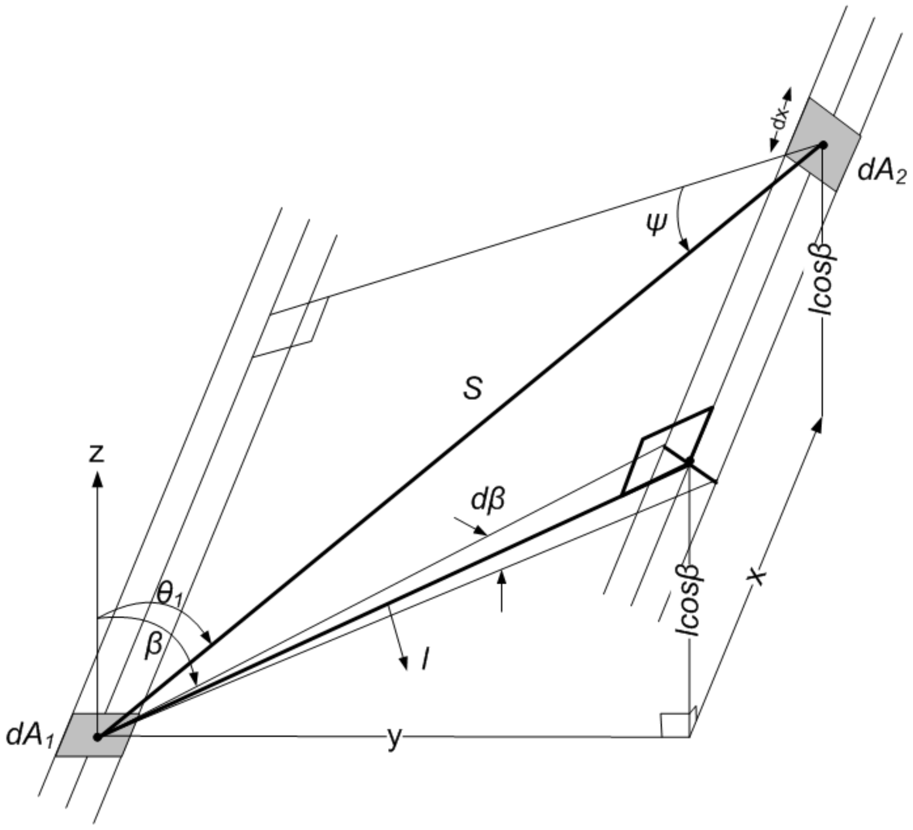


Figure 31. Geometry for configuration factor between elements on strips formed by parallel generating lines

The distance S can be expressed as

$$S^2 = l^2 + x^2 \tag{58}$$

and hence

$$\cos\theta_1 = \frac{l\cos\beta}{S} = \frac{l\cos\beta}{\sqrt{(l^2+x^2)}}. \tag{59}$$

The angle β is in the cross section (yz plane) normal to the two strips. The solid angle subtended by dA_2, when viewed from dA_1, is

$$\begin{aligned} d\omega_1 &= \frac{\text{projected area of d}A_2}{S^2} \\ &= \frac{(\text{Projected width of d}A_2)(\text{Projected length of d}A_2)}{S^2} \\ &= \frac{(l\mathrm{d}\beta)(\mathrm{d}x\cos\Psi)}{S^2} = \frac{l\mathrm{d}\beta\mathrm{d}x}{S^2}\frac{l}{S} \end{aligned} \tag{60}$$

Substituting Equation (59) and Equation (60) into Equation (46) gives

$$\begin{aligned} \frac{\cos\theta_1 d\omega_2}{\pi} &= dF_{d1-d2} \\ &= \frac{\cos\theta_1 \mathrm{d}\omega_2}{\pi} \\ &= \frac{1}{\pi}\frac{l\cos\beta}{\sqrt{(l^2+x^2)}}\frac{l^2\mathrm{d}\beta\mathrm{d}x}{(l^2+x^2)^{3/2}} \\ &= \frac{l^3\cos\beta\mathrm{d}\beta\mathrm{d}x}{\pi(l^2+x^2)^2} \end{aligned} \tag{61}$$

which is the desired configuration factor between dA_1 and dA_2.

To find the factor when dA_2 becomes an infinite strip as in Figure 31 [38], integrate over all x to obtain

$$\begin{aligned} dF_{d1-strip,2} &= \frac{l^3 \cos\beta \mathrm{d}\beta}{\pi} \int_{-\infty}^{\infty} \frac{\mathrm{d}x}{(l^2+x^2)^2} \\ &= \frac{l^3 \cos\beta \mathrm{d}\beta}{\pi} \left[\frac{x}{2l^2(l^2+x^2)} + \frac{1}{2l^3} \tan^{-1} \frac{x}{l} \right]_{-\infty}^{\infty} \\ &= \frac{\cos\beta \mathrm{d}\beta}{2} \end{aligned} \tag{62}$$

When the two strips face each other, the $\cos\beta = 1$ and $\mathrm{d}\beta$ becomes $\frac{W_2}{S}$ where W_2 is the width of $\mathrm{d}A_2$. Note that integrating over the length of the strip of $\mathrm{d}A_1$ does not change the end result

$$dF_{strip1-strip2} = \frac{W_2}{2S}. \tag{63}$$

5.8 Two Finite Strips

As shown in [80], if the two strips are made finite and of length b and parallel and of separation S

$$\mathrm{d}F_{\mathrm{d1-d2}} = \frac{\cos\phi \mathrm{d}\phi}{\pi} \tan^{-1}\left(\frac{b}{S}\right). \tag{64}$$

Note that for two strips facing each other, $\phi = 0$ and $S\mathrm{d}\phi = W_2$. Hence

$$\mathrm{d}F_{\mathrm{d1-d2}} = \frac{W_2}{S}\left(\frac{1}{\pi}\tan^{-1}\frac{b}{S}\right). \tag{65}$$

When the strips get very long, $b \to \infty$ and hence $\tan^{-1}\frac{b}{S} \to \pi/2$. Then for two infinitely long strips facing each other at a distance S :

$$\mathrm{d}F_{\mathrm{d1-d2}} = \frac{\mathrm{d}\phi}{\pi}\frac{\pi}{2} = \frac{W_2}{2S}. \tag{66}$$

In agreement with Equation (63).

5.9 Validation Using 2D Cavity

As an example, we selected the model used in the COMSOL Heat Transfer Module User's Guide which is shown in Figure 32. This 2D model illustrates the use of the surface-to-surface radiation feature. This geometry consists of three rectangles with lengths 3, 4 and 5 m respectively, and all have a width of 1 m. The rectangles are placed as such a way that they form a triangular cavity. The rectangles are made of copper, which is a good thermal conductor, and they transfer heat internally by conduction. The vertical and inclined blocks are subjected to heat fluxes $q_2^{"}$ and $q_3^{"}$ on their outer boundaries, respectively. The three inner boundaries that form the cavity can exchange heat only by means of surface-to-surface radiation.

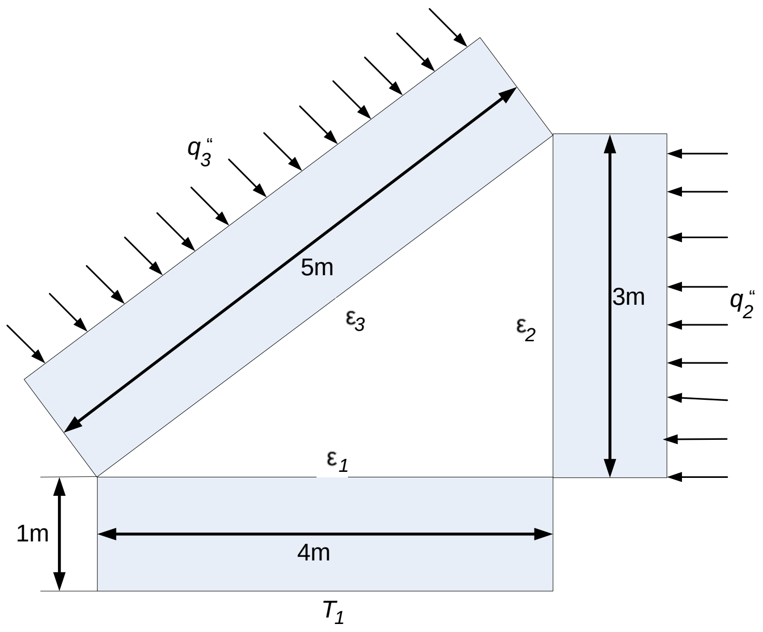


Figure 32. The completed geometry for the 2D cavity

The lowest boundary is kept at $T_1 = 300$K. The surface emissivities are $\varepsilon_1 = 0.4$ on the bottom boundary, $\varepsilon_2 = 0.6$ on the vertical boundary , and $\varepsilon_3 = 0.8$ on the inclined boundary. The system hold the temperate on the (outer) lower boundary at $T_1 = 300$K. The system experiences an inward heat flux of $q_2 = 2000$W/m^2 on the outer boundary of the vertical rectangle and $q_3 = 1000$W/m^2 on the outer boundary of the tilted one. All other outer boundaries are kept insulated.

This is both a radiation and a conduction problem. Conduction takes place inside the blocks. Surface-to-surface radiation occurs at the inner boundaries of the triangular cavity. Conditions at the outer boundaries are specified by heat fluxes, ambient temperature, and thermal insulation. A two-dimensional model setup is appropriate to solve this problem in COMSOL. In COMSOL, the inner corners of the blocks must be offset to ensure that the plates do not touch as shown in Figure 33. This prevents heat from being exchanged between the blocks by conduction. The problem can be tackled analytically by omitting conduction and considering a 3-4-5 triangle that forms the cavity. Same heat flux boundary conditions apply. The FE mesh used consisted of $6,774$ nodes, and $12,812$ triangular elements on the 2D subdomains as shown on Figure 34.

Figure 35 details the temperature distribution along the inclined boundary of the cavity. The lowest temperature appears on the bottom left, and the highest temperature is on the right, which seems reasonable because that position of the boundary is located adjacent to the vertical boundary, which has the highest inward heat flux as shown contour plot in Figure 36.

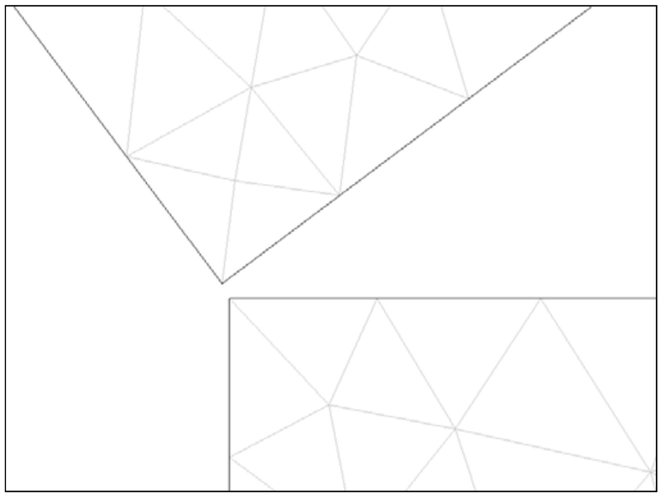


Figure 33. The blocks do not touch

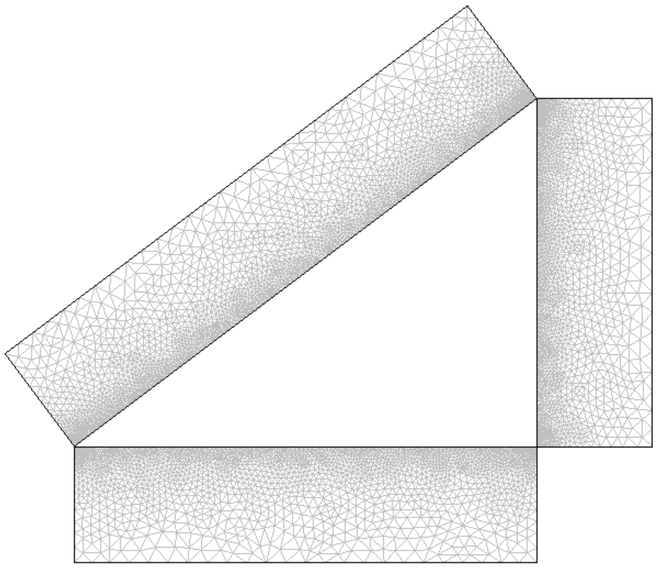


Figure 34. 2D Cavity mesh

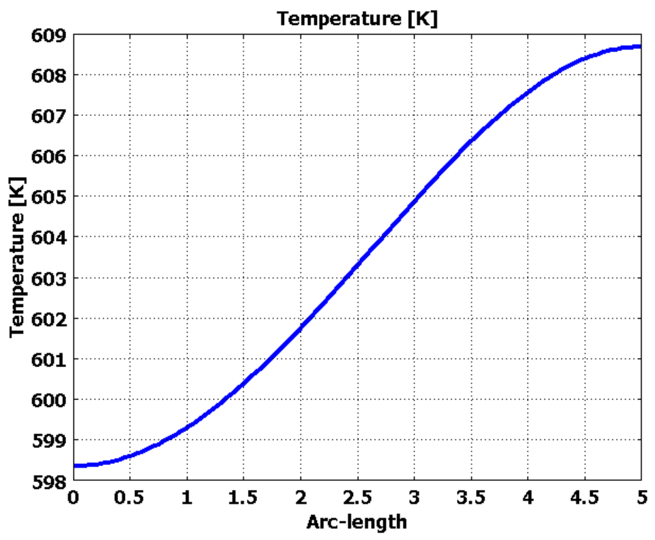


Figure 35. Temperature distribution along the inclined boundary of the cavity

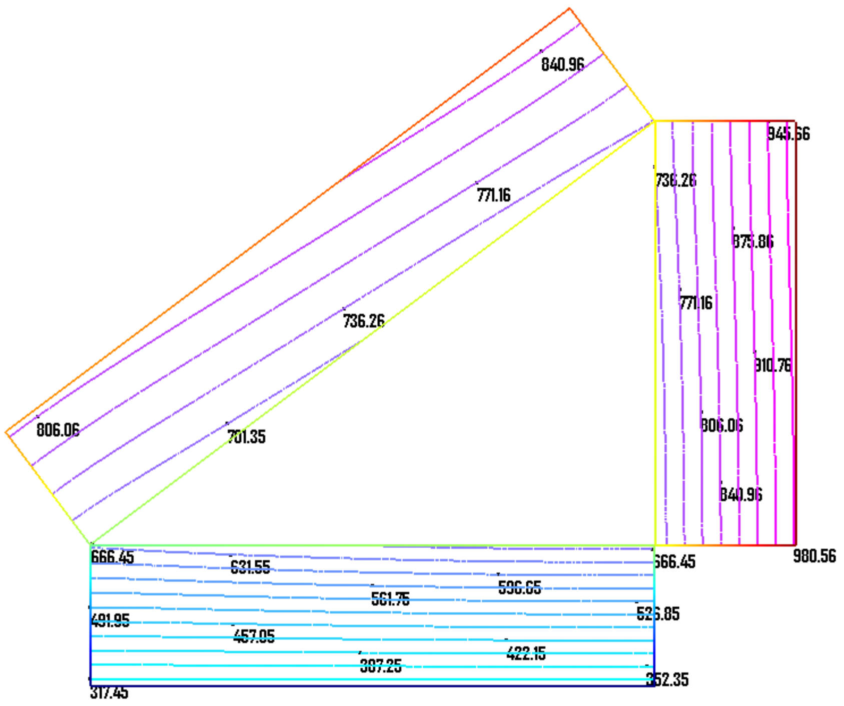


Figure 36. Contour plot of 2D cavity

Figure 37 plots the radiosity along the inclined boundary (in other words, the total heat flux that leaves the boundary into the cavity). The radiosity is the sum of the heat flux the boundary emits plus the heat flux it reflects. Like temperature, the radiosity is fairly constant along the boundary.

In general, the number of regions required for getting an accurate reduced-order model depends on the complexity of the system to be modeled. The FE mesh is decomposed by METIS into 63 regions as shown in Figure 38. The average relative error in steady-state temperature rise for the 63 regions was 17%. The FE mesh of the 2D cavity is decomposed by METIS into 255 regions as shown in Figure 39.

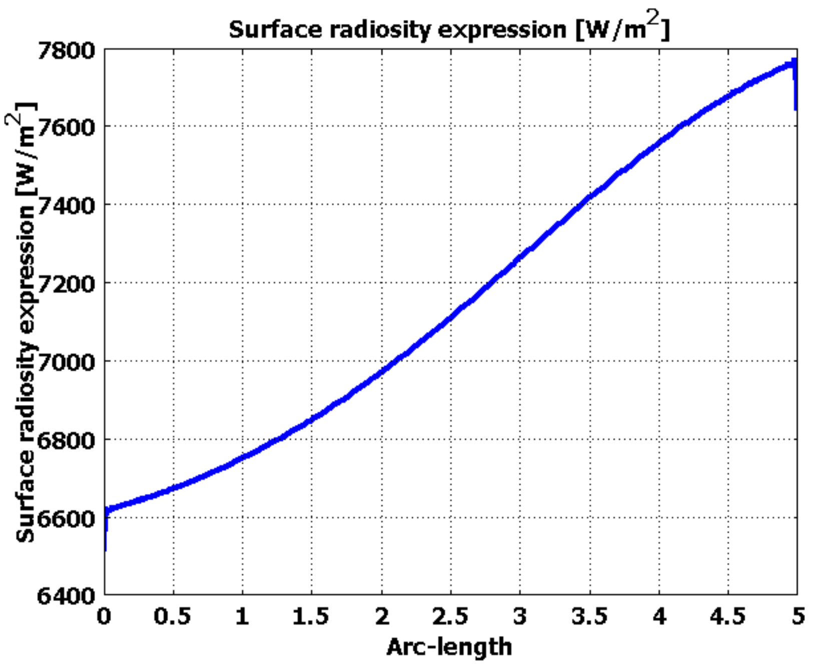


Figure 37. Radiosity along the inclined boundary of the cavity

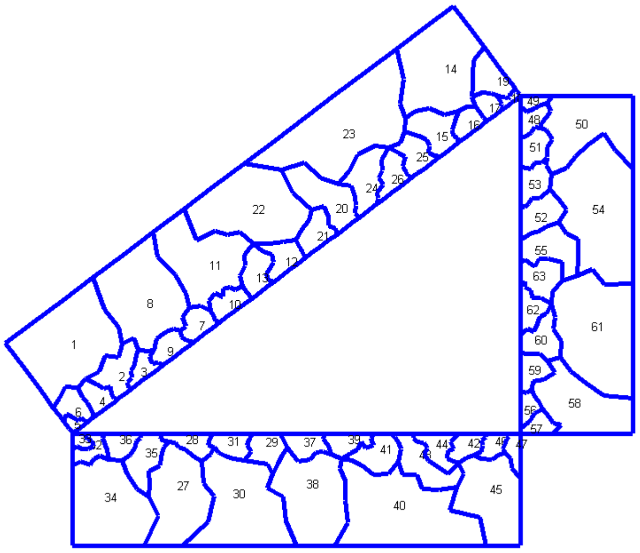


Figure 38. FE mesh is decomposed 63 regions by METIS

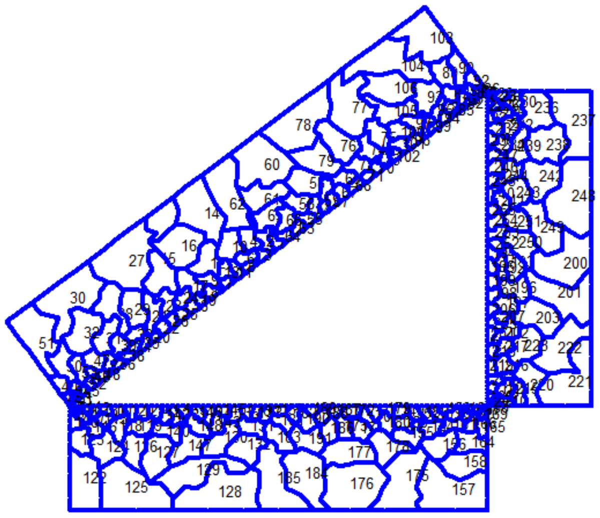


Figure 39. FE mesh is decomposed 255 regions by METIS

The part of PSPICE that is most relevant to this study is the source file that describes the circuit to be simulated or analyzed. The circuit was derived using the methodology of Chapter 2 and uses the view factor expressions in Equation (66). Hence every edge in the 2D model is assumed to represent an infinite strip in 3D. The corresponding input file for PSPICE is a text (ASCII) file that has the file type "CIR". This file includes the elements, connections the models of the elements and the type of analysis. Figure 40 shows the part of .cir source file of the 2D Cavity example.

The comparison between the steady-state behaviors of the FE model (using COMSOL) and the EC (performed in PSPICE) is shown in Table 8 for five regions/nodes. The average relative error in steady-state temperature rise for the 255 regions was 9% due to the fact that reflection was not properly accounted for.

```
R1 1 2 0.12939
R2 1 64 0.049825
R3 2 3 0.18366
...
Vso1 256 0 DC 300
Vso2 257 0 DC 300
Vso3 258 0 DC 300
...
C1 1 0 46471.7883
C2 2 0 89155.2733
...
Iso14 0 14 DC 338.9979
Iso15 0 15 DC 338.9763
...
G1 1 0 value = {5.669e-007*2.1265e-006*(V(1)^4-V(109)^4)}
G2 1 0 value = {5.669e-007*3.3197e-005*(V(1)^4-V(111)^4)}
G3 1 0 value = {5.669e-007*5.7055e-005*(V(1)^4-V(112)^4)}
...
.OP
.END
```

Figure 40. Excerpts of the circuit description using PSPICE

Table 8. Comparison of a full-model with a reduced-model at Steady-State average temperature

Number of Region = 63			**Number of Region = 255**		
Region No	**FE Result**	**EC Result**	**Region No**	**FE Result**	**EC Result**
1	769.11	672.94	1	710.99	655.25
2	706.97	608.39	2	721.73	665.24
3	693.44	605.51	3	705.98	653.56
4	692.83	603.60	4	731.82	677.79
5	678.28	602.20	5	702.81	651.40
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

CHAPTER 6
EDGE TO EDGE VISIBILITY DETERMINATION

Occlusion complicates the calculation of thermal radiation computations in 3D computer graphics, hidden surface determination, also known as hidden surface removal (HSR), occlusion culling (OC) or visible surface determination (VSD), is the process used to determine which surfaces and parts of surfaces are not visible from a certain viewpoint. Given a large model and a viewpoint, the goal of the visibility culling and hidden surface removal algorithms is to determine the set of primitives visible from that viewpoint. In this chapter, we will discuss and propose algorithms that can alleviate some of the computational aspects. We will concentrate on 2D problems.

6.1 Visible Surface Algorithms

Many visible surface determination algorithms such as Z-buffering [81], Painter's algorithm [82-83], Binary Space Partitioning (BSP) [84], ray tracing [85] and Warnock algorithm [86] have been developed over the years. They are fundamentally an exercise in sorting, and usually vary in the order in which the sort is performed and how the problem is subdivided. This section describes these well-known algorithms.

Painter's algorithm [82-83]: The painter's algorithm, also known as a priority fill, is one of the simplest solutions to the visibility problem in 3D computer graphics [87]. The painter's algorithm, sometimes called depth-sorting, gets its name from the process which an artist renders a scene using oil paints. First, the artist will paint the background colors of the sky and ground. Next, the most distant objects are painted, then the nearer objects, and so forth. Note that oil paints are basically opaque, thus each sequential layer completely obscures the layer that its covers. A very similar technique can be used for rendering objects in a three-dimensional scene. First, the list of surfaces are sorted according to their distance from the viewpoint. The objects are then painted from back-to-front. Painter's algorithm also suffers from the fact that it has a computation time that varies exponentially with the number of polygons in a scene [88].

Z-buffering [81]: The basic idea is to test the z-depth of each surface to determine the closest (visible) surface. The depth of a generated pixel (z coordinate) is stored in a buffer (the z-buffer or z-depth buffer). Declare an array z-depth buffer (x, y) with one entry for each pixel position. The Z-buffer algorithm always works and is simple to implement, but it may paint the same pixel several times, and computing the color of a pixel may be expensive with large memory requirements. The pixel is only overwritten if the depth value of the current point is less than the depth value stored in the z-buffer.

Binary Space Partitioning (BSP) [84]: A BSP tree is a hierarchical subdivision of n-dimensional space into convex subspaces. Each node has a front and a back leaf. Starting with the root node, all subsequent insertions are partitioned by the hyperplane of the current node. In two-dimensional space, a hyperplane is a line. In three-dimensional space, a hyperplane is a plane. The end goal of a BSP tree is for the hyperplanes of the leaf nodes to be trivially "behind" or "in front" of the parent hyperplane. One disadvantage of this algorithm is that hard to balance the tree.

Ray tracing [85]: This method attempts to model the path of light rays to a viewpoint by tracing rays from the viewpoint into the scene. Although this is not a hidden surface removal algorithm it implicitly solves the hidden surface removal problem by finding the nearest surface along each view-ray. Effectively this is equivalent to sorting all the geometry on a per pixel basis.

Warnock's algorithm [86]: This algorithm can be implemented both in image and object space [89]. Warnock's algorithm is a recursive area-subdivision algorithm and looks at an area of the image. It is generally easy to determine which polygons are visible in the area they are drawn. Otherwise, the area is subdivided into smaller parts and the algorithm recourses. Eventually an area will be represented by a single non-intersecting polygon. The disadvantage of this algorithm is that complex scenes usually have small polygons and high depth complexity [90].

There are also visible line determination or hidden line removal algorithms [91]. These algorithms are used mainly in the context of wireframe displays. Every visible surface determination algorithm can be modified into a visible line determination algorithm, but not vice versa. If one wants to display only visible surfaces, one must check each face against every other face to see if one obscures the other. The algorithm then cuts down the number of faces that have to be looked at by a factor of two on average, which is a substantial savings, and so visible surface determination algorithms usually have this built in as a preprocessing step because it is so easy to do so.

When looking at the object, only part of it will be visible because the portion closer to viewpoint blocks the view of the part further away. Back face elimination provides a way to remove the back part of the objects that are not visible. This step is easy to carry out and improves the efficiency of algorithms.

For a convex object, an oriented face is called a backface with respect to a vector $\vec{v}$ (typically the view direction of a camera) if the angle between its normal vector $\vec{n}$ and $\vec{v}$ is between 0 and 90 degrees. Mathematically, this simply means that the condition $\vec{n} \cdot \vec{v} \leq 0$ is necessary and sufficient for the surface to be visible. If $\vec{n} \cdot \vec{v} \geq 0$, the surface is invisible. For a non-convex object, $\vec{n} \cdot \vec{v} \leq 0$ is necessary but no longer sufficient for visibility. We will present a related algorithm in Section 6.3.

6.2 Edge-to-Edge Visibility

For radiation problems, the concept of node-to-node visibility must be updated to include the effect of partial or edge-to-edge visibility. In this section, we propose a new algorithm that characterizes edge-to-edge visibility.

To illustrate the concept of edge-to-edge visibility, we consider the two surfaces represented by uv and wz which are partly occluded by an obstacle as shown in Figure 41.

1- Let $u^{'}$ be the intersection the ray uO with the support of WZ;

2- Let $v^{'}$ be the intersection the ray vO with the support of WZ;

3- $OB^{''}$ is orthogonal to $\beta\beta^{'}$ and point away from the obstacle;

4- If $u^{'}$ is outside of $WZ,$ then WZ is totally invisible from u;

5- If $v^{'}$ is outside of WZ then WZ is fully visible from $v,$

6- Since $wv^{'}$ is totally invisible from uv, then

$$F_{uv} = 0. \tag{67}$$

7- Since $u^{'}z$ is totally visible from uv, then the corresponding view factor is

$$F_{u^{'}z} = \frac{1}{\overline{uv}} \int_{uv} \int_{u^{'}z} \frac{\cos\theta_1 \cos\theta_2}{2R_{12}} \mathrm{d}L_1 \mathrm{d}L_2 \tag{68}$$

where $\mathrm{d}L_1 \subset uv$, $\mathrm{d}L_2 \subset u'z$ and R_{12} = Distance between two edges. We have assumed the 2 infinte strip formula (see Equation 63) but other formulas seek as the 2 finite strips formula can also be used.

8- Since $v^{'}u^{'}$ is partially visible to uv, then

$$F = \frac{1}{\overline{uv}} \int_{uv} \int_{u'v'} \frac{\cos\theta_1 \cos\theta_2}{2R_{12}} s \mathrm{d}L_1 \mathrm{d}L_2 \tag{69}$$

where $L_1 \subset uv$, $L_2 \subset u'v'$ and

$$s = \left\{ \begin{array}{ll} 0 & \text{if } \overrightarrow{O\delta} \cdot \overrightarrow{O\beta^{''}} \leq 0 \\ 1 & \text{if } \overrightarrow{O\delta} \cdot \overrightarrow{O\beta^{''}} \geq 0 \end{array} \right\}.$$

To illustrate how partial visibility can be used to model thermal radiation problems, we consider three polygons as shown in Figure 42. The view factors from u_1u_3 to w_1w_4 are illustrated below:

1- $u_1^1 w_1$ is fully visible from u_1u_3 and one must use Equation (68).

2- $w_4 u_1^1$ is partially visible from u_1u_3 and one must use Equation (69).

3- $u_3^2 w_4$ is partially visible from u_1u_3 and one must use Equation (69)

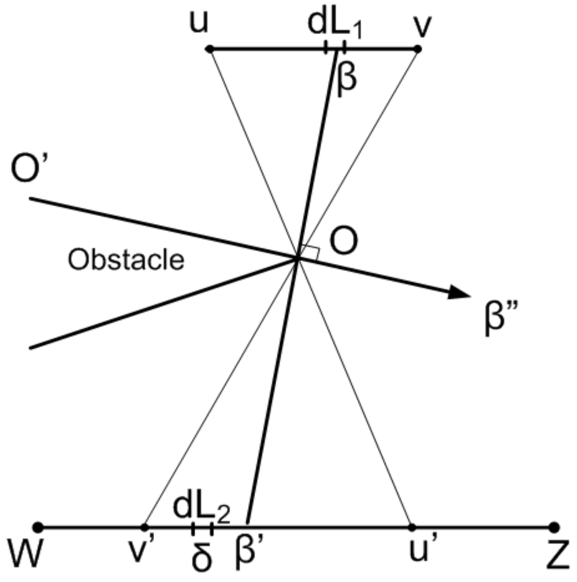


Figure 41. Edge-to-edge occlusion

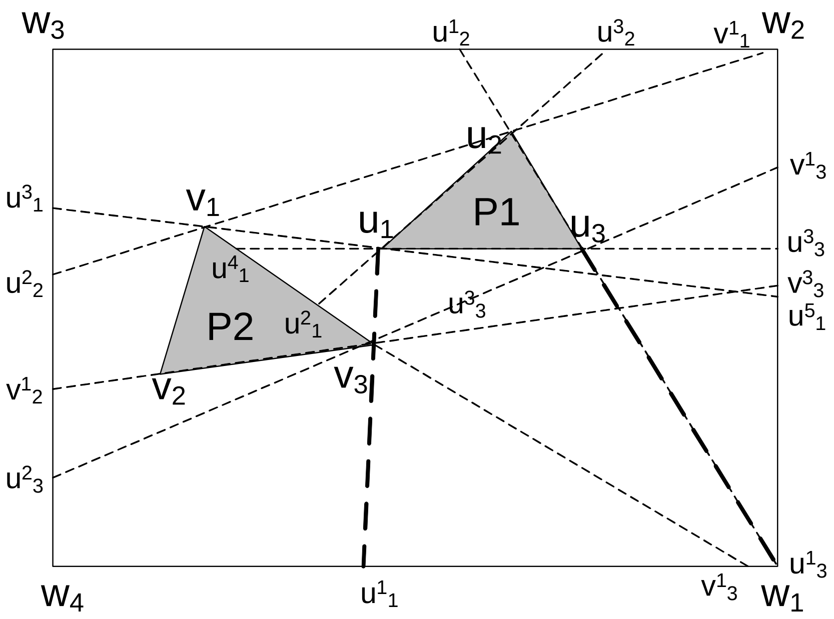


Figure 42. Complexity of the edge-to-edge visibility

The edges of polygon 1 are u_1u_3, u_1u_2 and u_2u_3. The edge visibility from edges u_1u_3, u_1u_2 and u_2u_3 are shown in Table 9 where 'P' indicates 'partially visible' edges and 'F' indicates 'fully visible' edges. The expressions to calculate current and view factors are shown in the Table. The corresponding equivalent-circuit for polygon 1 is shown in Figure 43. The radiation boundary conditions represented by 12 voltage-controlled current sources. The expression for each current is given in Table 9. The view factors are calculated with Equations (67), (68) or (69). The voltage at node 1 of the equivalent circuit represents a temperature distribution T_1 over a region in the geometry decomposition. The heat source in the region is represented by an independent current source. Capacitance represents the thermal capacitance of the region and is directly proportional to the volume of the region. Figure 44 shows the part of .cir source file of the equivalent-circuit.

Table 9. Visibility of edges from edges u1u3, u1u2, and u2u3 on Polygon 1

	Edge	**Edge**	**P/F**	**Current**	**Equation Number for View Factor**
1	u_1u_3	$u_1^1w_1$	F	$I_1 = u_1^1w_1F_{u_1u_3 \longrightarrow u_1^1w_1}\sigma(T_{u_1u_3}^4 - T_{u_1^1w_1}^4)$	Eq. 68
2		$u_1^1w_4$	P	$I_2 = u_1^1w_4F_{u_1u_3 \longrightarrow u_1^1w_4}\sigma(T_{u_1u_3}^4 - T_{u_1^1w_4}^4)$	Eq. 69
3		$u_3^1w_1$	F	$I_3 = u_3^1w_1F_{u_1u_3 \longrightarrow u_3^1w_1}\sigma(T_{u_1u_3}^4 - T_{u_3^1w_1}^4)$	Eq. 68
4		$u_1^4v_3$	F	$I_4 = u_1^4v_3F_{u_1u_3 \longrightarrow u_1^4v_3}\sigma(T_{u_1u_3}^4 - T_{u_1^4v_3}^4)$	Eq. 68
5		$u_3^2w_4$	P	$I_5 = u_3^2w_4F_{u_1u_3 \longrightarrow u_3^2w_4}\sigma(T_{u_1u_3}^4 - T_{u_3^2w_4}^4)$	Eq. 69
6		$u_3^1u_3^3$	F	$I_6 = u_3^1u_3^3F_{u_1u_3 \longrightarrow u_3^1u_3^3}\sigma(T_{u_1u_3}^4 - T_{u_3^1u_3^3}^4)$	Eq. 68
7	u_1u_2	$u_1^2v_1$	F	$I_7 = u_1^2v_1F_{u_1u_2 \longrightarrow u_1^2v_1}\sigma(T_{u_1u_2}^4 - T_{u_1^2v_1}^4)$	Eq. 68
8		$u_2^3w_3$	F	$I_8 = u_2^3w_3F_{u_1u_2 \longrightarrow u_2^3w_3}\sigma(T_{u_1u_2}^4 - T_{u_2^3w_3}^4)$	Eq. 68
9		$u_2^2u_1^3$	P	$I_9 = u_2^2u_1^3F_{u_1u_2 \longrightarrow u_2^2u_1^3}\sigma(T_{u_1u_2}^4 - T_{u_2^2u_1^3}^4)$	Eq. 69
10		$u_1^3w_3$	F	$I_{10} = u_1^3w_3F_{u_1u_2 \longrightarrow u_1^3w_3}\sigma(T_{u_1u_2}^4 - T_{u_1^3w_3}^4)$	Eq. 68
11	u_2u_3	$u_2^1w_2$	F	$I_{11} = u_2^1w_2F_{u_2u_3 \longrightarrow u_2^1w_2}\sigma(T_{u_2u_3}^4 - T_{u_2^1w_2}^4)$	Eq. 68
12		$u_3^1w_2$	F	$I_{12} = u_3^1w_2F_{u_2u_3 \longrightarrow u_3^1w_2}\sigma(T_{u_2u_3}^4 - T_{u_3^1w_2}^4)$	Eq. 68

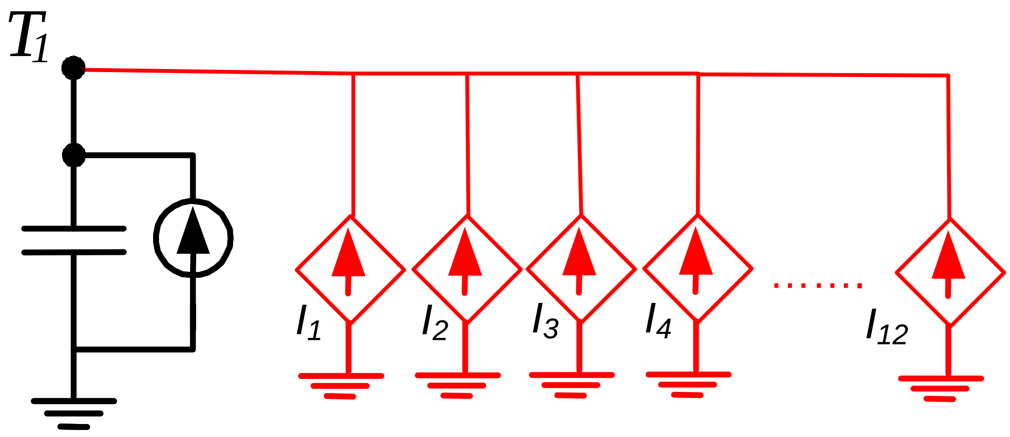
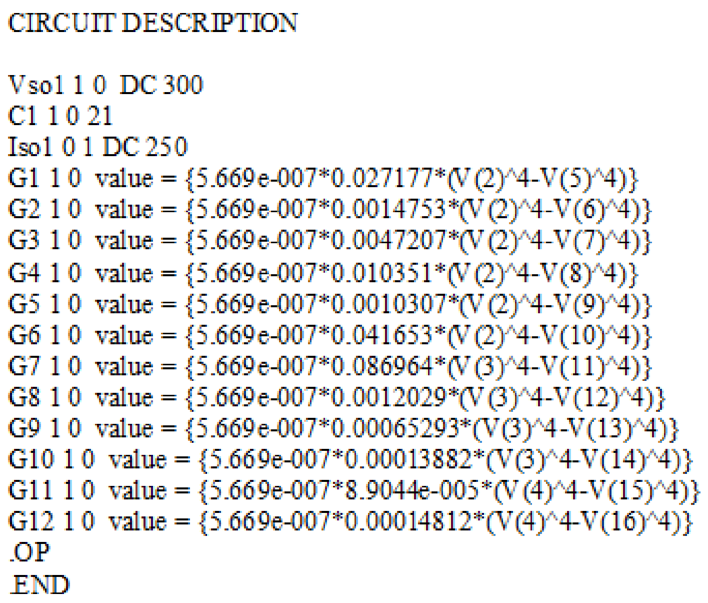


Figure 43. Equivalent-Circuit corresponding for Polygon 1 in Figure 42



```
CIRCUIT DESCRIPTION

Vso1 1 0  DC 300
C1  1 0 21
Iso1  0 1 DC 250
G1  1 0  value = {5.669e-007*0.027177*(V(2)^4-V(5)^4)}
G2  1 0  value = {5.669e-007*0.0014753*(V(2)^4-V(6)^4)}
G3  1 0  value = {5.669e-007*0.0047207*(V(2)^4-V(7)^4)}
G4  1 0  value = {5.669e-007*0.010351*(V(2)^4-V(8)^4)}
G5  1 0  value = {5.669e-007*0.0010307*(V(2)^4-V(9)^4)}
G6  1 0  value = {5.669e-007*0.041653*(V(2)^4-V(10)^4)}
G7  1 0  value = {5.669e-007*0.086964*(V(3)^4-V(11)^4)}
G8  1 0  value = {5.669e-007*0.0012029*(V(3)^4-V(12)^4)}
G9  1 0  value = {5.669e-007*0.00065293*(V(3)^4-V(13)^4)}
G10 1 0  value = {5.669e-007*0.00013882*(V(3)^4-V(14)^4)}
G11 1 0  value = {5.669e-007*8.9044e-005*(V(4)^4-V(15)^4)}
G12 1 0  value = {5.669e-007*0.00014812*(V(4)^4-V(16)^4)}
.OP
.END
```

Figure 44. Equivalent-Circuit description using PSPICE

6.3 Edge-to-Edge Visibility Determination

The algorithm we developed in this dissertation is partly inspired by the node-to-node visibility algorithm explained in Appendix.

6.3.1 Sorting Node Angles

The first step of the edge-to-edge visibility algorithm is to find the angle of the nodes with respect to a selected view point. The view point is considered in the center of the coordinate system. We need a function to sort the vertices of the polygon in an anticlockwise order around the view point. There are two inputs for the function: (a) `xyViewPoint` denoting the coordinates of the view point; and (b) the coordinates of a node `xyNode` which we are trying to find the angle. The pseudocode for algorithm `SortPolygonVertices` is shown in Figure 45.

```
Algorithm A = SortPolygonVertices(xyViewPoint, xyNode)

PURPOSE: The function sorts the vertices of the polygon in an anticlockwise order.
INPUTS: xyViewPoint: coordinates of the viewpoint
        xyNode: coordinates of the node

1. P.x = xyNode(1)-xyViewPoint(1)
2. P.y = xyNode(2)-xyViewPoint(2)
3. IF P.x>0
        a. IF P.y>0 angle = atan((P.y)/(P.x))
        b. ELSE angle = atan((P.y)/(P.x))+2*π
4. IF P.x<0
        a. IF P.y>0 angle = atan((P.y)/(P.x))+ π
        b. ELSE angle = atan((P.y)/(P.x))+ π
5. IF P.x==0
        a. IF P.y>0 angle = (π/2)
        b. ELSEIF P.y<0 angle = (3*(π/2))
        c. ELSE angle = 0
6. A = (angle*180)/π
```

Figure 45. Pseudocode summarizing the `SortPolygonVertices` algorithm

6.3.2 Self Visibility

Self-visibility arises for nonconvex obstacles where the viewpoint belongs to the obstacle and one part of the obstacle occludes another part.

Figure 46 shows 4 examples for self-visibility. Thick-black-solid lines show the visible edges from the view point and thin-black lines are the invisible edges. Figure 46(a) and (b) show fully visible edges. Figure 46(c) shows a partially visible edge. Some part of the edge can not be seen because one or more vertices of the same polygon is blocking. Figure 46(d) has no self-visibility. The pseudocode for algorithm `SelfVisible` is shown in Figure 47.

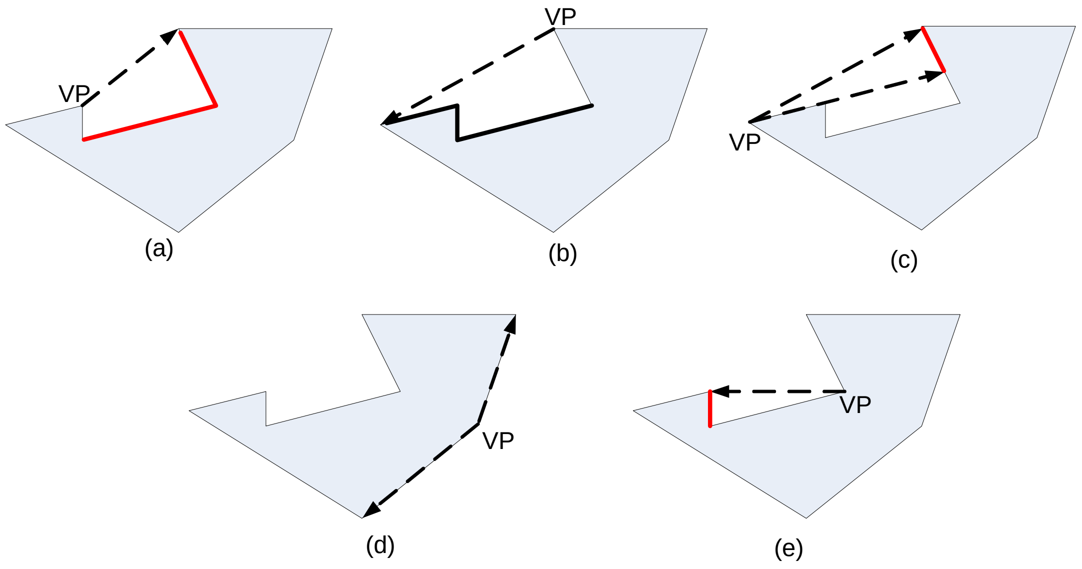


Figure 46. Example of self-visibility: (a), (b) and (e) Fully visible edges, (c) partially visible edge, (d) invisible edges

```
Algorithm [Polygon, visible] = SelfVisible (ViewPoint, Polygon)

PURPOSE: The function finds if one part of the polygon occludes another part or not.
INPUTS:  ViewPoint: name of the view point
         Polygon: This structure includes information such as vertices, edges etc. for the
         Polygon.

1. CALL WalkAroundPolygon(Polygon)
2. Find the ordered nodes of the Polygon
3. FOR every node iN
      a. Define a vector l between node iN to the ViewPoint
      b. Test if vector l intersects polygons
      c. IF it intersects , mark node iN as visible
      d. ELSE mark the node iN as invisible
      e. Find the edges of ViewPoint belongs
4. Find the nodes adjacent to ViewPoint
5. FOR every edge iE in obstacle
      a. Find the nodes of the edge iE
      b. IF Both nodes are visible, Mark the edge iE as visible,
      c. ELSE IF Both nodes are invisible, Mark the edge iE as invisible,
      d. ELSE IF One node is visible and the other node is invisible,
            i. Decide visible and invisible nodes from ViewPoint
           ii. IF the invisible node is one of the adjacent nodes, Mark the edge iE as visible
          iii. ELSE CALL function DecideWay (Denotes front edge by iF, back edge by
               iB),
           iv. Mark the edge iB as invisible
            v. IF the edge iF is never marked before
                  1. Find nodes of the edge iF
                  2. Mark the edge iF as partially visible
                  3. CALL function LinesIntersectionOutside to find the
                     intersection point on the edge iF
                  4. Update the visible and invisible coordinates of the edge
           vi. END IF
      e. END IF
   6. ENF FOR
```

Figure 47. Pseudocode summarizing the `SelfVisible` algorithm

6.3.3 Occlusion by External Obstacles

When a solid polygon is viewed from a viewpoint, some edges are necessarily occluded. External occlusions are caused by another polygon intervening between the view point and the edge of interest.

The function `OutsideOccluded` handles external occlusions and its pseudocode is shown in Figure 48. It accepts 2 inputs. (a) the `ViewPoint` and (b) the `Polygon` is a structure which includes information such as vertices, edges etc. for the polygon. This function calls three functions `DecideWay`, `WalkBackEdges`, `WalkFrontEdges`. The algorithm starts with finding the minimum and maximum angles of the polygon. The node with minimum angle is assigned as a `StartingNode` and the node with maximum angle is assigned as `EndNode`. The pseudocode for algorithm `OutsideOccluded` is shown in Figure 48.

```
Algorithm Polygon = OutsideOccluded (ViewPoint, Polygon)

PURPOSE:  The function marks visible, partially visible and invisible edges by calling 3
           functions.
INPUTS:   ViewPoint: name of the view point
          Polygon: This structure includes information such as vertices, edges etc.
          for the  Polygon.

1.  Find the minimum and maximum angles of the polygon
2.  Assign StartingNode = index of minimum angle
3.  Assign EndNode = index of maximum angle
4.  Initialize all the edges as not walked
5.  CALL function DecideWay
6.  CALL function WalkBackEdges
7.  CALL function WalkFrontEdges
```

Figure 48. Pseudocode summarizing the `OutsideOccluded` algorithm

Next, one must determine which of the two edges incident on `StartNode` is behind the obstacle as seen from `ViewPoint`. This is achieved using the function `DecideWay` whose pseudocode is shown in Figure 49. We find angles θ_i and θ_j between vector l (from viewpoint to starting node) and edge i and j respectively. The edge with a smaller angle is assigned as a back edge, and remaining edge assigned as a front edge.

Figure 50 shows an example for deciding the front edge and back edge. Node 3 in polygon 1 is selected as the view point. After deciding the front and back edges, we start walking on back side of the polygon. All the back edges are marked as invisible. The pseudocode for algorithm `WalkBackEdges` is shown in Figure 51.

```
Algorithm [BackEdge FrontEdge]=DecideWay(StartingNode,Polygon,
                                          ViewPoint)

PURPOSE: The function determines which of the two edges incident to StartingNode
          is behind the obstacle as seen from the ViewPoint.
INPUTS:  StartingNode: This is the minimum angle node of the obstacle.
         Polygon: This structure includes information such as vertices, edges etc.
         for the Polygon.
         ViewPoint : Name of the view point.

  1. Find the edges of the StartingNode
  2. Define the vector l from a ViewPoint to a StartingNode
  3. Find θ1 between vector l and edge i
  4. Find θ2 between vector l and edge j
  5. IF θ1 < θ2 BackEdge = i, FrontEdge = j
  6. ELSE BackEdge = j, FrontEdge = i
```

Figure 49. Pseudocode summarizing the `DecideWay` algorithm

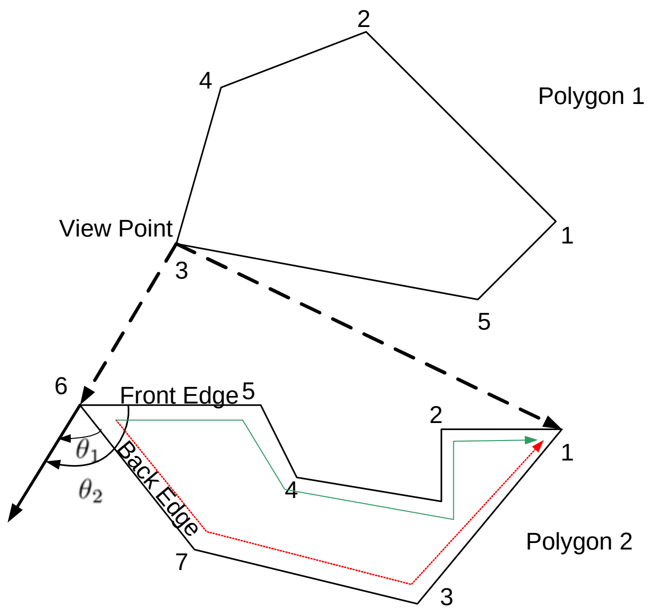


Figure 50. Deciding a front edge and back edge by calculating the angles

```
Algorithm Polygon=WalkBackEdges(NextNode, BackEdge, EndNode)

PURPOSE:  The function walks back of the obstacle and mark the edges invisible.
INPUTS:  NextNode: The other end of the BackEdge
         BackEdge:  Name of the BackEdge
         EndNode:  This is the maximum angle node of the polygon

1.  NextNode=the other node of BackEdge
2.  WHILE NextNode  is not equal to EndNode
    a.  Find next edge NextEdge incident on NextNode
    b.  Mark edge invisible
    c.  Let NextNode  =See other node of NextEdge
```

Figure 51. Pseudocode summarizing the `WalkBackEdges` algorithm

The most difficult part of this algorithm is to walk front edges of the polygon and decide which edges are partially or fully visible. The pseudocode for algorithm `Walk-FrontEdges` is shown in Figure 52. This function return a structure which holds information about Polygon. Such as `Polygon.Edge.visible`, `Polygon.Vertex.xy` etc. Referring to Figure 53, some part of edge $\overline{st}$ is occluded by $\overline{xy}$. In this example, only one vertex is blocking the edge partially. The function begins by walking from a vertex which has the minimum angle in the obstacle. Each time when it passes a node or edge, they are stored in sequence in a vector to use them for future decisions.

In Figure 53, the nodes are walked through in the order y, x, t, s where

$$angle(y) < angle(x) \tag{70}$$

$$angle(x) > angle(t) \tag{71}$$

$$angle(t) < angle(s) \tag{72}$$

Since the function is attempting to move consistently from a minimum angle to a maximum angle, Equation (71) will flag a self-occlusion situation. When this occurs the first intersection $x^{'}$ of the ray Ox with the obstacle must be determined and stored. Obvisously all edges from x to $x^{'}$ are invisible except perhaps the edges containing $x^{'}$. The algorithm subsequently walks the boundary of the obstacle backwards from $x^{'}$ towards x and all found edges are marked as invisible.

```
Algorithm Polygon=WalkFrontEdges(StartingNode, NextNode, FrontEdge
EndNode)

PURPOSE:  The function walks the front of the obstacle and marks the edges invisible.
INPUTS:  StartingNode :  Starting node of the FrontEdge
         NextNode: The other end of the FrontEdge
         FrontEdge: Name of the FrontEdge
         EndNode:  This is the maximum angle node of the polygon

1. NextNode = the other end of FrontEdge
2. Find the angle of StartingNode (Denotes angle by AStartingNode)
3. Find the angle of NextNode (Denotes angle by ANextNode)
4. Find the sign of the FrontEdge as signStarting=AStartingNode-ANextNode
5. WHILE NextNode not equal to EndNode
   a. Find NextEdge incident on NextNode
   b. FutureNode = the other end of NextEdge
   c. Find the angle of FutureNode (Denotes angle by AFutureNode)
   d. Find the sign of the FrontEdge as signNextEdge=ANextNode-
      AFutureNode
   e. IF signStarting is not equal to signNextEdge
        i. Walking backward so mark NextEdge as invisible
       ii. Add blocking node inside BlockingNode Vector (Keep track of blocking
           node)
   f. ELSE
        i. Find is there any node between NextNode and FutureNode by looking the
           angles
       ii. IF there is a node
             1. IF node is inside the BlockingNode vector
             2. Find the intersection point x'
             3. Mark the NextEdge partially visible
      iii. ELSE Mark the edge fully visible
       iv. END IF
   g. ENDIF
   h. Let NextNode = FutureEdge
6. END WHILE
```

Figure 52. Pseudocode summarizing the `WalkFrontEdges` algorithm

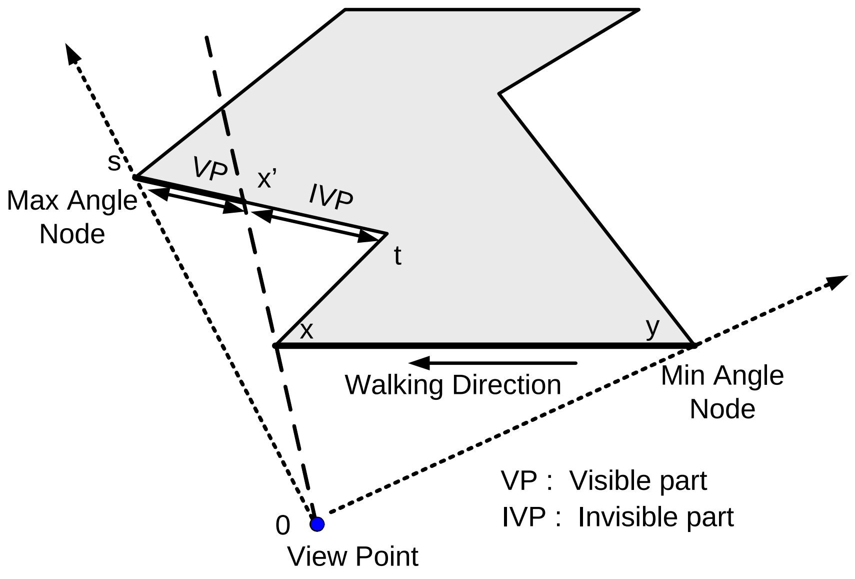


Figure 53. Edge $\overline{st}$ is bounded by extension edge of $\overline{xy}$

Figure 54 can be analyzed/generated by using the algorithms explained above. Each polygon includes convex or/and concave vertices. The first 4 polygons (from polygon 1 to 4) are surrounded by the 5^{th} polygon. The view point for this example is selected in polygon 4 as seen in Figure 54.

The analysis proceeds as follows (briefly):

1- The front and back of Polygon 4 (as seen from ViewPoint) are determined using `WalkFrontEdges` and `WalkBackEdges`.

2- Polygon 3 is easily determined to be totally invisible because all the nodes in Polygon 3 have angles between those of nodes 1 and 7 of the Polygon 4.

3- The front and back edges of Polygon 1 are easily determined using `DecideWay` function.

4- The ranges of angles of Polygons 1 and 2 intersect. Hence the tip of Polygon 1 must occlude part of Polygon 2.

For an arbitrary vertex n as shown in Figure 55, one must find the fully visible and invisible edges by the pseudocode given as follow:

1- Find all edges e1, e2,. . . ,em that are fully or partially visible from n

2- If e1 is partially visible from n find all visible points P1,. . . , Pk on e1 such that nPi passes through a obstacle vertex qi

3- Labels all subedges fully visible to n

We can conclude that $P_1'P_2$ =Fully Visible, $P_2'P_3$ =Invisible, $P_3'P_4$ =Fully Visible, $P_4'P_e$ =Invisible, P_0P_1 =Invisible.

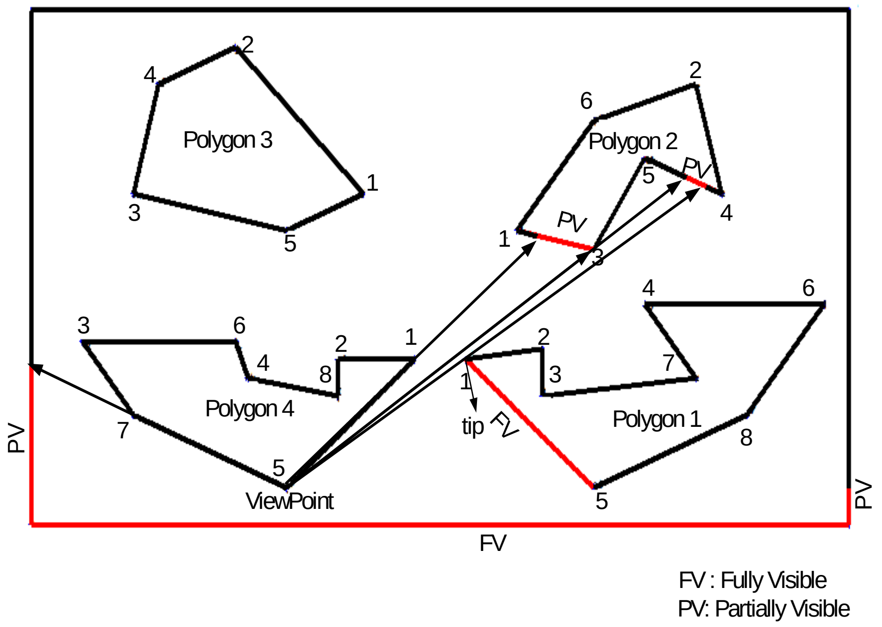


Figure 54. Node 5 on Polygon 4 is selected as view point

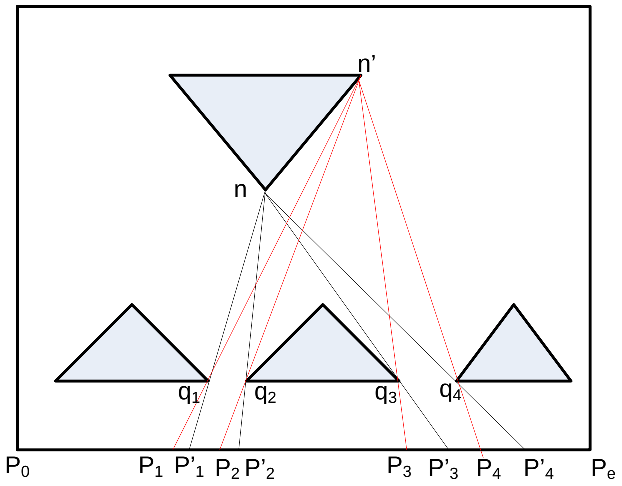


Figure 55. Fully visible and invisible edges

We need to decide partially visible edges. As shown in Figure 56, vertex n can see WW' and vertex n' can see ZZ'. If WW' is visible to n, ZZ' is visible to n' than $qq' = WW' \cap ZZ'$ is fully visible and $(WW' \cup ZZ') - (qq')$ is partially visible.

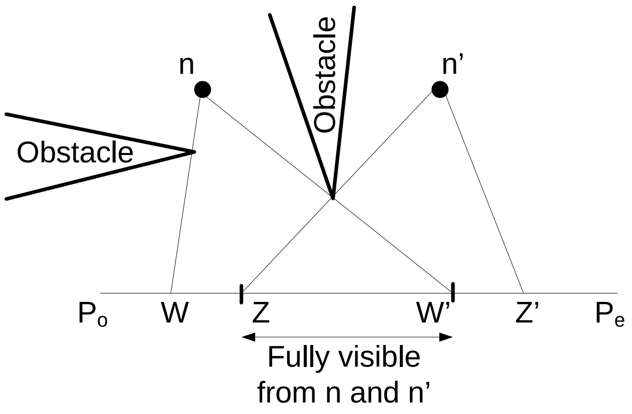


Figure 56. Fully visible and partially visible edges from two vertices

CHAPTER 7
CONCLUSIONS

We established a general automatic and versatile procedure to derive an equivalent circuit for a thermal system using temperature data obtained from FE simulations. The EC topology was deduced from the FE mesh using a robust and general graph-partitioning algorithm. The method was shown to yield models that are independent of the boundary conditions for complicated 3D thermal systems such as an electronic chip. The results are strongly correlated with the geometry, and the EC can be extended to yield variable medium-order EC models. Moreover, a variety of heat sources and boundary conditions can be accommodated, and the EC models are inherently modular. A reliable method to compute thermal resistors connecting different regions was developed. It appropriately averages several estimates of a thermal resistance where each estimate is obtained using simulation data obtained under different boundary or heating conditions. The concept of fictitious heat sources was used to increase the number of simulation datasets.

A series of validation tests were conducted using three thermal models. The average percentage error in steady-state temperature average over a region was 0.2375% for the 3D rectangular slab. For the motor pole, the average relative temperature rise error was 0.2% at steady-state analysis, while for the electronic chip it was 7%. The transient errors were dominated by the steady-state errors. The models had good agreement for transient analysis as well.

We have also implemented a view factor based radiative heat transfer model by including voltage-controlled current sources in the equivalent circuit. A computational-geometry algorithm was developed to calculate the view factors in 2D including the effect of partial edge visibility due to occlusion. The method was shown to yield models that are independent of the BCs for complicated 2-D thermal systems such as a 2D cavity. A reliable method to compute thermal resistors connecting different regions was developed. In general, the number of regions required for getting an accurate reduced-order model depends on the complexity of the system to be modeled. The FE mesh is decomposed by METIS into 63 and 255 regions. The average relative error in steady-state temperature rise for the 63 regions was 17% and 9% for the 255 regions. We can see clearly that increasing the number of regions increases the accuracy of the equivalent circuit. For complex geometries decomposed into a large number of regions, one needs a systematic procedure to generate measurements datasets that can be used to estimate all thermal resistances accurately.

We implemented the rotational plane sweep algorithm [92] as shown in Appendix. This algorithm only defines vertex-to-vertex visibility graphs. We implemented the edge-to-edge visibility graph for a 2D dimensional polygons. These polygons can both be concave and convex where the most important difference from other approaches. Self-visibility arises for nonconvex polygons where the viewpoint belongs to the polygon and one part of the polygon occludes another part. These proposed algorithms provide an excellent, versatile, and powerful approach that can be applied more complex geometries.

This research can be extended in several directions. For instance, it can be even more automated, and made to calculate the transient errors for radiation heat transfer. In 2D space, we say that two points are mutually visible or see each other if there is a straight line not intersecting any other part of the configuration from one object to the other. If we want to use this concept in 3D, instead of using the line, we must use the plane. In the future, the author is planning to extend the algorithm to 3D and test on 3D radiation heat transfer examples.

BIBLIOGRAPHY

[1] P. E. Bagnoli, C. Casarosa, M. Ciampi, E. Dallago, "Thermal resistance analysis by induced transient (TRAIT) method for power electronic devices thermal characterization. I. Fundamentals and theory," *IEEE Trans. Power Electronics*, vol. 13, pp. 1208-1219, November 1998.

[2] S. Clemente, "Transient thermal response of power semiconductors to short power pulses," *IEEE Trans. Power Electronics*, vol. 8, pp. 337-341, October 1993.

[3] A. R. Hefner, D. L. Blackburn, "Simulating the dynamic electrothermal behavior of power electronic circuits and systems," *IEEE Trans. Power Electronics*, vol. 8, pp. 376–385, October 1993.

[4] J. Garcia, G. D. Tanguy, C. Rombaut, "Modeling and simulating the dynamic electrothermal behavior of power electronic circuits using bond graphs," *27th Annual IEEE Power Electronics Specialists Conf.*, 1996, pp. 1641–1647.

[5] K. Gorecki, J. Zarebski, "The Electrothermal Analysis of a Switched Mode Voltage Regulator," *Mixed Design of Integrated Circuits and Systems, 14th International Conf.*, 2007, pp. 597–602.

[6] J. K. John, J. S. Hu, S. G. Ziavras, "Optimizing the thermal behavior of subarrayed data caches," *IEEE International Conf. on Computer Design: VLSI in Computers and Processors*, 2005, pp. 625–630.

[7] Y. Chen, C. Chen, Q. Dong, R.W. Stephenson, "Thermal management for motor," *Thermal and Thermomechanical Phenomena in Electronic Systems Conf.*, 2002, pp. 545-551.

[8] O. A. Kabov, "Heat transfer in cooling systems of microelectronic equipment with partially submerged condensers, " *IEEE Trans. Components, Packaging, and Manufacturing Technology*, Part A, vol. 19, pp. 157–162, June 1996.

[9] A. F. Armor, M. V. K. Chari, "Heat flow in the stator core of large turbine-generators, by the method of three-dimensional finite elements part II: Temperature distribution in the stator iron," *IEEE Trans. Power Apparatus and Systems*, vol. 95, pp. 1657–668, September 1976.

[10] D. V. Malyna, E. C. W. De Jong, J. A. Ferreira, M. A. M. Hendrix, J. L. Duarte, P. Bauer, A. J. A. Vandenput, "Combined electrical and thermal modeling in power supplies," *Power Electronics and Applications Conf.*, 2005, pp. 1-10.

[11] J. Galloway, S. Shidore, "Implementing compact thermal models under non-symmetric trace routing conditions," *IEEE Semiconductor Thermal Measurement and Management*

Symposium, 2004, pp. 255–261.

[12] Y. Tal, "An improved two-resistors compact thermal model by means of modified top-surface-area," *IEEE Semiconductor Thermal Measurement and Management Symposium*, 2002, pp. 64–70.

[13] H. Rosten, J. Parry, C. J. M. Lasance et al, "Final report to SEMI-THERM XIII on the European-funded project DELPHI - the development of libraries and physical models for an integrated design environment," *in Proc. of the Thirteenth IEEE SEMI-THERM Symposium*, 1997, pp. 73-91.

[14] http://electronics-cooling.com/articles/2007/may/a2/.

[15] L. T. Pillage, R. A. Rohrer, "Asymptotic waveform evaluation for timing analysis," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, pp. 352–366, April 1990.

[16] L. M. Silveira, M. Kamon, I. Elfadel, J. White, "A coordinate-transformed Arnoldi algorithm for generating guaranteed stable reduced-order models of RLC circuits," *Computer-Aided Design ACM International Conf.*, 1996, pp. 288–294.

[17] I. M. Elfadel, D. D. Ling, "Zeros And Passivity Of Arnoldi-reduced-order Models For Interconnect Networks," *in Proc. Design Automation Conf.*, 1997, pp. 28–33.

[18] J. A. Gibson, M. A. Hamilton-Jenkins, "Transfer-function models of sampled systems," *IEEE Proc. Electronic Circuits and Systems*, 1983, pp. 37–44.

[19] R. W. Freund, "Passive reduced-order models for interconnect simulation and their computation via Krylov-subspace algorithms," *in Proc. Design Automation Conf.*, 1999, pp. 195 - 200.

[20] A. R. Hefner, D. L. Blackburn, "Thermal component models for electrothermal network simulations," *IEEE Trans. on Components, Packaging, and Manufacturing Technology*, vol. 17, pp. 413-424, September 1994.

[21] M. Cooper, "Validation of SABER temperature measurements using ground-based instruments," *IEEE Proc. Geoscience and Remote Sensing Symposium*, 2004, pp. 4099-4101.

[22] H. A. Mantooth, M. Vlach, "Beyond SPICE with Saber and MAST," *IEEE Proc. Circuits and Systems*, 1992, pp. 77-80.

[23] D. Celo, G. Xiaoming, P. K. Gunupudi, R. Khazaha, D. J. Walkey, T. Smy, M. S. Nakhla, "The creation of compact thermal models of electronic components using model reduction," *IEEE Transactions on Advanced Packaging*, vol 28, pp. 240-251, May 2005.

[24] P. Gunupudi, M. Nakhla, "Multi-dimensional model reduction of VLSI interconnects," *IEEE Proc. Custom Integrated Circuits Conf.*, 2000, pp. 499-502.

[25] L. Daniel, O. C. Siong, S. C. Low, K. H. Lee, J. K. White, "A multiparameter moment matching model reduction approach for generating geometrically parameterized interconnect performance models," *Trans. on Computer-Aided Design of Integrated Circuits*, vol. 23 pp. 678-693, May 2004.

[26] J. R. Phillips, "Variational interconnect analysis via PMTBR," in *Proc. of IEEE/ACM International Conf. on Computer Aided-Design*, 2004, pp. 872-879.

[27] K. C. Suo, A. Megretski, L. Daniel, "A quasi-convex optimization approach to parameterized model order reduction," in *Proc. of the IEEE/ACM Design Automation Conf.*, 2005, pp. 933-938.

[28] B. Bond, L. Daniel, "Parameterized model order reduction of nonlinear dynamical systems," *IEEE/ACM International Conf. Computer-Aided Design*, 2005, pp. 487-494.

[29] C. H. Tsai, S. M. Kang, "Fast temperature calculation for transient electrothermal simulation by mixed frequency/time domain thermal model reduction," in *Design Automation Conf.*, 2000, pp. 750-755.

[30] R. L. Linton, D. Agonafer, "Coarse & Detailed CFD Modeling of a Finned Heat Sink," in *Proc. Thermal Phenomena in Electronic Systems Conf.*, 1994, pp. 156-161.

[31] S. Narasimhan, J. Majdalani, "Caracterization of compact heat sink models in natural convection," *IEEE Transections on Components and Packaging Technologies*, vol. 25, pp. 78-86, March 2002.

[32] U. Drofenik, J. W. Kolar, "A Thermal Model of a Forced-Cooled Heat Sink for Transient Temperature Calculations Employing a Circuit Simulator," *IEEJ Trans. IA*, vol. 126, pp. 841-851, 2006.

[33] I. Obinelo, "Heat Sink Discussion Series, Part I: Generalized Approach for Characterization of Heat Sinks for System Level Modeling Using CFD Methods," http://www.degreec.com/obinelo_heatsink1_5_00.pdf (Milford, NH).

[34] J. Green, "Radiation Heat Transfer", Eldec Corp. Internal Presentation, Lynnwood, WA, 1994.

[35] E. Gatard, R. Sommet, R. Quere, "Nonlinear Thermal Reduced Model for Power Semiconductor Devices," *Thermal and Thermomechanical Phenomena in Electronics Systems Conf.*, 2006, pp. 638–644.

[36] L. H. Feng, E. B. Rundnyi, J. G. Korvink, C. Bohm, T. Hauck, "Compact electro-thermal

model of semiconductor device with nonlinear convection coefficent," in *Proc. Thermal, Mechanical and Multi-Physics Simulation and Experiments in Micro-Electronics and Micro-Systems*, 2005, pp. 372-375.

[37] H. C. Hottel, A. F. Saro m, Radiative transfer, New York: McGraw–Hill, 1967.

[38] R Siegel, J, R. Howell, Thermal radiation heat transfer, Washington: Hemisphere Publishing,1992.

[39] J F. Baumeister, "Thermal radiation characteristics of nonisothermal cylindrical enclosures using a numerical ray tracing technique," *ASME-HTD,* 1990, 137, pp. 73–79.

[40] M. H. N Naraghi, B. T. F. Chung, "A stochastic approach for radiative exchange in enclosures with nonparticipating medium," J *Heat Transfer,* 1984, 106, pp. 690–698.

[41] S. Mazumder, "Methods to accelerate ray tracingin the MonteCarlo method for surface-to-surface radiation transport", *Journal of Heat Transfer*, 128(9), 2006, 945–52.

[42] M. F. Modest, *Radiative Heat Transfer*, second ed., Academic Press, San Diego, CA, 2003.

[43] D. N. Trivic, B. Djordjevic, Z. Grbavcic, "Influence of particles size and concentration in particles cloud radiation," *Strojniski Vestnik/Journal of Mechanical Engineering [STROJ VEST]*, Vol. 47, no. 8, pp. 417-423. 2001.

[44] S. M. Drucker,Radiosity: An illuminating perspective. Technical report, Media Laboratory, Massachusetts Insitute of Technology, 1992.

[45] Al. Watt and M. Watt, Advanced Animation and Rendering Techniques: Theory and Practice, Addison-Wesley, 1992.

[46] A. Thompson, "The navigation system of the JPL robot," *5th Int. Joint Conf. on Arti cial Intelligence*, 1977, pp. 749-757.

[47] J. H. Jeong , E. W. Lee, H. K. Cho, "Analysis of Transient State of the Squirrel Cage Induction Motor by Using Magnetic Equivalent Circuit Method," *CEMS 2003. Sixth International Conference on Electrical Machines and Systems,* 2003, pp. 720 - 723.

[48] A. Davoudi, P. L. Chapman, "Eddy Current Modeling with Order-Reduction in Magnetic Equivalent Circuits," *IEEE Power Electronics Specialists Conf.,* 2007, pp.2069 - 2074.

[49] S. Parler, "Thermal Modeling of Aluminum Electrolytic Capacitors," *34th Annual Meeting of the IEEE IAS,* Oct. 1998, pp. 2418 – 2429.

[50] F. R. Incropera, D. P. Witt, Fundamental of Heat and Mass Transfer. New York , NY: John Wiley & Sons, 1990, Chap. 2-7.

[51] S. Haykin, Neural Networks A Comprehensive Foundation. Pearson Education, 2006.

[52] C. Leiserson, "Area-efficient graph layout (for VLSI)", *In Proceedings of the 21st Annual Symposium on the Foundations of Computer Science (1980), IEEE*, New York, 1980, pp. 270-281.

[53] J. D. Ullman, Computational Aspects of VLSI. Computer Science Press, Rockville, Md., 1984.

[54] C. Alpert and A. Kahng, “Recent Directions in Netlist Partitioning: A Survey”, *Integration—The VLSI J.*, vol. 19, nos. 1-2, pp. 1-81, 1995.

[55] F.M. Johannes, “Partitioning of VLSI Circuits and Systems," *Proc. 33rd Design Automation Conf.*, pp. 83-87, 1996.

[56] D. I. Cheng, C. Lin, M. M. Sadowska, "Circuit partitioning with logic perturbation," *IEEE/ACM International Conf. on Computer-Aided Design*, 1995, pp. 650-655.

[57] G.C. Fox, “A Review of Automatic Load Balancing an Decomposition Methods for the Hypercube,” *Numerical Algorithms for Modern Parallel Computer Architectures*, M. Schultz, ed., Springer, pp. 63-76, 1988.

[58] L. Sanchis, “Multiple-Way Network Partitioning,” *IEEE Trans. Computers,* vol. 38, pp. 62-81, 1989.

[59] H.D. Simon, “Partitioning of Unstructured Problems for Parallel Processing,” *Computing Systems in Eng.*, vol. 2, pp. 135-148, 1991.

[60] J.R. Gilbert, G.L. Miller, and S.H. Temg, “Geometric Mesh Partitioning: Implementation and Experiments,” *Proc. Ninth Int’l Parallel Processing Symp. (IPPS ’95)*, 1995.

[61] B. Hendrickson and R. Leland, “An Improved Spectral Graph Partitioning Algorithm for Mapping Parallel Computations,” *SIAM J. Scientific Computing*, vol. 16, no. 2, pp. 452-469, 1995.

[62] C. H. Q Ding, H. Xiaofeng, Z. Hongyuan, G. Ming, H. D. Simon, "A min-max cut algorithm for graph partitioning and data clustering," *IEEE Proc. Data Mining Conf.*, 2001, pp. 107-114.

[63] F. Li, Q. Zhang, W. Zhang, "Graph partitioning strategy for the topology design of industrial network," *IET Communications,* vol. 1, pp. 1104-1110, December 2007.

[64] A. Shamir, "Feature-space analysis of unstructured meshes," in *Proc.of the 14th IEEE Visualization Conference*, 2003, pp. 185-192.

[65] J. Chang; A. C.-C. Shih, H.-Y. M. Liao, W. Fang, "Principal Component Analysis-based

Mesh Decomposition," *IEEE Workshop on Multimedia Signal Processing*, 2007, pp. 292-295.

[66] K. Schloegel, G. Karypis, V. Kumar, *Graph Partitioning for High-Performance Scientific Simulations*. San Francisco, CA: Morgan Kaufmann Publishers, 2003, pp. 491-541.

[67] A. Grama, A. Gupta, G. Karypis, V. Kumar, Introduction to Parallel Computing: Design and Analysis of Algorithms. Redwood City, CA: Adison Wesley, 2003.

[68] M. R. Garey, D. S. Johnson, Computers and Instractability: A Guide to the Theory of NP-Completeness. San Francisco, CA: W. H. Freeman, 1979.

[69] S. Shlafman, A. Tal, and S. Katz, *Metamorphosis of Polyhedral surfaces using Decomposition*, The Eurographics Association and Blackwell Publishers, Computer Graphics Forum, vol. 21, 2002, pp. 219-228.

[70] S.T. Barnard and H.D. Simon, “Fast Multilevel Implementation of Recursive Spectral Bisection for Partitioning Unstructured Problems,” *Concurrency: Practice and Experience*, vol. 6, no. 2, pp. 101-117, 1994.

[71] C.J. Alpert, J.-H. Huang, and A.B. Kahng, “Multilevel Circuit Partitioning,” *Proc. 34th Design Automation Conf.*, pp. 530-533, 1997.

[72] Y.G. Saab, “A New 2-Way Multi-Level Partitioning Algorithm,” VLSI Design J., vol. 11, no. 3, pp. 301-310, 2000.

[73] Y.G. Saab, “An Effective Multilevel Algorithm for Bisecting Graphs and Hypergraphs,” *IEEE Trans. Computers*, vol. 53, no. 6, pp. 641-652, June 2004.

[74] C. Walshaw, “Multilevel Refinement for Combinatorial Optimisation Problems,” *Annals of Operations Research*, no. 131, pp. 325-372, 2004.

[75] G. Karypis, V. Kumar, "METIS A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices,
" http://glaros.dtc.umn.edu/gkhome/views/metis.

[76] M. Bikdash, P. Vega, M. El-Morsi, G. Nellis, "Reduced-order transient thermal model obtained by graph-theoretic partitioning of finite-element meshes," in *Proc. of 2005 CPES Annual Power Electronics Seminar*, 2005, pp. 49-54.

[77] L. J. Pinson, Electro-Optics, Krieger Publishing Company, October 1985.

[78] Y. A. Cengel, Heat Transfer A practical Approach. Mc Graw Hill, 2003.

[79] G. S. Sawhney, Heat and Mass Transfer, I. K. International Publishing House Pvt., , India, 2008.

[80] M. Jakob, Heat Transfer, vol. 2, John Wiley & Sons, New York, 1957:

[81] N. Greene, M. Kass, and G. Miller, "Hierarchical z-buffer visibility," In *Proc. of ACM Siggraph*, 1993, pp. 231-238.

[82] N. Carr, R. Mech and G. Miller, "Coherent Layer Peeling for Transparent High-Depth-Complexity Scenes", *Proceedings of the 23rd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics hardware*, 2008, pp. 33–40.

[83] F. Liuz, M.-C. Huangz, X.-H. Liuz and E.-H. Wuzx, "Efficient Depth Peeling via Bucket Sort", *Proceedings of the Conference on High Performance Graphics 2009,* 2009, pp. 51–57.

[84] H. Fuchs, Z. Kedem, and B. Naylor, "On visible surface generation by a priori tree structures", *Proc. of ACM Siggraph*, Vol:14(3), pp. 124–133, 1980.

[85] http://www.siggraph.org/education/materials/HyperGraph/raytrace/rtrace1.htm

[86] J. Warnock, "A hidden-surface algorithm for computer generated half-tone pictures," *Technical Report*, TR 4–15, NTIS AD-733 671, University of Utah, Computer Science Department, 1969.

[87] http://en.wikipedia.org/wiki/Painter%27s_algorithm.

[88] http://everything2.com/title/Painter%2527s+Algorithm.

[89] Visualization Concepts, http://web.iitd.ac.in/~hegde/cad/lecture/L36_visibility.pdf.

[90] Introduction to computer graphics. www.cs.virginia.edu/~gfx/Courses/2004/Intro.Spring.04/.../lecture21.ppt

[91] Max K. Agoston, Computer Graphics and Geometric Modelling: Implementation & Algorithms , 8 Feb 2005.

[92] M. D. Berg, O. Cheong, M. V. Kreveld, M. Overmarsm, Computational Geometry Algorithms and Applications. Springer-Verlag Berlin Heidelberg, 2008.

[93] E. W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik, 1 (1959), pp. 269-271.

[94] J. Lee, S. Y. Shin and K. Chwa, "Visibility-based pursuit-evasion in a polygonal room with a door", in *Proceedings of the 15th Annual ACM Symposium on Computational Geometry*, Miami Beach, FL, (1999), pp. 119-128.

[95] R. Orti, F. Durand, S. Riviere and C. Peuch, "Using the visibility complex for radiosity computation", in *1st ACM Workshop on Applied Computational Geometry,* WAGC'96,

Philadelphia, PA, Springer-Verlag Lecture Notes in Computer Science, vol. 1148, (1996), pp. 177-190.

[96] H. E. Gindy, D. Avis, "A linear algorithm for computing the visibility polygon from a point", *Journal of Algorithms*, 2, 186-197, 1981.

[97] T. Asano, "An efficient algorihm for finding the visibility polygon for polygonal region with holes", *Trans. IECE-Japan*, E-68, 557-559, 1985.

[98] T. Asamo, H. Umeo, "Systolic algorithms for computing the visibility polygon and triangulation og a polygonal region", *Parallel Computing*, 6, 209-217, 1988.

APPENDIX

COMPUTING THE VERTEX-TO-VERTEX VISIBILITY GRAPH

A $graph\ G$ is a non-empty set of objects called $vertices$ together with a set of unordered pairs of distinct vertices of G called $edges$. Each edge is incident with two vertices called the endpoints of the edge. If $u, v \in V(G)$ are endpoints of an edge e, then we write $e = uv$, and u and v are $adjacent$.

The node-to-node visibility graph, or simply the visibility graph is a fundamental geometric structure useful in many applications, including illumination and rendering, motion planning, pattern recognition and sensor networks. A common use for it has been for finding the shortest path which has been used in robot motion planning. Exploiting the fact that the shortest path consists of arcs of the visibility graph, one can find the shortest path by running Dijkstra's algorithm [93] on it. The visibility graph can also be used to solve the art gallery problem by finding the minimum dominating set of the visibility graph (NP-hard). More recently, visibility has been used in pursuer-evader problems, e.g. in [94]. Finally, the visibility complex, which contains more information than the visibility graph, has been used in illumination problems [95].

Sequential algorithms for the visibility problem have been presented both for the case of a simple polygon and of a polygon with holes. A linear algorithm was proposed based on a scan of the vertices of the polygon during which hidden regions of the polygon are identified [96]. Asano [97] presented for a polygon with h holes an $O(n\log h)$ time algorithm. A systolic algorithm was presented for the visibility within a polygon with holes [98]. The algorithm requires an $O(n)$ time bound on a linear systolic array with n processors.

Definition: A vertex v_i sees vertex v_j if and only if either $v_i = v_j$, or they lie on a line l_{ij} and the segment v_iv_j is nowhere exterior to P.

The pseudocode shown in Figure 57 describes the `VISIBLE_VERTICES` algorithm [92]. This algorithm summarizes a rotational plane sweep. The sweep is started with the half-line r pointing into the positive x-direction and proceeds in clockwise direction as shown in Figure 58. So the algorithm first sorts the vertices by the clockwise angle that the segment from p to each vertex makes with the positive x-axis. If this angle is equal for two or more vertices, the vertices are treated in order of increasing distance to p. The pseudocode for algorithm `VISIBLE` is shown in Figure 59, and it decides whether a vertex w_i is visible.

Figure 60(a)-(d) adopted from [92] shows some examples where r contains multiple vertices that can occur during VISIBLE search. In all these cases w_{i-1} is visible. In the left two cases w_i is also visible and in the right two cases w_i is not visible. $\overline{pwi}$ may or may not intersect the interior of the obstacles incident to these vertices. One can decide on the visibility of w_i as follows. If w_{i-1} is not visible then w_i is not visible either. If w_{i-1} is visible then there are two ways in which w_i can be invisible. Either the whole segment $\overline{w_{i-1} - w_i}$ lies in an obstacle of which both w_{i-1} and w_i are vertices as shown in Figure 60(d), or the segment $\overline{w_{i-1}w_i}$ is intersected by a edge in T as shown in Figure 60(b).

```
Algorithm VISIBLE_VERTICES (PolygonSet S, Point p)

1. Initialize vector T.
2. Sort the obstacle vertices in set S according to the clockwise angle that the
   half-line from p to each vertex makes with the positive x-axis. In case of ties
   vertices closer to p should come before vertices farther from p. Let
   $w_1, ..., w_n$ be the sorted list.
3. Let r be the half-line parallel to the positive x-axis starting at p. Find the
   obstacle edges that are properly intersected by r, and store them in a vector T
   in the order in which they are intersected by r.
4. Initialize W=0
5. for i =1 to n // rotate the ray to the first vertex
   a. Delete from T the obstacle edges incident to $w_i$ that lie on the
      counterclockwise side of the half-line from p to $w_i$
   b. if (VISIBLE(p, Wi, T)), Add $w_i$ to W
   c. Insert into T the obstacle edges incident to $w_i$ that lie on the
      clockwise side of the half-line from p to $w_i$
```

Figure 57. Pseudocode summarizing the VISIBLE_VERTICES algorithm

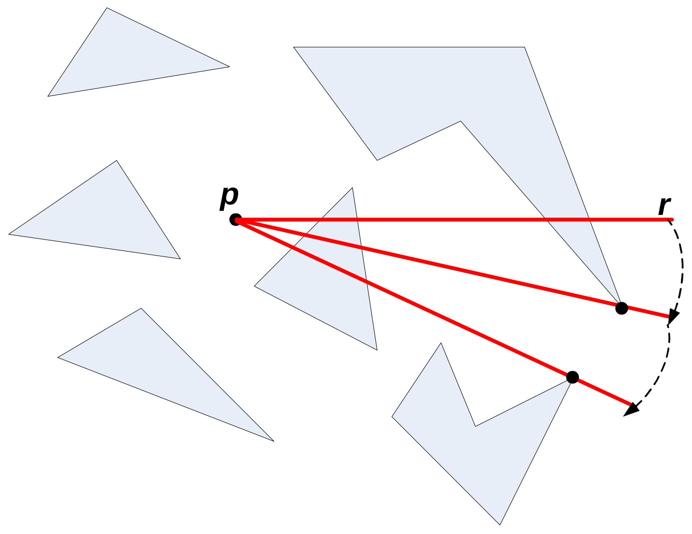


Figure 58. Rotational plane sweep

```
Algorithm VISIBLE (Point p, Point W_i, Tree T)

1. Let line l be a line from p to W_i
2. Search in T for the edge e nearest to p
3. IF e exists and e intersects l, return false
4. ELSE return true
```

Figure 59. Pseudocode summarizing the `VISIBLE` algorithm

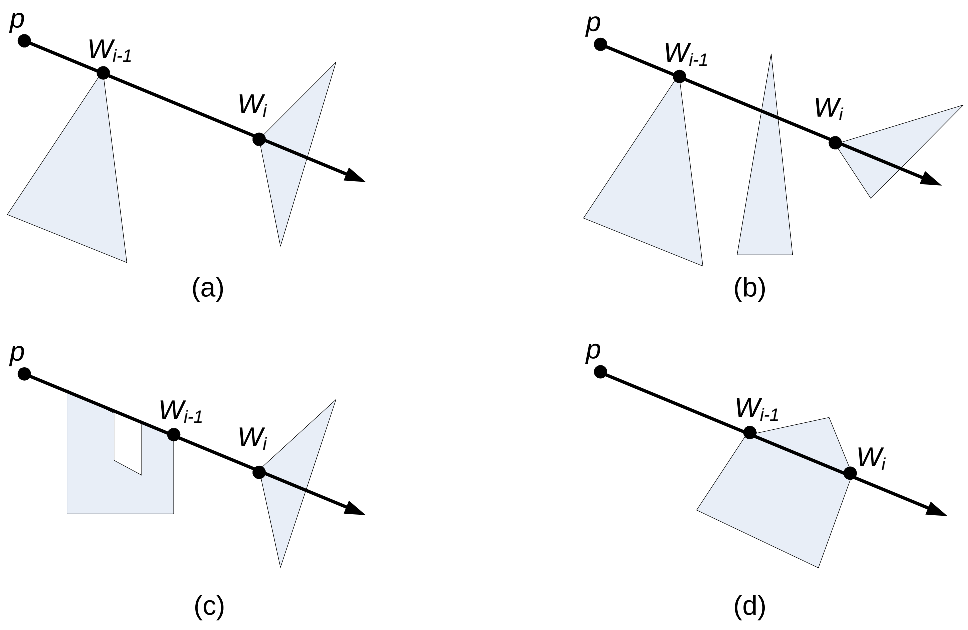


Figure 60. Some examples where r contains multiple vertices