Efficient Techniques for Handling and Sparsification of Dense Matrices in Very Large-Scale Circuit Simulation

by

Charalampos Antoniadis

Submitted to the Department of Electrical and Computer Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical and Computer Engineering

at the

UNIVERSITY OF THESSALY

September 2019

Author .....

Charalampos Antoniadis
Department of Electrical and Computer Engineering
September 25, 2019

Certified by .....

Nestor Evmorfopoulos
Assistant Professor
Thesis Supervisor

Accepted by .....

Nikolaos Bellas
Chairman, Department Committee on Graduate Theses

Αποδοτικές τεχνικές χειρισμού και αραιοποίησης πυκνών πινάκων κατά την προσομοίωση κυκλωμάτων πολύ μεγάλης κλίμακας.

Χαράλαμπος Αντωνιάδης

Περίληψη

Η τοποθέτηση περισσότερου υλικού σε ένα μοντέρνο SoC (System-on-Chip) εξαιτίας της ολοένα και μειούμενης τεχνολογίας ολοκλήρωσης έχει οδηγήσει σε πολύ μεγάλα παρασιτικά RLC δίκτυα αποτελούμενα από εκατομμύρια κόμβους, τα οποία πρέπει να προσομοιωθούν σε πολλές χρονικές στιγμές και συχνότητες έτσι ώστε να επαληθευτεί η σωστή λειτουργία του ολοκληρωμένου κυκλώματος. Επίσης, ο έλεγχος αξιοπιστίας ενός ολοκληρωμένου, εξαιτίας και πάλι της μειούμενης τεχνολογίας ολοκλήρωσης, απαιτεί να ληφθεί υπόψιν η επίδραση εκτός της παρασιτικής αυτεπαγωγής των υποσυστημάτων σε ένα SoC, που επαρκούσε σε παλαιότερες τεχνολογίες, αλλά και όλων των αμοιβαία επαγωγικών συζεύξεων μεταξύ αυτών. Ωστόσο η συμπερίληψη όλων των αμοιβαία επαγωγικών συζεύξεων καταλήγει σε ένα πλήρως πυκνό πίνακα επαγωγών που καθιστά την προσομοίωση κυκλώματος δύσκολα διαχειρίσιμη τόσο από αποθηκευτικής όσο και από υπολογιστικής πλευράς. Τεχνικές *Μείωση Τάξης Μοντέλου* (MTM) έχουν χρησιμοποιηθεί συστηματικά για να αντικαταστήσουν το πολύ μεγάλης κλίμακας παρασιτικό RLC μοντέλο με ένα πολύ μικρότερης τάξης μοντέλο με παρόμοια απόκριση στις θύρες εισόδου/εξόδου. Ωστόσο, όλες οι τεχνικές MTM καταλήγουν σε ένα μοντέλο με πυκνούς πίνακες που καθιστούν την προσομοίωση, και σε αυτήν την περίπτωση, μη πρακτική.

Έτσι, σε αυτή την διδακτορική διατριβή παρουσιάζουμε αποδοτικές τεχνικές για την επίλυση των γραμμικών συστημάτων που προκύπτουν κατά την μεταβατική ανάλυση πολύ μεγάλων αμοιβαία επαγωγικών κυκλωμάτων, όπως επίσης και μια μεθοδολογία για την αραιοποίηση των πυκνών πινάκων που προκύπτουν μετά την MTM. Οι προτεινόμενες τεχνικές για την επίλυση των γραμμικών συστημάτων στην μεταβατική ανάλυση περιλαμβάνουν την συμπίεση του πυκνού πίνακα επαγωγών προσεγγίζοντας κατάλληλα υπό-μπλοκ του αρχικού πίνακα με low-rank γινόμενα, όπως επίσης και την ανάπτυξη ενός προρυθμιστή γενικού σκοπού για την επαναληπτική επίλυση του γραμμικού συστήματος που πρέπει να λύσουμε κατά την μεταβατική ανάλυση (το οποίο συνίσταται από αραιά μπλοκ μαζί με το πυκνό μπλοκ των επαγωγών). Από την άλλη μεριά, η προτεινόμενη μεθοδολογία για την αραιοποίηση των πυκνών πινάκων μετά την MTM χρησιμοποιεί μια ακολουθία αλγορίθμων βασισμένη στον υπολογισμό του κοντινότερου διαγώνια υπερισχύων πίνακα, που έχει μια ένα-προς-ένα αντιστοιχία με γράφο, και την εν-συνεχεία αραιοποίηση αυτού του γράφου. Τα πειραματικά μας αποτελέσματα υποδεικνύουν ότι μπορεί να επιτευχθεί ένα αρκετά αραιό σύστημα ελαττωμένων πινάκων, μετά την MTM, με πολύ μικρή μείωση της ακρίβειας προσομοίωσης ενώ οι προτεινόμενες τεχνικές στην επίλυση των γραμμικών συστημάτων κατά την μεταβατική ανάλυση υποδεικνύουν ουσιώδη συμπίεση του

πίνακα επαγωγών χωρίς να θυσιάζεται η ακρίβεια προσομοίωσης, σε συνδυασμό με μια αξιοπρόσεχτη μείωση στον αριθμό των επαναλήψεων και τον συνολικό χρόνο εκτέλεσης των επαναληπτικών μεθόδων επίλυσης.

Efficient Techniques for Handling and Sparsification of Dense Matrices in Very Large-Scale Circuit Simulation

by

Charalampos Antoniadis

Submitted to the Department of Electrical and Computer Engineering
on September 25, 2019, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical and Computer Engineering

Abstract

The integration of more components into modern Systems-on-Chip (SoCs) due to the ever increasing technology scaling has led to very large parasitic networks consisting of million of nodes, which have to be simulated in many times or frequencies to verify the proper operation of chip. Moreover, because of the aggressive technology scaling, the reliability analysis of a SoC requires to take into account, except for the self-inductance which was sufficient in older technologies, but also the mutual inductance between the different subsystems in SoC. However, the inclusion of all mutual inductive couplings results in a fully dense inductance matrix that renders the circuit simulation computationally expensive. Model Order Reduction (MOR) techniques have been employed routinely to substitute the large scale parasitic model by a model of lower order with similar response at the input/output ports. However, all established MOR techniques result in dense system matrices that render their simulation impractical.

To this end, in this dissertation we present efficient techniques for the solution of the linear systems arising in transient analysis of large mutually inductive circuits, as well as a methodology for the sparsification of the dense matrices resulting from MOR. The proposed techniques for solving the linear systems in transient analysis involve the compression of the dense inductance matrix, approximating suitably sub-blocks of it by low-rank products and the development of a general purpose preconditioner for the iterative solution of the transient linear system (which comprises sparse blocks alongside the dense inductance block). On the other hand, our proposed methodology for the sparsification of the dense MOR matrices employs a sequence of algorithms based on the computation of the nearest diagonally dominant matrix, which has a direct correspondence to graph and the subsequent sparsification of that graph. Experimental results for the sparsification of the dense MOR matrices indicate that high sparsity ratio of the reduced system matrices can be achieved with very small loss of accuracy, while the proposed techniques for solving the linear systems in transient analysis of large mutually inductive circuits indicate substantial compression rates of the dense inductance matrix without compromising accuracy, along with considerable

reduction in iteration count and execution time of iterative solution methods.

Thesis Supervisor: Nestor Evmorfopoulos
Title: Assistant Professor

Acknowledgments

As my doctoral studies come to an end I would like to express my sincere gratitude to a couple of people that stood by me through this long endeavor. It is certain that without their help, I would have not been able to complete my dissertation.

First of all, I would like to thank my Ph.D. advisors, Professor Nestor Evmorfopoulos, Professor George Stamoulis and Professor Thanasis Loukopoulos for their trust in me, the exemplary collaboration that we had and their fruitful advices they gave lavishly to me that make possible the seeds of research ideas on the topics the dissertation is all about, to flourish.

I would also like to thank my thesis committee members, Professor Spyridon Nikolaidis from Aristotle University of Thessaloniki, Professor Panagiotis Bozanis from International Hellenic University, as well as, Professor Christos Sotiriou and Professor Georgios Dimitriou both from University of Thessaly, for providing important feedback for my dissertation.

My sincere thanks to all my friends for their encouragement all these years. Especially, I would like to thank my friends from the Electronics Lab for their patience to tolerate any of my whimsies, sorry guys. Without their help, I would not have led such a fruitful and joyful life in the Department of Electrical and Computer Engineering. Particularly, I would like to thank all my close friends, they know who they are, for their constant support, help and valuable discussions. Also, many thanks to Mary, Lena, Maria, and Despoina for their continuous secretarial support.

Finally, I appreciate so much the unconditional love, help, trust, and support from my family. In particular my parents, Giorgos Antoniadis and Efi Fostiropoulou partially funded my doctoral studies and undoubtedly without their financial support maybe the completion of my doctoral studies in the University of Thessaly would be a figment of my imagination. There are no words that can express my gratitude and appreciation for all they have done for me. Without their patience and support, I would not have been able to finish my dissertation. The least I can do in recognition is to dedicate this dissertation to them.

This doctoral thesis has been examined by a Committee of the Department of Electrical and Computer Engineering and the Department of Computer Science and Biomedical Informatics both from the University of Thessaly, the Aristotle University of Thessaloniki and the International Hellenic University as follows:

Professor Nestor Evmorfopoulos .....
Chairman, Thesis Supervisor
Assistant Professor of Electrical and Computer Engineering
University of Thessaly

Professor Georgios Stamoulis .....
Member, Thesis Committee
Professor of Electrical and Computer Engineering
University of Thessaly

Professor Thanasis Loukopoulos .....
Member, Thesis Committee
Assistant Professor of Computer Science and Biomedical Informatics
University of Thessaly

Professor Spyridon Nikolaidis .....
Member, Examination Committee
Professor of Physics
Aristotle University of Thessaloniki

Professor Panayiotis Bozanis .....
Member, Examination Committee
Professor of Science and Technology
International Hellenic University

Professor Christos Sotiriou .....
Member, Examination Committee
Associate Professor of Electrical and Computer Engineering
University of Thessaly

Professor Georgios Dimitriou .....
Member, Examination Committee
Assistant Professor of Computer Science and Telecommunications
University of Thessaly

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Related Work

The motivation behind our research in this dissertation is to provide sparsification solutions and methods for handling dense matrices that arise in circuit simulation. In particular, we deal with the dense inductance matrix, when all mutual inductances are taken into consideration, as well as the dense matrices resulting from the application of MOR techniques.

1.1.1 Simulation of large dense mutually inductive circuits

In the past, it has been sufficient to include only self inductance in the RLC simulation, resulting in linear systems with sparse coefficient matrices. For such systems, iterative solvers are the methods of choice as they offer small memory requirements and excellent scaling against problem size. However, in modern high-frequency SoCs there is an increasing demand for modeling all mutual inductive couplings between the different blocks of the chip [1], leading to a fully dense inductance matrix that offsets the scaling properties and storage requirements of iterative solvers.

As direct sparsification (by truncation) of the dense inductance matrix is well-known to lead to loss of circuit passivity and simulation stability [2], most previous attempts in the literature have focused on sparsifying the inverse matrix (called the

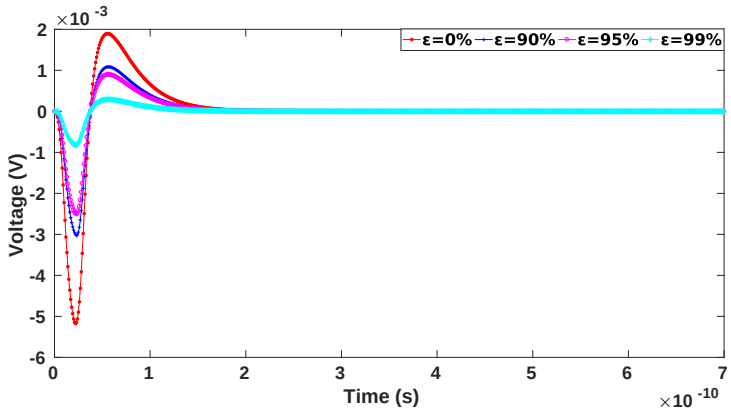


Figure 1-1: Voltage response at the far end of the second wire (by applying a ramp voltage source at the first wire) in a bus of two-parallel wires, for different sparsity ratios of the sparse approximation of the reluctance matrix by direct truncation.

reluctance matrix), which is especially amenable to sparsification due to its diagonal dominance [3][4]. Taking advantage of the diagonal dominance of the reluctance matrix, one can obtain a naive sparse approximation of it by directly truncating all its off-diagonal entries that are smaller than a predefined threshold. By doing so, the diagonal dominance property and consequently the positive definiteness of the reluctance matrix is not violated, making the sparse approximation of the reluctance matrix suitable to be integrated into a circuit simulation framework. However, the direct truncation approach introduces unacceptable error in transient analysis for the high sparsity ratios that we are typically interested in. We demonstrate that observation in Fig. 1-1.

Other approaches proposed in the literature for the sparsification of the reluctance matrix focus on the avoidance of the expensive (both from a computational as well as a memory storage perspective) inversion of the inductance matrix. In [5] authors predict initially the locations of the non-zero entries of matrix the reluctance matrix, exploiting a graph colouring technique, and then they compute only those entries for a prescribed sparsity ratio. Window-based approaches which work either directly on the matrix [6] or the physical geometry [7][8] result in local inversion of smaller matrices, where each one corresponds to a window around a conductor segment. However, all these approaches only aim at accelerating the sparsification procedure, and do not

improve on the accuracy obtained by direct truncation. In [9] a more sophisticated sparsification, which is based on the equivalence of the reluctance matrix with a graph, leads to better results in simulation accuracy compared to direct truncation but requires like direct truncation the prior inversion of the large dense matrix which is a very expensive computational procedure, making that kind of approaches feasible only for problems of moderate size.

1.1.2 Dense MOR matrices

Sign-off analysis of a SoC entails the simulation of RLC parasitics extracted from various components in an integrated circuit, in order to verify the proper operation of the circuit. However, due to the ever increasing technology scaling and the subsequent integration of even more components into a chip the parasitic network can be gigantic. The simulation of such parasitic networks can be extremely time- consuming or, in some cases, even infeasible due to their sheer size.

Model Order Reduction (MOR) techniques are typically employed to substitute the large scale models by lower dimensional ones with similar response at the input/output ports. MOR techniques are generally classified into two classes, namely moment-matching or Krylov-subspace techniques [10][11][12][13][14] and balancing type or Gramian-based techniques [15][16]. The application of both classes of MOR techniques leads to reduced order models with dense matrices, whose cost of employing in simulation can easily overshadow the benefits obtained from dimension reduction.

Specifically, [17] firstly divides the nodes into a group of nodes corresponding to ports that have to be preserved and a group of internal nodes that can be eliminated, and then finds the Schur complement of the system, and performs sparse matrix manipulations on top of it. The approach in [18] partitions the circuit into subcircuits and then substitutes each subcircuit with suitable RLC macromodels that catch the low-order moments of the admittance matrix in each partition. Furthermore, [19] after deriving a description of the circuit with respect to node voltages and magnetic flux, reorders system matrices to Boarder Block Diagonal (BBD) form and finally applies

MOR. The problem with all the above methods is that they either do not produce very sparse and accurate models, or they rely on heuristics and circuit specific criteria. None of them provides a rigorous mathematical framework to address sparsification problem of the dense matrices resulting from the application of MOR procedures.

1.2 Contribution

In order to address the problems arising in circuit simulation after the modeling of all mutual inductive couplings between the different components in an IC, as well as the dense matrices resulting after the application of MOR, in this dissertation we present two approaches that provide a solution to each problem separately, as described below:

- We propose an approach for the sparsification of dense MOR circuit matrices, which employs a sequence of algorithms based on the computation of the nearest diagonally dominant matrix and the subsequent sparsification of the corresponding graph. Our main contribution is that we transform the problem of the sparsification of the reduced model matrices to the sparsification of the nearest matrices corresponding to a graph, in order to exploit efficient graph sparsification techniques. In addition, since the sparsified matrices are of the Laplacian kind with direct correspondence to weighted graphs, the sparsified reduced order model has a straightforward realization to an equivalent RC circuit with positive elements.

- We propose the compression (instead of sparsification) of the actual inductance matrix via the approximation of large off-diagonal blocks by low-rank products, in a scheme known as hierarchical matrix (or $\mathcal{H}$-matrix) format. This format conserves memory by storing only the low-rank factors of the blocks, while enabling the execution of basic matrix-vector operations of iterative solvers in near optimal complexity. Moreover, we propose a block preconditioner based on an efficient approximation of the Schur complement, which improves considerably

the convergence rate of iterative methods and has inexpensive application inside every iteration.

1.3 Outline

The rest of the PhD dissertation is organized as follows. Firstly, Chapter 2 presents a rigorous methodology for the sparsification of dense matrices resulting from MOR. Chapter 3 explains the compression of the dense inductance matrix by low-rank rank products in $\mathcal{H}$-matrix format and presents a genreal purpose preconditioner for the iterative solution of the transient linear system (which comprises sparse blocks alongside the dense inductance block). Finally, Chapter 4 concludes the dissertation and proposes ideas for future work.

Chapter 2

Efficient Sparsification of Dense Circuit Matrices in Model Order Reduction

2.1 Background

Consider an RLC circuit with n nodes (excluding ground), ℓ inductive branches with mutual coupling between them, g conductive branches, c capacitive branches, p inputs and q outputs which is described according to the Modified Nodal Analysis (MNA) formulation [20] in the time domain as follows:

$$\begin{bmatrix} \mathbf{A}_{cr}\mathbf{C}_d\mathbf{A}_{cr}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{L} \end{bmatrix} \begin{bmatrix} \mathbf{v}'(t) \\ \mathbf{i}'(t) \end{bmatrix} + \begin{bmatrix} \mathbf{A}_{gr}\mathbf{G}_d\mathbf{A}_{gr}^T & \mathbf{A}_L \\ -\mathbf{A}_L^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{v}(t) \\ \mathbf{i}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{B}_u \\ \mathbf{0}^{\ell \times p} \end{bmatrix} \mathbf{u}(t)$$
$$\mathbf{y}(t) = \begin{bmatrix} \mathbf{E}_u & \mathbf{0}^{q \times \ell} \end{bmatrix} \begin{bmatrix} \mathbf{v}(t) \\ \mathbf{i}(t) \end{bmatrix} \tag{2.1}$$

where $\mathbf{v} \in \Re^n$, $\mathbf{i} \in \Re^\ell$, $\mathbf{u} \in \Re^p$, $\mathbf{y} \in \Re^q$ are the vectors of node voltages, branch currents, input excitations from independent sources at the nodes (with voltage sources being transformed into Norton-equivalent current sources) and output measurements

respectively, $\mathbf{B}_u \in \Re^{n\times p}$ and $\mathbf{E}_u \in \Re^{q\times n}$ are the input-to-node and node-to-output connectivity matrices, $\mathbf{A}_{gr} \in \Re^{n\times g}$ and $\mathbf{A}_{cr} \in \Re^{n\times c}$ are the node-to-branch *reduced* incidence matrices (with respect to ground), $\mathbf{G}_d \in \Re^{g\times g}$, $\mathbf{C}_d \in \Re^{c\times c}$ are positive diagonal matrices containing the branch conductances and branch capacitances respectively, $\mathbf{L} \in \Re^{\ell\times\ell}$ is dense inductance matrix (with self-inductances as diagonal entries and mutual inductances as off-diagonal entries), and $\mathbf{A}_L \in \Re^{n\times\ell}$ is the corresponding node-to-inductive branch incidence matrix. It is well-known [21] that the products

$$\mathbf{G}_n \equiv \mathbf{A}_{gr}\mathbf{G}_d\mathbf{A}_{gr}^T, \quad \mathbf{C}_n \equiv \mathbf{A}_{cr}\mathbf{C}_d\mathbf{A}_{cr}^T \tag{2.2}$$

are symmetric and diagonally dominant (SDD) matrices with positive diagonal elements and non-positive off-diagonal elements, i.e. $g_{ii} > 0$, $c_{ii} > 0$ (sum of conductances or capacitances connected to node i), $g_{ij} = g_{ji} \leqslant 0$, $c_{ij} = c_{ji} \leqslant 0$ (opposite of conductance or capacitance between nodes i and j), and $g_{ii} \geq -\sum_{\substack{j=1\\ j\neq i}}^{n} g_{ij}$, $c_{ii} \geq -\sum_{\substack{j=1\\ j\neq i}}^{n} c_{ij}$, $\forall i = 1, \ldots n$. These types of matrices are subsets of the broader class of *M-matrices* [23], and we are going to refer to them as "circuit-type" M-matrices. However, the inductance matrix $\mathbf{L}$ is not diagonally dominant but its inverse $\mathbf{L}^{-1}$ (called the *reluctance* matrix) is known to be SDD with positive diagonal elements and non-positive off-diagonal elements [3], i.e a circuit-type M-matrix.

The fact that $\mathbf{A}_{gr}$ and $\mathbf{A}_{cr}$ are reduced incidence matrices (not containing the row corresponding to the ground node) means that both $\mathbf{G}_n = \mathbf{A}_{gr}\mathbf{G}_d\mathbf{A}_{gr}^T$ and $\mathbf{C}_n = \mathbf{A}_{cr}\mathbf{C}_d\mathbf{A}_{cr}^T$ will have at least one row sum greater than zero, i.e. $g_{ii} > -\sum_{j=1,j\neq i}^{n} g_{ij}$ and $c_{ii} > -\sum_{j=1,j\neq i}^{n} c_{ij}$ for the nodes i where a conductance or capacitance is connected to ground. These can be written as $\mathbf{G}_n = \mathbf{D}_G + \mathbf{A}_g\mathbf{G}\mathbf{A}_g^T$ and $\mathbf{C}_n = \mathbf{D}_C + \mathbf{A}_c\mathbf{C}\mathbf{A}_c^T$, where $\mathbf{D}_G$ and $\mathbf{D}_C$ are diagonal matrices with the conductances and capacitances to ground, $\mathbf{A}_g$ and $\mathbf{A}_c$ are normal (not reduced) incidence matrices for the branches not being connected to ground, and $\mathbf{G}$, $\mathbf{C}$ are diagonal matrices with the conductances and capacitances of these branches. The products:

$$\mathbf{L}_G \equiv \mathbf{A}_g \mathbf{G} \mathbf{A}_g^T, \quad \mathbf{L}_C \equiv \mathbf{A}_c \mathbf{C} \mathbf{A}_c^T \tag{2.3}$$

are known as the *Laplacian* matrices [24] of the weighted graphs consisting of the conductive branches and the capacitive branches not connected to ground, and having as weights the conductances and capacitances of these branches. Thus, any circuit-type M-matrix can be written as the sum of a diagonal matrix and the Laplacian matrix of a weighted graph.

Formally, given a weighted graph $G = (V, E, w)$ with set of vertices (nodes) $V = \{1, 2, \ldots, n\}$, set of edges (branches) $E = \{(i, j) |\ i, j \in V\}$, and weight function $w(i, j)$, the Laplacian matrix of G is defined as follows:

$$\mathbf{L}_G = \mathbf{A}_G \mathbf{W} \mathbf{A}_G^T = \sum_{(i,j) \in E} w(i, j)(\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^T \tag{2.4}$$

where $\mathbf{W}$ is a diagonal matrix with the edge weights and $\mathbf{A}_G$ is the vertex-to-edge incidence matrix with columns for every edge (i, j) that equal $\mathbf{e}_i - \mathbf{e}_j$, where $\mathbf{e}_i$ is the elementary unit vector with 1 at the position i and 0's everywhere else. Because each row and column sum of $\mathbf{L}_G$ equals zero, a Laplacian matrix is always rank-deficient and there does not exist a regular inverse of $\mathbf{L}_G$. However, the algorithms presented in this paper employ $\mathbf{L}_G$ only on vectors existing in its column space $\mathcal{R}(\mathbf{L}_G)$ (i.e. not in its nullspace $\mathcal{N}(\mathbf{L}_G) = \mathcal{R}^{\perp}(\mathbf{L}_G)$), where the *Moore-Penrose pseudoinverse* $\mathbf{L}_G^+$ acts as a normal inverse, i.e. for each $\mathbf{y} \in \mathcal{R}(\mathbf{L}_G)$, $\mathbf{L}_G^+ \mathbf{y}$ is the unique $\mathbf{x} \in \mathcal{R}(\mathbf{L}_G)$ such that $\mathbf{L}_G \mathbf{x} = \mathbf{y}$.

Model Order Reduction (MOR) techniques aim at approximating the model of (2.1) by another model of reduced order $r < (n + \ell)$, through a process of projecting the model matrices onto lower-dimensional subspaces of dimension r. Instead of projecting the original block model matrices $\widetilde{\mathbf{G}}$, and $\widetilde{\mathbf{C}}$, structure-preserving MOR techniques like [25] and [19] work on the individual blocks $\mathbf{G}_n$, $\mathbf{C}_n$, $\mathbf{L}$ and $\mathbf{A}_L$. The difference between them is that [25] involves the inductance matrix $\mathbf{L}$ while [19] involves the reluctance matrix $\mathbf{L}^{-1}$. However, they both have to work with the incidence block $\mathbf{A}_L$ which is not SDD.

A different approach is to solve for the unknown currents from the lower ℓ equations in (2.1) and substitute to the upper n equations, in order to obtain the *second-order* expression of (2.1):

$$\begin{aligned} \mathbf{C}_n\mathbf{v}'(t) + \mathbf{G}_n\mathbf{v}(t) + \mathbf{\Gamma}\int_0^t \mathbf{v}(t) &= \mathbf{B}_u\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{E_u}\mathbf{v}(t) \end{aligned} \tag{2.5}$$

where $\mathbf{\Gamma} = \mathbf{A}_L\mathbf{L}^{-1}\mathbf{A}_L^T$. MOR techniques that work on the above second-order formulation, like [26][27][28][29], obtain a reduced-order model through the following matrix transformations:

$$\begin{gathered} \widehat{\mathbf{G}}_n = \mathbf{U}^T\mathbf{G}_n\mathbf{V}, \quad \widehat{\mathbf{C}}_n = \mathbf{U}^T\mathbf{C}_n\mathbf{V}, \quad \widehat{\mathbf{\Gamma}} = \mathbf{U}^T\mathbf{A}_L\mathbf{L}^{-1}\mathbf{A}_L^T\mathbf{V} \\ \widehat{\mathbf{B}}_u = \mathbf{U}^T\mathbf{B}_u, \quad \widehat{\mathbf{E}}_u = \mathbf{E}_u\mathbf{V} \end{gathered} \tag{2.6}$$

where $\mathbf{U}$, $\mathbf{V}$ are projection matrices onto a lower dimensional subspace.

The above projections (and, generally, the projections of any MOR technique) do not generally preserve the properties of circuit-type M-matrices, i.e. diagonal dominance and non-positive off-diagonal elements, which allow them to have a direct correspondence to circuits. The biggest problem however, from the projections inherent in MOR is that sparsity is lost, which can render impractical any time or frequency domain simulation involving the solution of linear systems in the model matrices, and offsets the benefits from the reduction of order. We will address these problems in the remainder of the chapter.

2.2 Approximation of Projected MOR Matrices by Circuit-type M-matrices

2.2.1 Back projection to the nearest SDD matrix

The matrices $\widehat{\mathbf{G}}_n$, $\widehat{\mathbf{C}}_n$ and $\widehat{\mathbf{\Gamma}}$ resulting from the MOR projections (2.6) are not diagonally dominant, but they are expected to be close to a DD matrix (in a suitable matrix norm) since they are obtained through the projections $\mathbf{U}^T\mathbf{G}_n\mathbf{V}$, $\mathbf{U}^T\mathbf{C}_n\mathbf{V}$, and $(\mathbf{A}_L^T\mathbf{U})^T\mathbf{L}^{-1}(\mathbf{A}_L^T\mathbf{V})$ which, being good approximations, preserve the dominant eigenvalues (modes) of the initial DD matrices $\mathbf{G}_n$, $\mathbf{C}_n$, and $\mathbf{L}^{-1}$ (we present experimental evidence of the proximity of $\widehat{\mathbf{G}}_n$, $\widehat{\mathbf{C}}_n$ and $\widehat{\mathbf{\Gamma}}$ to DD matrices in section 2.6) and therefore, in this section we consider the problem of finding the nearest SDD matrix to a given $n \times n$ matrix $\mathbf{A}$ with positive diagonal elements, in some suitable matrix norm which is chosen to be the Frobenius norm, i.e. the following least-squares problem:

$$\begin{aligned} \min_{\mathbf{X}} \quad & \|\mathbf{A}-\mathbf{X}\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}(a_{ij}-x_{ij})^2} \\ \text{s.t.} \quad & \mathbf{X} \in \text{SDD} \end{aligned} \tag{2.7}$$

This constrained optimization problem can be solved by Algorithm [30] which (after initializing $\mathbf{X} = \mathbf{A}$) alternatingly projects $\mathbf{X}$ onto the set of symmetric matrices by $\frac{\mathbf{X}+\mathbf{X}^T}{2}$, and the set of diagonally dominant matrices by Algorithm 2 (whose non-empty intersection determines the feasible set of solutions), until the distance between two consecutive projections reaches a pre-specified tolerance (see Fig. 2-1). The DD-projection Algorithm 2 computes for each non-DD row an "average" of the off-diagonal elements which then adds to the diagonal element while subtracting it from the rest of the elements for this particular row.

The Algorithm 1 can be proven to converge to the nearest SDD matrix [31] while its rate of convergence depends upon the angle between the active faces at the solution. Criteria for the selection of the supporting hyperplanes of the active faces have been proposed in [32] to improve the convergence rate. As for the computational complexity

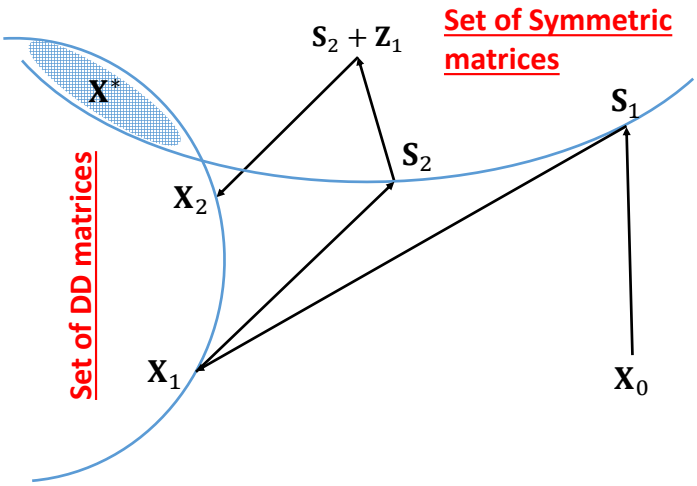


Figure 2-1: The sequence of projections between the set of symmetric and the set of diagonally dominant matrices in Algorithm 1. $\mathbf{X}_0$ is equal to the matrix we are interested in projecting to the set of symmetric and diagonally dominant matrices. The solution $\mathbf{X}^*$ lies in the intersection of the set of DD matrices and the set of Symmetric matrices, while its Frobenius distance from $\mathbf{X}_0$ is the smallest one (there may be more than one $\mathbf{X}_i$ in the intersection of the two sets).

of every iteration itself, both the symmetric projection $\frac{\mathbf{X}+\mathbf{X}^T}{2}$ and the DD projection entail $O(n^2)$ flops (the "while" loop in Algorithm 3 only takes a few iterations in practice [31]), and also exhibit significant degree of parallelism (since each row is treated independently of the others in Algorithm 2).

Algorithm 1 Find the nearest SDD matrix to an $n \times n$ matrix $\mathbf{A}$ under the Frobenius norm

function $\mathbf{X} =$ PRJSDD($\mathbf{A}$, tol)
  $\mathbf{Z} = \mathbf{0}$; $\mathbf{X} = \mathbf{A}$
  repeat
    $\mathbf{S} = \frac{\mathbf{X}+\mathbf{X}^T}{2}$
    $\mathbf{X}_{prv} = \mathbf{X}$
    $\mathbf{X} = prjDD(\mathbf{S} - \mathbf{Z})$
    $\mathbf{Z} = \mathbf{X} - \mathbf{S} + \mathbf{Z}$
  until $\|\mathbf{X} - \mathbf{X}_{prv}\|_F \leq tol$
end function

Algorithm 2 Project an $n \times n$ matrix $\mathbf{A}$ with positive diagonal elements onto the set of DD matrices

```
1: function X = PRJDD(A)
2:     for i =1 to n do
3:         a = A(i,:)
4:         s = sum_{j=1,j≠i}^{n} |a(j)|
5:         if a(i) ≥ s then
6:             x = a
7:         end if
8:         if a(i) ≥ 0 && a(i) < s then
9:             x = prjDDrow(a, i)
10:        end if
11:        X(i,:) = x
12:    end for
13: end function
```

2.2.2 Improvement in case of RC circuits

In the case of RC circuit we can apply appropriate congruence transformations like those first appeared in the PACT method [33], so that the $\widehat{\mathbf{G}}_n$ needs not be approximated by its nearest diagonally dominant.

Before applying the congruence transformations, we have to rearrange the equations of (2.5) (of course without the term $\mathbf{\Gamma} \int_0^t \mathbf{v}(t)$ because we consider RC circuits) so that the first $o = p + q$ equations correspond to port nodes, while the rest of them ($i = n - o$) correspond to internal nodes. Thus, (2.5) can be re-written as:

$$\begin{bmatrix} \mathbf{G}_o & \mathbf{G}_c^T \\ \mathbf{G}_c & \mathbf{G}_i \end{bmatrix} \begin{bmatrix} \mathbf{x}_o(t) \\ \mathbf{x}_i(t) \end{bmatrix} + \begin{bmatrix} \mathbf{C}_o & \mathbf{C}_c^T \\ \mathbf{C}_c & \mathbf{C}_i \end{bmatrix} \begin{bmatrix} \frac{d\mathbf{x}_o(t)}{dt} \\ \frac{d\mathbf{x}_i(t)}{dt} \end{bmatrix} = \begin{bmatrix} \mathbf{B}_o \\ \mathbf{0}^{i \times p} \end{bmatrix} \mathbf{u}(t)$$
$$\mathbf{y}(t) = \begin{bmatrix} \mathbf{E}_o & \mathbf{0}^{q \times i} \end{bmatrix} \begin{bmatrix} \mathbf{x}_o(t) \\ \mathbf{x}_i(t) \end{bmatrix} \tag{2.8}$$

where $\mathbf{x}_o$ and $\mathbf{x}_i$ represent the o port and the i internal unknown node voltages respectively, $\mathbf{G}_o$ and $\mathbf{C}_o \in \Re^{o \times o}$ represent the conductive and the capacitive interconnections among the port nodes, $\mathbf{G}_i$ and $\mathbf{C}_i \in \Re^{i \times o}$ describe the conductive and the capacitive interconnections among the internal nodes, $\mathbf{G}_c$ and $\mathbf{C}_c \in \Re^{o \times i}$ represent the con-

Algorithm 3 Compute the row of the projection of an $n \times n$ matrix $\mathbf{A}$ onto the set of DD matrices

function $\mathbf{x}$ = PRJDDROW($\mathbf{a}$, i)
  $d = \sum_{j=1, j\neq i}^{n} |\mathbf{a}(j)|$
  $d = d - \mathbf{a}(i)$
  $\mathbf{a}_{off} = [\mathbf{a}(1), \ldots, \mathbf{a}(i), \mathbf{a}(i+1), \ldots, \mathbf{a}(n)]$
  $c = \#\text{nonzeros}(\mathbf{a}_{off}) + 1$
  $b = d/c$
  $s = 1$
  while $s == 1$ **do**
    $s = 0$
    for j =1 `to` n **do**
      if $\mathbf{a}(j) \neq 0$ && $j \neq i$ **then**
        $aux = \mathbf{a}(j) - \text{sign}(\mathbf{a}(j)) \cdot b$
        if $\text{sign}(aux) \cdot \text{sign}(\mathbf{a}(j)) < 0$ **then**
          $d = d - |\mathbf{a}(j)|$
          $c = c - 1$
          $\mathbf{a}(j) = 0$
          $s = 1$
        end if
      end if
    end for
    $b = d/c$
  end while
  for j =1 `to` n **do**
    if $j \neq i$ **then**
      if $\mathbf{a}(j) == 0$ **then**
        $\mathbf{x}(j) = 0$
      else if $\mathbf{a}(j) > 0$ **then**
        $\mathbf{x}(j) = \mathbf{a}(j) - b$
      else
        $\mathbf{x}(j) = \mathbf{a}(j) + b$
      end if
    else
      $\mathbf{x}(i) = \mathbf{a}(i) + b$
    end if
  end for
end function

ductive and the capacitive interconnections between the port nodes and the internal nodes, while $\mathbf{B}_o \in \Re^{o \times p}$ and $\mathbf{E}_o \in \Re^{q \times o}$ are the upper $o \times p$ and $q \times o$ submatrices of the input-to-node connectivity matrix $\mathbf{B}_u$ and node-to-output connectivity matrix $\mathbf{E}_u$ respectively after the rearrangement of equations. Then, we transform $\mathbf{G}_n$, $\mathbf{C}_n$ in (2.8) as follows:

$$\mathbf{G}_n^{'} = \mathbf{X}^T \mathbf{G}_n \mathbf{X} = \begin{bmatrix} \mathbf{G}_o - \mathbf{G}_c^T \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_i \end{bmatrix}$$

$$\mathbf{C}_n^{'} = \mathbf{X}^T \mathbf{C}_n \mathbf{X} = \begin{bmatrix} \mathbf{C}_o - \mathbf{B}^T \mathbf{A} - \mathbf{A}^T \mathbf{C}_c & \mathbf{B}^T \mathbf{L}^{-T} \mathbf{U} \\ \mathbf{U}^T \mathbf{L}^{-1} \mathbf{B} & \mathbf{C}_i^{'} \end{bmatrix} \tag{2.9}$$

with

$$\mathbf{X} = \begin{bmatrix} \mathbf{I}_o & \mathbf{0} \\ -\mathbf{A} & \mathbf{L}^T \end{bmatrix}$$

where $\mathbf{A} = \mathbf{G}^{-1}\mathbf{G}_c$, $\mathbf{B} = \mathbf{C}_c - \mathbf{C}_i\mathbf{A}$, $\mathbf{L}$ is the lower triangular matrix from the Cholesky factorization of $\mathbf{G}_i$ ($\mathbf{G}_i = \mathbf{L}\mathbf{L}^T$). The above transformations do not alter the transfer function of the system (2.8) and MOR techiques, like PACT and TurboMOR [34] focus on reducing the $(n-o) \times (n-o)$ submatrix $\mathbf{C}_i^{'}$ (since the $o \times o$ submatrix $\mathbf{G}_o$ is small if the ports are few).

Looking at the model in (2.9), it can be proven that matrix $\mathbf{G}_n^{'}$ is circuit-type M-matrix because it consists of the Schur complement of matrix $\mathbf{G}_n$, which is circuit-type M matrix itself, and the identity matrix. Therefore, it is not required to approximate $\mathbf{G}_n^{'}$ with its nearest circuit-type M matrix and as result the error induced due to this approximation can be avoided.

2.2.3 Conversion of a general SDD matrix to a circuit-type M-matrix

The back projection procedure to the nearest SDD matrix leaves the sign of the off-diagonal elements unchanged, and thus does not generally lead to a circuit type M-matrix where the off-diagonal elements are all non-positive. However, if we decompose a matrix $\mathbf{A}$ into $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2$ then $\mathbf{Ax} = \mathbf{b}$ is equal to $\mathbf{A}_1\mathbf{x} + \mathbf{A}_2\mathbf{x} = \mathbf{b}$ (or $\mathbf{A}_1\mathbf{x} - \mathbf{A}_2(-\mathbf{x}) = \mathbf{b}$) and also $-\mathbf{A}_1\mathbf{x} - \mathbf{A}_2\mathbf{x} = -\mathbf{b}$ (or $-\mathbf{A}_2\mathbf{x} + \mathbf{A}_1(-\mathbf{x}) = -\mathbf{b}$). Now, if $\mathbf{A}$ is SDD and $\mathbf{A}_1 = \mathbf{A}_{dn}$, $\mathbf{A}_2 = \mathbf{A}_p$, where $\mathbf{A}_{dn}$ contains the diagonal and negative off-diagonal elements and $\mathbf{A}_p$ contains the positive off-diagonal elements of $\mathbf{A}$ (and zeros everywhere else) then we can write the previous systems as $\mathbf{A}_{dn}\mathbf{x} - \mathbf{A}_p(-\mathbf{x}) = \mathbf{b}$ and $-\mathbf{A}_p\mathbf{x} + \mathbf{A}_{dn}(-\mathbf{x}) = -\mathbf{b}$. This, can be put in the form of the augmented sytem:

$$\begin{bmatrix} \mathbf{A}_{dn} & -\mathbf{A}_p \\ -\mathbf{A}_p & \mathbf{A}_{dn} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ -\mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ -\mathbf{b} \end{bmatrix} \tag{2.10}$$

which will be SDD with non-positive off-diagonal elements. Thus, for all simulation purposes, an SDD matrix can be substituted by a circuit-type M-matrix with equivalent solution and twice the size (which is hardly an issue for reduced order models whose dimension is at most a few thousand and where matrix density is the major problem).

2.3 Sparsification of Circuit-type M-matrices

The matrices resulting from the nearest projection and conversion procedures of the previous section are dense circuit-type M-matrices, and thus they can be decomposed into the sum of a diagonal matrix and a dense Laplacian matrix. In this section we focus on sparsifying those dense Laplacian matrices. Because a Laplacian matrix $\mathbf{L}_G$ has a direct correspondence to a weighed graph, by considering every non-zero off-diagonal element l_{ij} $(i \neq j)$ as an edge (i, j) between vertices i and j with weight $w(i, j) = -l_{ij}$, we can cast the problem of sparsifying $\mathbf{L}_G$ to sparsifying its corre-

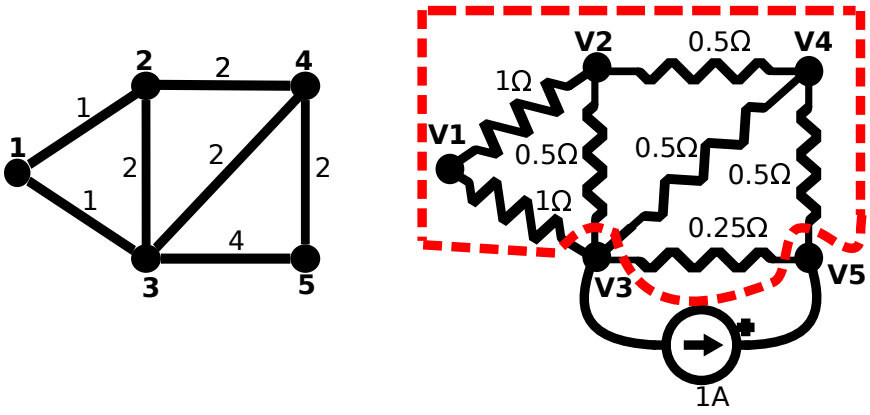


Figure 2-2: Electrical circuit analogy of graph, where each edge has resistance the inverse of its weight. The "effective resistance" of edge $(3, 5)$ is the voltage drop across the vertices 3 and 5 when we apply a current source of 1A between them.

sponding graph. An efficient graph sparsification method should not naively truncate edges with weight below a certain threshold, but should instead aim at preserving the graph "energy" or equivalently the eigenvalues of its Laplacian matrix. Such a method that for a given dense graph $G = (V, E, w)$ with $O(n^2)$ edges (where n is the number of vertices) constructs a sparse subgraph $H = (V, \tilde{E}, \tilde{w})$ of G with $O(nlogn)$ edges, in a way that the energy of G is preserved in H to the largest extent possible, is given in [35]. The proposed algorithm is based on the concept of "effective resistance" of an edge (i, j) which is defined as follows:

$$R^{eff}_{(i,j)} = (\mathbf{e}_i - \mathbf{e}_j)^T \mathbf{L}^+_G (\mathbf{e}_i - \mathbf{e}_j) \tag{2.11}$$

where $\mathbf{L}^+_G$ is the pseudoinverse of $\mathbf{L}_G$ and $\mathbf{e}_i$, $\mathbf{e}_j$ are elementary unit vectors with 1 at positions i and j respectively, and zeros everywhere else. In electrical network analogy the effective resistance of edge (i, j) represents the voltage drop across (i, j) when we apply a unit current source between its endpoint vertices i and j (see Fig. 2-2).

It is proven in [35], for a given $\epsilon \in [\frac{1}{\sqrt{n}}, 1]$ representing a tradeoff between sparsity and accuracy, a random sampling (with replacement) of $\frac{0.09nlogn}{\epsilon^2}$ edges of G with probability mass function (PMF) $p(i, j) = \frac{w(i,j)R^{eff}_{(i,j)}}{n-1}$ to include in H, will preserve the

eigenvalues of Laplacian $\mathbf{L}_G$ in Laplacian $\mathbf{L}_H$ (and the corresponding graph energy) as:

$$(1-\epsilon)\lambda_i^G \leq \lambda_i^H \leq (1+\epsilon)\lambda_i^G, \quad \forall i = 1, \ldots, n$$

Below, we have modified the algorithm of [35] to initialize H by the "Maximum likelihood" Spanning Tree (MST), so as to include the $(n-1)$ most probable edges (and avoid the possibility of H being disjoint), as well as instead of adding $q = (\frac{0.09nlogn}{\epsilon^2} - (n-1))$ edges to the MST by analogous random trials, which would require more computations and memory, we compute the expected times an edge would have been selected if we had performed a random sampling of q trials (by multiplying the probability to select an edge p_{e_i} with the total number of trials q and round their product to the nearest integer) and then add those edges, that the expected times to be selected is greater than 0, to the MST (lines 8-10). By doing so, we also remove the randomness from the algorithm of [35] and thus, the modified algorithm, shown in Algorithm 4, results always into the same sparse graph.

Algorithm 4 For a dense graph $G = (V, E, w)$ with Laplacian matrix $\mathbf{L}_G$, construct a sparse subgraph $H = (V, \tilde{E}, \tilde{w})$ with Laplacian matrix $\mathbf{L}_H$

1: **function** H = SparLap(G, ϵ)
2: Calculate effective resistance $R^{eff}_{(i,j)}$ for each edge $(i,j) \in E$ (either exactly by (2.11) or approximately by Algorithm 5)
3: Set $p(i,j) = \frac{w_{(i,j)} R^{eff}_{(i,j)}}{n-1}$, $\forall (i,j) \in E$
4: $H = MST(G')$ where $G' = (V, E, p)$
5: Set $q = \frac{0.09nlogn}{\epsilon^2} - (n-1)$
6: **for** $i = 1$ `to` $|E|$ **do**
7: $\tilde{w}_{e_i} = round(p_{e_i} * q) \frac{w_{e_i}}{q * p_{e_i}}$
8: **if** $\tilde{w}_{e_i} \neq 0$ **then**
9: add e_i in H with weight $\tilde{w}_{e_i}$
10: **end if**
11: **end for**
12: **end function**

The most expensive operation in Algorithm 4 is the computation of effective resistances, which involves the pseudoinverse $\mathbf{L}_G^+$ of $\mathbf{L}_G$. To alleviate the computational cost, we also describe a procedure given in [35] for the approximate calculation of

effective resistances. Firstly, notice that the effective resistance (2.11) can be written as:

$$\begin{aligned} R^{eff}_{(i,j)} &= (\mathbf{e}_i - \mathbf{e}_j)^T \mathbf{L}_G^+ (\mathbf{e}_i - \mathbf{e}_j) \\ &= (\mathbf{e}_i - \mathbf{e}_j)^T \mathbf{L}_G^+ \mathbf{L}_G \mathbf{L}_G^+ (\mathbf{e}_i - \mathbf{e}_j) \\ &= ((\mathbf{e}_i - \mathbf{e}_j)^T \mathbf{L}_G^+ \mathbf{A}_G \mathbf{W}^{1/2})(\mathbf{W}^{1/2} \mathbf{A}_G^T \mathbf{L}_G^+ (\mathbf{e}_i - \mathbf{e}_j)) \\ &= \left\| \mathbf{W}^{1/2} \mathbf{A}_G^T \mathbf{L}_G^+ (\mathbf{e}_i - \mathbf{e}_j) \right\|_2^2 \end{aligned}$$

Thus, the effective resistance of edge (i, j) equals the squared Euclidean distance between the $d \times 1$ vectors $\mathbf{W}^{1/2} \mathbf{A}_G^T \mathbf{L}_G^+ \mathbf{e}_i$ and $\mathbf{W}^{1/2} \mathbf{A}_G^T \mathbf{L}_G^+ \mathbf{e}_j$ where d is the total number of edges (these are effectively the columns i and j of the $d \times n$ matrix $\mathbf{W}^{1/2} \mathbf{A}_G^T \mathbf{L}_G^+$). Now, a result proven in [36] states that if we project a set of n vectors $d \times 1$ (like the columns of matrix $\mathbf{W}^{1/2} \mathbf{A}_G^T \mathbf{L}_G^+$) onto the k-dimensional subspace spanned by the columns of a random matrix $\mathbf{Q} \in \Re^{k \times d}$ with entries either $+\frac{1}{\sqrt{k}}$ or $-\frac{1}{\sqrt{k}}$, where $k = \lceil 24 \log n / \delta^2 \rceil$ for given δ, then the distances between the vectors in the set are preserved with tolerance $1 \pm \delta$, providing analogous bounds for the effective resistance, i.e.

$$\begin{aligned} (1-\delta) \left\| \mathbf{W}^{1/2} \mathbf{A}_G^T \mathbf{L}_G^+ (\mathbf{e}_i - \mathbf{e}_j) \right\|_2^2 &\leq \left\| \mathbf{Q} \mathbf{W}^{1/2} \mathbf{A}_G^T \mathbf{L}_G^+ (\mathbf{e}_i - \mathbf{e}_j) \right\|_2^2 \\ &\leq (1+\delta) \left\| \mathbf{W}^{1/2} \mathbf{A}_G^T \mathbf{L}_G^+ (\mathbf{e}_i - \mathbf{e}_j) \right\|_2^2, \quad \forall i, j = 1 \ldots n \end{aligned}$$

$$\Rightarrow (1-\delta) R^{eff}_{(i,j)} \leq \left\| \mathbf{Q} \mathbf{W}^{1/2} \mathbf{A}_G^T \mathbf{L}_G^+ (\mathbf{e}_i - \mathbf{e}_j) \right\|_2^2 \leq (1+\delta) R^{eff}_{(i,j)}$$

The product $\mathbf{Q} \mathbf{W}^{1/2} \mathbf{A}_G^T \mathbf{L}_G^+$ can be computed as the matrix $\mathbf{Z}$ such that $\mathbf{Z} \mathbf{L}_G = \mathbf{Q} \mathbf{W}^{1/2} \mathbf{A}_G^T$. This entails solving only $k = O(logn)$ linear systems $\mathbf{L}_G \mathbf{z}_i = \mathbf{y}_i$ where $\mathbf{z}_i$ and $\mathbf{y}_i$ are the i-th rows of $\mathbf{Z}$ and $\mathbf{Y} = \mathbf{Q} \mathbf{W}^{1/2} \mathbf{A}_G^T$ respectively (instead of n systems that are required for computing $\mathbf{L}_G^+$). The solution of these linear systems in the Laplacian $\mathbf{L}_G$ can be performed efficiently by Laplacian iterative solvers (*LapSolve*) such as [37] and [38], which rely on Preconditioned Conjugate Gradients (PCG) to give the unique solution $\mathbf{z}_i = \mathbf{L}_G^+ \mathbf{y}_i$ that lies in the column space $\mathcal{R}(\mathbf{L}_G)$. The whole algorithm for approximate computation of effective resistances is given in Algorithm

5.

Algorithm 5 Approximate computation of effective resistances for edges of graph $G = (V, E, w)$ with Laplacian $\mathbf{L}_G$

function $R^{eff}_{(i,j)} = \text{APPROXREFF}(\mathbf{L}_G, \delta)$
  $k = \lceil 24 \log n/\delta^2 \rceil$
  Construct a $k \times d$ random matrix $\mathbf{Q}$ whose entries are either $+\frac{1}{\sqrt{k}}$ or $-\frac{1}{\sqrt{k}}$
  $\mathbf{Y} = \mathbf{Q}\mathbf{W}^{1/2}\mathbf{A}_G^T$
  $\tilde{\mathbf{z}}(i,:) = LapSolve(\mathbf{L}_G, \mathbf{y}(i,:)),\ \forall i = 1 \ldots k$
  $R^{eff}_{(i,j)} = \left\| \tilde{\mathbf{Z}}(\mathbf{e}_i - \mathbf{e}_j) \right\|_2^2,\ \forall (i,j) \in E$
end function

2.4 Proposed Methodology for Sparsification of Dense MOR Models

Combining all the algorithms presented in the previous sections, the complete methodology for sparsifying dense MOR models is given in Algorithm 6. The sparsity-accuracy tradeoff ϵ is usually set at its smallest possible value $\epsilon = \frac{1}{\sqrt{r}}$ (where r is the order of the reduced model).

Algorithm 6 Sparsification of dense models resulting from MOR

function $\widehat{\mathbf{G}}_{sp}, \widehat{\mathbf{C}}_{sp}, \widehat{\mathbf{\Gamma}}_{sp} = \text{MORSPARSE}(\widehat{\mathbf{G}}_n, \widehat{\mathbf{C}}_n, \widehat{\mathbf{\Gamma}}, \epsilon)$
  $\mathbf{G}_{DD} = prjSDD(\widehat{\mathbf{G}}_n, \text{1e-6})$
  $\mathbf{C}_{DD} = prjSDD(\widehat{\mathbf{C}}_n, \text{1e-6})$
  $\mathbf{\Gamma}_{DD} = prjSDD(\widehat{\mathbf{\Gamma}}, \text{1e-6})$
  Apply conversion (2.10) to $\mathbf{G}_{DD}$, $\mathbf{C}_{DD}$ and $\mathbf{\Gamma}_{DD}$, to obtain the matrices $\mathbf{G}_M$, $\mathbf{C}_M$ and $\mathbf{\Gamma}_M$ respectively
  Extract Laplacian part of $\mathbf{G}_M$, $\mathbf{C}_M$ and $\mathbf{\Gamma}_M$ in matrices $\mathbf{L}_G$, $\mathbf{L}_C$ and $\mathbf{L}_\Gamma$, and store diagonal remainder of strictly diagonally dominant rows in matrices $\mathbf{D}_G$, $\mathbf{D}_C$ and $\mathbf{D}_\Gamma$
  $\widehat{\mathbf{G}}_{sp} = SparLap(\mathbf{L}_G, \epsilon) + \mathbf{D}_G$
  $\widehat{\mathbf{C}}_{sp} = SparLap(\mathbf{L}_C, \epsilon) + \mathbf{D}_C$
  $\widehat{\mathbf{\Gamma}}_{sp} = SparLap(\mathbf{L}_\Gamma, \epsilon) + \mathbf{D}_\Gamma$
end function

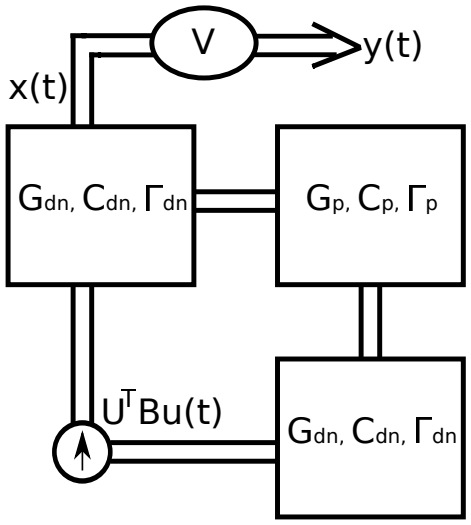


Figure 2-3: Two copies of circuit corresponding to $\mathbf{C}_{dn}$, $\mathbf{G}_{dn}$ and $\mathbf{\Gamma}_{dn}$ connected by circuit branches corresponding to $\mathbf{C}_p$, $\mathbf{G}_p$ and $\mathbf{\Gamma}_p$. The inputs to the reduced-order model are not the original inputs, but the inputs from the projection $\mathbf{U}^T\mathbf{Bu}(t)$.

2.5 Synthesis of MORSparse Models

An additional benefit of the matrices resulting from the application of Algorithm 6 (beyond sparsity) is that they are in standard MNA second order form, and thus can be readily synthesized into RLC circuit with only positive elements.

The approximation of the projected MOR matrices by circuit-type M-matrices can be considered as a procedure of finding the nearest (in numerical value sense) realizable circuit to the MOR model matrices which do not have necessarily a physical correspondence to a circuit (as they just result from an algebraic projection of the original model matrices (see 2.6)). Moreover, the sparsification algorithm *SparLap* does not violate the circuit-type M-matrix property (diagonally dominant matrix with negative off-diagonal) of the given matrix for sparsification. Therefore, the resulting sparse model matrices can be still realized into a circuit by following the inverse MNA procedure [39] (see Fig. 2-4).

Below we demonstrate how any model of SDD matrices as those resulting from steps 2, 3, 4 in Algorithm 6 can be synthesized into a circuit with only positive elements. Let

$$\mathbf{G}_{DD} = \begin{bmatrix} 0.92 & 0.11 & -0.71 & 0 \\ 0.11 & 1 & -0.3 & 0.59 \\ -0.71 & -0.3 & 1.21 & 0 \\ 0 & 0.59 & 0 & 1.59 \end{bmatrix},$$

$$\mathbf{C}_{DD} = \begin{bmatrix} 1 & 0.72 & 0 & 0 \\ 0.72 & 1.63 & 0 & -0.91 \\ 0 & 0 & 1 & 0 \\ 0 & -0.91 & 0 & 1 \end{bmatrix},$$

$$\mathbf{\Gamma}_{DD} = \begin{bmatrix} 0.12 & 0 & 0 & 0.12 \\ 0 & 0.3 & 0 & -0.2 \\ 0 & 0 & 0.5 & 0 \\ 0.12 & -0.2 & 0 & 0.32 \end{bmatrix}$$

Now, by applying conversion (2.10) to $\mathbf{G}_{DD}$, $\mathbf{C}_{DD}$ and $\mathbf{\Gamma}_{DD}$ we obtain

$$\mathbf{G}_{M} = \begin{bmatrix} 0.92 & 0 & -0.71 & 0 & 0 & -0.11 & 0 & 0 \\ 0 & 1 & -0.3 & 0 & -0.11 & 0 & 0 & -0.59 \\ -0.71 & -0.3 & 1.21 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.59 & 0 & -0.59 & 0 & 0 \\ 0 & -0.11 & 0 & 0 & 0.92 & 0 & -0.71 & 0 \\ -0.11 & 0 & 0 & -0.59 & 0 & 1 & -0.3 & 0 \\ 0 & 0 & 0 & 0 & -0.71 & -0.3 & 1.21 & 0 \\ 0 & -0.59 & 0 & 0 & 0 & 0 & 0 & 1.59 \end{bmatrix},$$

$$\mathbf{C}_M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & -0.72 & 0 & 0 \\ 0 & 1.63 & 0 & -0.91 & -0.72 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.91 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -0.72 & 0 & 0 & 1 & 0 & 0 & 0 \\ -0.72 & 0 & 0 & 0 & 0 & 1.63 & 0 & -0.91 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0.91 & 0 & 1 \end{bmatrix},$$

$$\mathbf{\Gamma}_M = \begin{bmatrix} 0.12 & 0 & 0 & 0 & 0 & 0 & 0 & -0.12 \\ 0 & 0.3 & 0 & -0.2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & 0 & 0 & 0 & 0 \\ 0 & -0.2 & 0 & 0.32 & -0.12 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.12 & 0.12 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.3 & 0 & -0.2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & 0 \\ -0.12 & 0 & 0 & 0 & 0 & -0.2 & 0 & 0.32 \end{bmatrix}$$

The above model is in standard MNA form and thus the non-zero off-diagonal elements from the upper-diagonal (or lower-diagonal, because they are symmetric matrices) parts of $\mathbf{G}_M$, $\mathbf{C}_M$ and $\mathbf{\Gamma}_M$, let g_{ij}, c_{ij} and γ_{ij}, can be interpreted into a resistor, a capacitor and an inductor with resistance $\frac{-1}{g_{ij}}$, capacitance $-c_{ij}$ and inductance $\frac{-1}{\gamma_{ij}}$ respectively between nodes i and j, while the diagonal entries of $\mathbf{G}_M$, $\mathbf{C}_M$ and $\mathbf{\Gamma}_M$, let g_{ii}, c_{ii} and γ_{ii}, can be interpreted into a resistor, a capacitor and an inductor with resistance $\frac{1}{g_{ii} - \sum\limits_{\substack{j=1,\\ j\neq i}}^{8} |g_{ij}|}$, capacitance $c_{ii} - \sum\limits_{\substack{j=1,\\ j\neq i}}^{8} |c_{ij}|$ and inductance $\frac{1}{\gamma_{ii} - \sum\limits_{\substack{j=1,\\ j\neq i}}^{8} |\gamma_{ij}|}$ respectively between node i and ground. Therefore, the circuit synthesis of our example leads into the circuit shown in Figure 2-4.

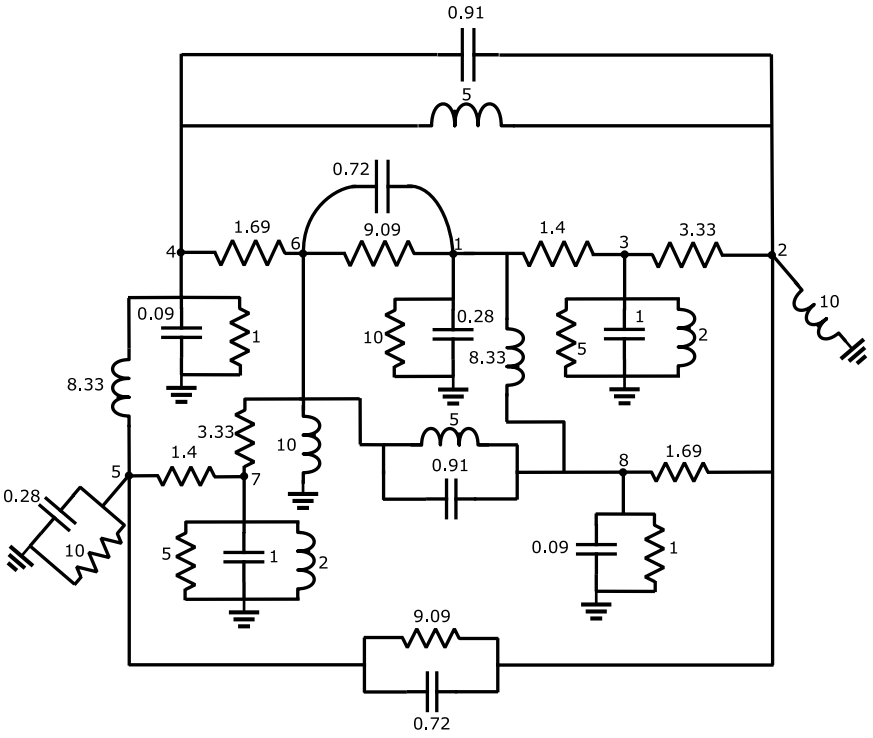


Figure 2-4: The synthesized circuit of the example. The enumeration of the nodes in the above circuit has been performed so that node i corresponds to the i-th column/row of $\mathbf{G}_M$, $\mathbf{C}_M$ and $\mathbf{\Gamma}_M$.

2.6 Experimental Evaluation

For the experimental evaluation of MORSparse presented in Algorithm 6 we consider a set of synthetic RLC interconnects in a 3d grid topology extracted from FastHenry[46]. The RLC grids in one dimension are consisted of RL branches and in the other two are consisted of resistive only branches, while we assume a capacitive branch to the ground at their intersection points (see Fig. 2-5). Moreover, we assume a set of RC netlists extracted from real designs [40] namely a Transmission Line (TL), a Low Noise Amplifier (LNA), an RF mixer (MX3) and an RC interconnect (RCintc). We provide in Table 2.1 all the structural details of each benchmark.

Firstly, in all experiments we formulate the system of equations as in (2.5) (of course without the term $\mathbf{\Gamma} \int_0^t \mathbf{v}(t)$ in case of RC benchmarks), seeking for the unknown node voltages, and then obtain a reduced order model (ROM) using SAPOR [29], for RLC benchmarks and PACT for RC benchmarks. We implemented SAPOR, PACT

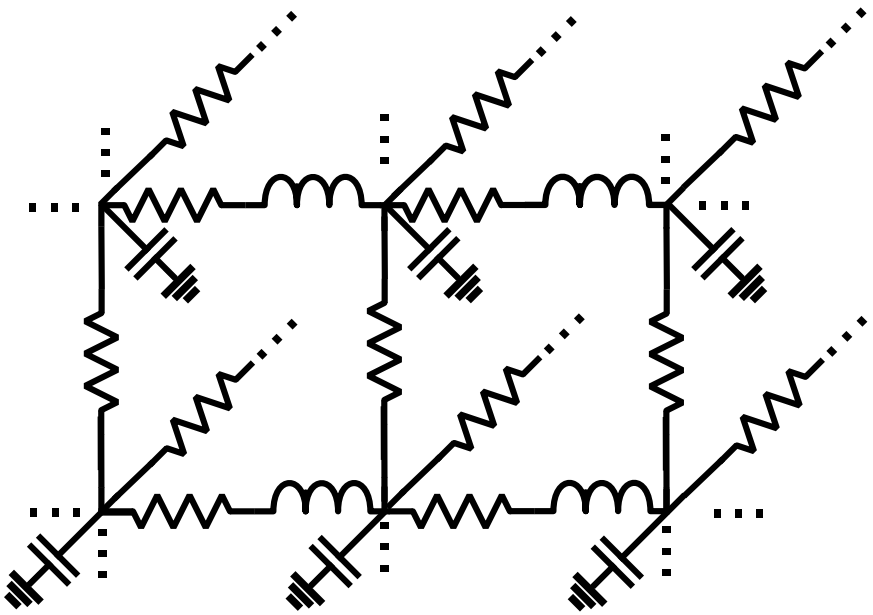


Figure 2-5: Structure of our synthetic RLC benchmarks

Table 2.1: Structural details of benchmarks under test

Bench.	**num. of RL branches**	**num. of ports**	**num. of nodes**
interc1	256	32	544
interc2	768	96	1632
interc3	2048	256	4352
interc4	4096	512	8704
TL	NA	22	3253
LNA	NA	79	29885
MX3	NA	110	867
RCintc	NA	646	16862

(see **Appendix A**) and the proposed MORSparse algorithm in MATLAB 2017a, and ran all our experiments on a system equipped with a 3.60 GHz Intel Core i7 CPU and 16 GB memory. In Table 2.2 we report the relative distance between the reduced and the DD-projected matrices in the spectral and the Frobenius norm $\left(\frac{||\mathbf{G}_{DD}-\widehat{\mathbf{G}}_n||}{||\widehat{\mathbf{G}}_n||}, \frac{||\mathbf{C}_{DD}-\widehat{\mathbf{C}}_n||}{||\widehat{\mathbf{C}}_n||}, \frac{||\mathbf{\Gamma}_{DD}-\widehat{\mathbf{\Gamma}}||}{||\widehat{\mathbf{\Gamma}}||}\right)$ for all the benchmarks we tested. It is apparent that the matrices resulting from the projection of MOR matrices to the nearest SDD ones constitute very good approximations, especially when the ROM comes from PACT (Notice the 0 distance of $\widehat{\mathbf{G}}_n$ from $\mathbf{G}_{DD}$ in case of RC benchmarks, as we avoid to project it to the set of DD matrices because it is DD in the first place).

Table 2.2: Relative distance between the reduced order and the DD-projected matrices in the spectral and the Frobenius norm

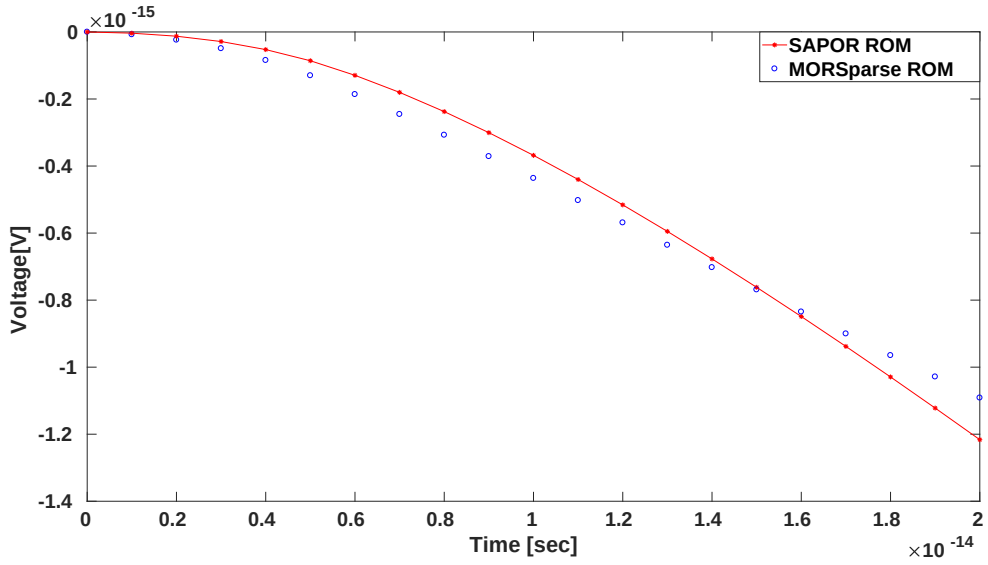
Bench	**Spectral distance**			**Frobenius distance**		
	$\mathbf{G}_n$	$\mathbf{C}_n$	$\mathbf{\Gamma}$	$\mathbf{G}_n$	$\mathbf{C}_n$	$\mathbf{\Gamma}$
interc1	0.26	0.34	0.08	0.26	0.26	0.05
interc2	0.3	0.35	0.16	0.27	0.26	0.08
interc3	0.24	0.4	0.08	0.23	0.27	0.04
interc4	0.21	0.47	0.13	0.2	0.28	0.04
TL	0	1.02e-4	NA	0	1.12e-4	NA
LNA	0	0.021	NA	0	0,029	NA
MX3	0	0.038	NA	0	0,046	NA
RCintc	0	0.55	NA	0	0.54	NA

Table 2.3: Comparison of sparse ROMs obtained with MORSparse against dense ROMs from SAPOR and PACT

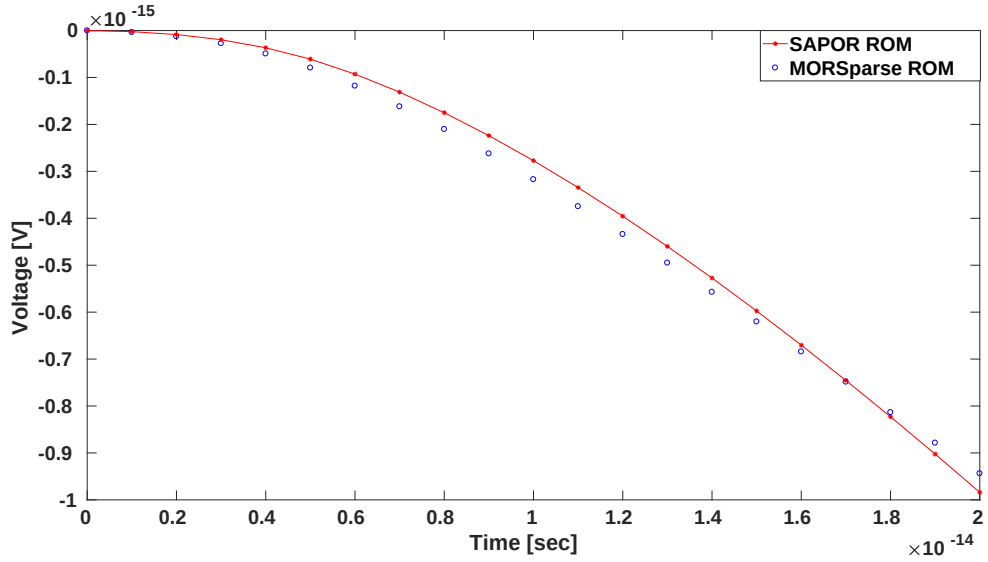
Bench	**Order of ROM**	**Sparsity of SAPOR/PACT ROM**			**Sparsity of MORSparse ROM**			**Sparsif. time**	**Sim. time dense ROM**	**Sim. time sparse ROM**	**Speedup**
		$\eta_{\mathbf{G}_n}$	$\eta_{\mathbf{C}_n}$	$\eta_{\mathbf{\Gamma}}$	$\eta_{\mathbf{G}_{sp}}$	$\eta_{\mathbf{C}_{sp}}$	$\eta_{\mathbf{\Gamma}_{sp}}$				
interc1	64	0%	0%	0%	95.4%	89%	94%	3.5s	12s	41s	1.5×
interc2	192	0%	0%	0%	98.4%	89.5%	97.4%	22.5s	55s	42s	x1.48×
interc3	512	0%	0%	0%	99.3%	95.3%	98.4%	83s	1300s	43s	x5.6×
interc4	1024	0%	0%	0%	99.7%	94.2%	98.3%	657s	4901s	44s	x4.6×
TL	22	78.9%	0%	NA	95.9%	78.3%	NA	0.89s	1.82s	3.3s	x0.55×
LNA	79	95.2%	66.7%	NA	98.7%	94.0%	NA	0.32s	3.1s	2.4s	x1.3×
MX3	110	97.3%	80.6%	NA	99.1%	94.2%	NA	0.41s	5.3s	3.2s	x1.65×
RCintc	663	96.7%	69.7%	NA	99.4%	97.0%	NA	4.5s	129s	31.9s	x4.0×

After the sparsification of the dense SAPOR and PACT ROMs, we examined the resulting MORSparse ROMs in terms of sparsity ratio ($\eta = \frac{\#zeros}{\#rows \times \#cols}$) and simulation accuracy. In all our experiments we applied at all ports a 1A step current source, executed the simulation for 10^6 timepoints and acquired the response at a randomly chosen port. Fig. 2-6 and Fig. 2-7 compare the voltage responses at a port of benchmarks interc1, interc3, when we assess the accuracy of RLC benchmarks and LNA, TL when we assess the accuracy of RC benchmarks, acquired from the simulation of SAPOR ROM and PACT ROM respectively and the simulation of MORSparse ROM for the smallest acceptable value of ϵ. It can be verified that the responses of MORSparse ROMs approximate the responses of SAPOR and PACT ROMs very well. Again see that the spare model when the ROM has been obtained from PACT achieves to match the waveform of the initial dense ROM almost perfectly, due to the avoidance of projecting $\widehat{\mathbf{G}}_n$ to the space of DD matrices.

Table 2.3 compares the SAPOR and the PACT ROMs with the MORSparse ROMs



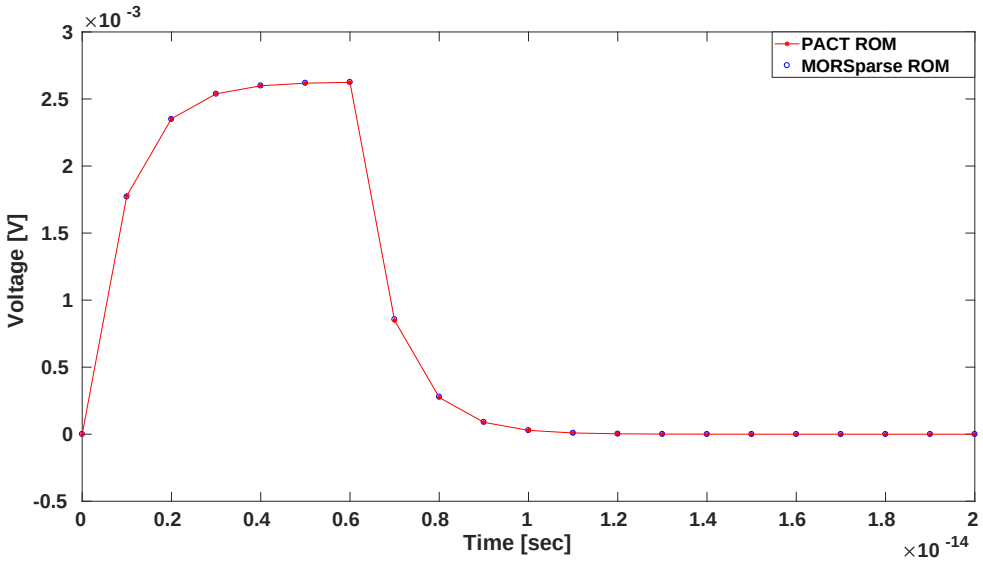
(a) Voltage response at a port of benchmark interc1. The rms value of the error between the two waveform is 5.55e-17.



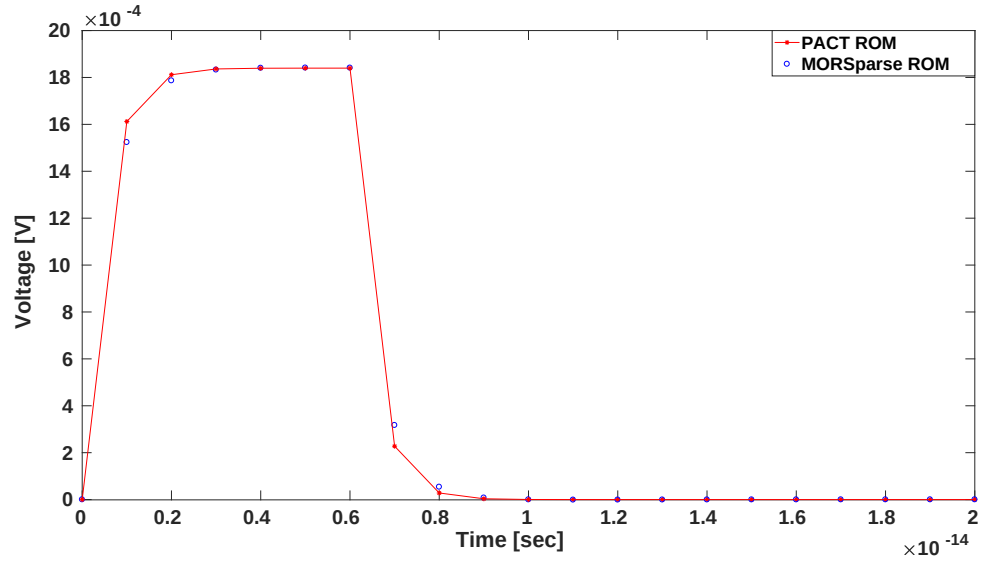
(b) Voltage response at a port of benchmark interc3. The rms value of the error between the two waveform is 2.64e-17.

Figure 2-6: Voltage response from simulation of the dense ROMs obtained from SAPOR and their sparse counterparts obtained with MORSparse.

in terms of sparsity and simulation time. We can observe that a sparsity ratio of over 89% for the resulting sparse ROMs was attained that led to speedups from x1.48 to x5.6 in simulation, (which are expected to increase for ROMs in the order of a few



(a) Voltage response at a port of benchmark TL. The rms value of the error between the two waveform is 7.9e-7.



(b) Voltage response at a port of benchmark LNA. The rms value of the error between the two waveform is 1.18e-7.

Figure 2-7: Voltage response from simulation of the dense ROMs obtained from PACT and their sparse counterparts obtained with MORSparse.

thousand encountered in practice. Actually, in case of TL we had a slowdown because it is too small in size), while the sparsification time is an one-time cost that does not constitute significant fraction of the total simulation time in practical scenarios.

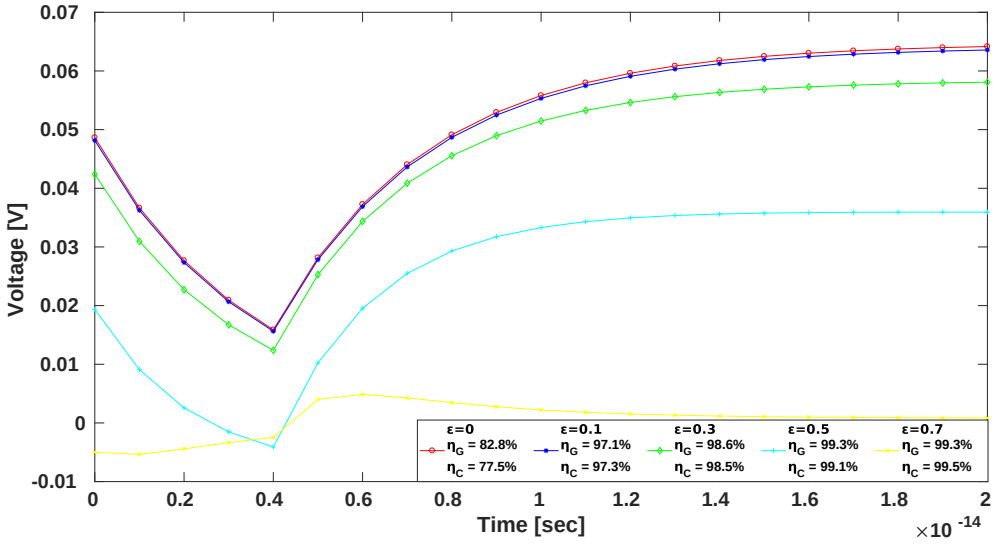


Figure 2-8: Voltage responses from simulation of the sparse MX3 ROM obtained with different ϵ values.

MORSparse offers simulation accuracy versus sparsity ratio trade-offs through the parameter ϵ. In an additional experiment we sparsified the ROM of the MX3 with different ϵ values. Fig. 2-8 shows that smaller values of ϵ lead to more accurate simulation but denser ROMs, as was naturally expected.

Chapter 3

Efficient Manipulation of Dense Inductance Matrix in Simulation of Large Mutually Inductive Circuits

3.1 Background

3.1.1 Transient analysis overview

Consider an RLC circuit composed of n nodes and m inductive branches with mutual inductive coupling between them (see Fig. 3-1), as well as its Modified Nodal Analysis (MNA) description [20]:

$$\widetilde{\mathbf{G}}\mathbf{x}(t) + \widetilde{\mathbf{C}}\dot{\mathbf{x}}(t) = \widetilde{\mathbf{e}}(t), \tag{3.1}$$

where

$$\widetilde{\mathbf{G}} = \begin{bmatrix} \mathbf{G}_n & \mathbf{A}_L \\ -\mathbf{A}_L^T & \mathbf{0} \end{bmatrix}, \widetilde{\mathbf{C}} = \begin{bmatrix} \mathbf{C}_n & \mathbf{0} \\ \mathbf{0} & \mathbf{L} \end{bmatrix}, \mathbf{x}(t) = \begin{bmatrix} \mathbf{v}(t) \\ \mathbf{i}(t) \end{bmatrix}, \widetilde{\mathbf{e}}(t) = \begin{bmatrix} \mathbf{e}(t) \\ \mathbf{0} \end{bmatrix}.$$

In the above, $\mathbf{G}_n \in \Re^{n\times n}$ and $\mathbf{C}_n \in \Re^{n\times n}$ are the node conductance and node capacitance matrices respectively, $\mathbf{L} \in \Re^{m\times m}$ is the dense inductance matrix (with

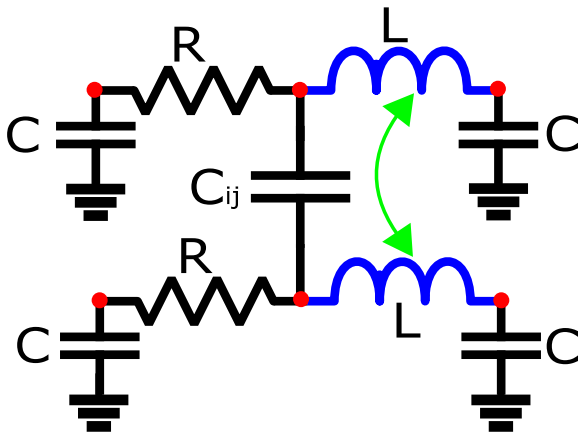


Figure 3-1: Example of an RLC circuit with 6 nodes (red) and 2 inductive branches (blue). The coupling between the two inductances is indicated in green.

self-inductances as diagonal entries and mutual inductances as off-diagonal entries), $\mathbf{A}_L \in \Re^{n\times m}$ is the corresponding node-to-branch incidence matrix, $\mathbf{v}(t) \in \Re^n$ and $\mathbf{i}(t) \in \Re^m$ are the vectors of the unknown node voltages and branch currents, and $\mathbf{e}(t) \in \Re^n$ is the vector of excitations from independent sources at the nodes (assuming, without loss of generality, that voltage sources have been transformed to Norton-equivalent current sources). Applying the Backward-Euler numerical integration method in (3.1), we arrive at the problem of solving a system of $n+m$ linear algebraic equations at each discrete time $t_k, k = 1, 2, \ldots$ (starting from initial values $\mathbf{x}(t_0) = \begin{bmatrix}\mathbf{v}(0) & \mathbf{i}(0)\end{bmatrix}^T$ of the unknown variables):

$$\mathbf{J}_k \mathbf{x}(t_k) = \mathbf{b}(t_k), \tag{3.2}$$

where

$$\mathbf{J}_k = \begin{bmatrix} \frac{1}{h_k}\mathbf{C}_n + \mathbf{G}_n & \mathbf{A}_L \\ -\mathbf{A}_L^T & \frac{1}{h_k}\mathbf{L} \end{bmatrix}, \quad \mathbf{b}(t_k) = \widetilde{\mathbf{e}}(t_k) + \frac{\widetilde{\mathbf{C}}}{h_k}\mathbf{x}(t_{k-1})$$

and $h_k = t_k - t_{k-1}$, $k = 1, 2, \ldots$ is the chosen time-step size (which can be either fixed or variable during the analysis). The above is a linear system of the form $\mathbf{Ax} = \mathbf{b}$ that has to be solved at every discrete time t_k, $k = 1, 2, \ldots$.

3.1.2 Iterative linear solvers

Direct methods (based on matrix factorization) are not practically feasible for solving large dense systems or systems with a large dense block (like $\frac{1}{h_k}\mathbf{L}$ in (3.2)), due to their excessive runtime and memory requirements. The only viable option for such systems is the use of iterative methods, and particularly Krylov-subspace iterative methods like GMRES (which is suitable for general unsymmetric systems like (3.2)) [41]. The operation with the dominant cost inside the iteration loop of Krylov-subspace methods (and GMRES in particular) is the matrix-vector multiplication, which for the system (3.2) can be dealt efficiently by the hierarchical matrix framework that is introduced in the next subsection and applied to the inductance matrix $\mathbf{L}$ in Section 3.2. Other operations of Krylov-subspace iterative methods like inner products, scalar-vector products and vector additions are not expensive computationally.

The convergence rate of Krylov-subspace methods is determined by the spread of the eigenvalues of the system matrix and their distance form 1 [42]. In particular, convergence is fast when the eigenvalues are tightly clustered together and slow when they are spread apart. A slow convergence rate can be alleviated by using a preconditioner matrix $\mathbf{M}$ and the equivalent solution of the system $\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}$. The application of $\mathbf{M}$ can be embedded inside the iterative method as the computation of the preconditioned residual $\mathbf{z}_j$, via the solution of the system $\mathbf{M}\mathbf{z}_j = \mathbf{r}_j$ in every iteration j (see Fig. 3-2). An effective preconditioner needs to satisfy the following two prerequisites:

- Preconditioning should deliver a much tighter clustering of the eigenvalues of $\mathbf{M}^{-1}\mathbf{A}$ than those of the original matrix $\mathbf{A}$, leading to a significant reduction in iteration count and acceleration of the convergence rate.

- The reduction in the number of iterations should offset the computational overhead introduced by the solution of $\mathbf{M}\mathbf{z}_j = \mathbf{r}_j$ in every iteration.

Inputs: System matrix **A**, preconditioner **M**, and RHS vector **b**

Output: Solution **x** of $\mathbf{Ax} = \mathbf{b}$

1: Set $\mathbf{x} := \mathbf{x}^{(0)}$ (initial guess); $\mathbf{r} := \mathbf{b} - \mathbf{Ax}^{(0)}$ (initial residual)

2: $\mathbf{p} := \mathbf{r}$ or $\mathbf{p} := \mathbf{M}^{-1}\mathbf{r}$ (initial search direction)

3: **while** not_converged

4: …

5: $\mathbf{q} := \mathbf{Ap}$

6: Update **x** and **r** (using **p** and **q**)

7: …

8: Solve $\mathbf{Mz} = \mathbf{r}$

9: Update **p** (using **z**)

10: …

11: **end while**

Figure 3-2: The general structure of Krylov-subspace iterative methods.

3.1.3 Low-rank products and hierarchical matrices

If $\mathbf{A} \in \Re^{n \times n}$ is a square matrix or matrix block with Singular Value Decomposition (SVD) $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{\Sigma} = diag(\sigma_1, \ldots, \sigma_n)$ and $\sigma_1 > \cdots > \sigma_n$, then by introducing the following partition:

$$\mathbf{U} = [\mathbf{U}_1 \quad \mathbf{U}_2], \quad \mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_2 \end{bmatrix}, \quad \mathbf{V}^T = \begin{bmatrix} \mathbf{V}_1^T \\ \mathbf{V}_2^T \end{bmatrix}$$

with $\mathbf{U}_1 \in \Re^{n\times r}$, $\mathbf{\Sigma}_1 \in \Re^{r\times r}$ and $\mathbf{V}_1^T \in \Re^{r\times n}$, the optimal low-rank product approximation of rank-r of $\mathbf{A}$ is defined as $\widetilde{\mathbf{A}} = (\mathbf{U}_1\mathbf{\Sigma}_1^{1/2})(\mathbf{V}_1\mathbf{\Sigma}_1^{1/2})^T \equiv \mathbf{Z}\mathbf{Y}^T$. It has been proven in [43] that this approximation satisfies the following optimization problem:

$$\min_{\widetilde{\mathbf{A}}} \|\mathbf{A} - \widetilde{\mathbf{A}}\| \quad \text{s.t. } \text{rank}(\widetilde{\mathbf{A}}) = r$$

for any common matrix norm. The benefit of the factorization $\widetilde{\mathbf{A}} = \mathbf{Z}\mathbf{Y}^T$ with $\mathbf{Z} = \mathbf{U}_1\mathbf{\Sigma}_1^{1/2}$ and $\mathbf{Y} = \mathbf{V}_1\mathbf{\Sigma}_1^{1/2}$ is that only the factors $\mathbf{Z}$ and $\mathbf{Y}$ have to be kept in memory instead of the whole $n \times n$ matrix $\mathbf{A}$ (see Fig. 3-3). The above low-rank product approximation can be straightforwardly extended to rectangular matrix blocks.

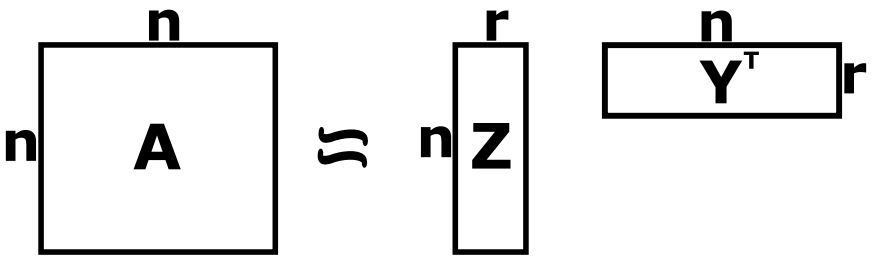


Figure 3-3: Approximation of a $n \times n$ dense block with a low rank product. Only $\mathcal{O}(2rn)$ storage is required, while parameter r controls the accuracy of the approximation.

Hierarchical matrices or $\mathcal{H}$-matrices [44] are a lossy compressed matrix format which relies on the partitioning of a dense matrix into a number of sub-matrix blocks that can be approximated efficiently and accurately by low-rank products. The special structure of $\mathcal{H}$-matrices allows the development of algorithms for the basic operations of matrix-vector multiplication and matrix factorization with near optimal asymptotic complexity.

3.2 Approximation of the Inductance Matrix with Hierarchical Matrices

Because of the natural fact that interconnect segments which are farther apart exhibit weaker mutual inductive interactions, the inductance matrix **L** will be characterized by progressively smaller off-diagonal elements while moving away from the diagonal (assuming that segments in close physical proximity are enumerated consecutively - otherwise suitable permutation matrices can be applied). Then the matrix blocks that are away from the diagonal can be efficiently approximated by low-rank products and the whole inductance matrix by an appropriate $\mathcal{H}$-matrix (see Fig. 3-4). The size and the number of blocks, as well as the order of the low-rank approximation of each block, constitute trade-off parameters between the degree of compression and the quality of approximation.

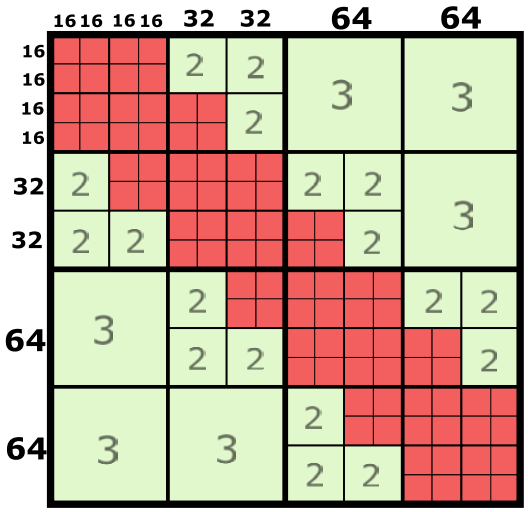


Figure 3-4: Example of 256×256 inductance matrix in $\mathcal{H}$-matrix format. Blocks around diagonal (red-colored blocks) correspond to mutual inductances in close physical proximity to each other. Blocks away form the diagonal (green-colored blocks) have progressively smaller numerical values and can be approximated by low-rank products (the rank of each block is indicated inside the block).

It is noted that the usage of $\mathcal{H}$-matrix format does not require the *a-priori* knowledge of the whole inductance matrix. Instead, only the spatial arrangement of the interconnects, which is available before the assembly of the inductance matrix, is re-

quired to perform the matrix blocking. Thus, provided that we integrate the $\mathcal{H}$-matrix library with the inductance extraction tool, we can approximate a matrix block by a low-rank product immediately after its computation, and next store it directly in $\mathcal{H}$-matrix format rather than as part of a dense inductance matrix.

3.3 Solution and Preconditioning of the Transient Linear System

3.3.1 Multiplication of transient system matrix with vector

The matrix $\mathbf{J}_k$ is composed of different storage formats, with $\mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n$ and $\mathbf{A}_L$ being in sparse format (compressed row or column form) and $\mathbf{L}$ in $\mathcal{H}$-matrix format. Thus, the multiplication of $\mathbf{J}_k$ with a vector inside the iteration loop of a Krylov-subspace method like GMRES, can be performed in a block fashion by calling the appropriate sparse and $\mathcal{H}$-matrix subroutines (see **Appendix B**).

3.3.2 Preconditioner formulation

Consider the block LU factorization of $\mathbf{J}_k$ of (3.2):

$$\mathbf{J}_k = \mathbf{L_J U_J} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{A}_L^T(\mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n)^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n & \mathbf{A}_L \\ \mathbf{0} & \mathbf{S} \end{bmatrix},$$

where

$$\mathbf{S} = \frac{1}{h_k}\mathbf{L} + \mathbf{A}_L^T(\mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n)^{-1}\mathbf{A}_L$$

is the Schur complement of $\mathbf{J}_k$. Because $\mathbf{L_J}$ is a block lower triangular matrix with identity blocks on the main diagonal, and thus has all eigenvalues equal to 1, the block upper triangular matrix

$$\mathbf{U_J} = \begin{bmatrix} \mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n & \mathbf{A}_L \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \tag{3.3}$$

constitutes an ideal preconditioner for $\mathbf{J}_k$ (meaning that GMRES will converge in only one iteration).

3.3.3 Preconditioner application

The preconditioning step of Fig. 3-2 involves the solution of the following linear system:

$$\mathbf{U_J z} = \mathbf{r} \Rightarrow \begin{bmatrix} \mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n & \mathbf{A}_L \\ \mathbf{0} & \mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix}. \tag{3.4}$$

The system (3.4) has a block upper triangular coefficient matrix and its solution can be achieved by block back substitution. Specifically, we first solve $\mathbf{S}\mathbf{z}_2 = \mathbf{r}_2$ (details are given in the next paragraph), then update the right hand side as $\mathbf{r}_1 = \mathbf{r}_1 - \mathbf{A}_L\mathbf{z}_2$, and finally solve $(\mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n)\mathbf{z}_1 = \mathbf{r}_1$. The latter is a $n \times n$ sparse linear system which can be solved by any direct or iterative sparse linear solver. Since it can be demonstrated that $\mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n$ is a symmetric diagonally dominant matrix with non-positive off-diagonal elements [21], it is recommended to use iterative methods for which very efficient preconditioners have been developed [38] [37].

The Schur complement $m \times m$ system $\mathbf{S}\mathbf{z}_2 = \mathbf{r}_2$ has coefficient matrix with two additive terms, $\mathbf{S}_1 \equiv \frac{1}{h_k}\mathbf{L}$ and $\mathbf{S}_2 \equiv \mathbf{A}_L^T(\mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n)^{-1}\mathbf{A}_L$, both of which are dense and in different matrix formats ($\mathbf{S}_1$ is stored as an $\mathcal{H}$-matrix). However, since the relative contributions of $\mathbf{S}_1$ and $\mathbf{S}_2$ depend on the time step size h_k, we can choose one term over the other for the typical range of step sizes in a variable-step simulation. Specifically, for small step sizes the term $\mathbf{S}_1 = \frac{1}{h_k}\mathbf{L}$ dominates and the system $\frac{1}{h_k}\mathbf{L}\mathbf{z}_2 = \mathbf{r}_2$ can be solved by LU factorization of $\mathbf{L}$ in $\mathcal{H}$-matrix format. For larger step sizes the term $\mathbf{S}_2 = \mathbf{A}_L^T(\mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n)^{-1}\mathbf{A}_L$ is dominant and the block preconditioner becomes

$$\begin{bmatrix} \mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n & \mathbf{A}_L \\ \mathbf{0} & \mathbf{A}_L^T(\mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n)^{-1}\mathbf{A}_L \end{bmatrix} \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix}$$

which can be straightforwardly derived to be equivalent to the system

$$\begin{bmatrix} \mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n & \mathbf{A}_L \\ -\mathbf{A}_L^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix}. \tag{3.5}$$

The latter is an $(n+m) \times (n+m)$ sparse linear system which can be solved by any direct or iterative linear solver.

3.3.4 Selection of the dominant term of Schur complement

The choice of the approximation of $\mathbf{S}$ as $\mathbf{S}_1$ or $\mathbf{S}_2$, for a given timestep, can be guided by quantifying the relative magnitude of each term in its spectral norm (see Algorithm 7). While there exist efficient routines to compute the spectral norm of $\frac{1}{h_k}\mathbf{L}$ in $\mathcal{H}$-matrix format, the spectral norm of $\mathbf{A}_L^T(\mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n)^{-1}\mathbf{A}_L$ can only be estimated by the inequality

Algorithm 7 Choice of suitable approximation of $\mathbf{S}$ for a given timestep h_k

function $\mathbf{S}$ = APPROX($\frac{1}{h_k}\mathbf{L}$, $\mathbf{A}_L^T(\mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n)^{-1}\mathbf{A}_L$)
  if $\left\|\frac{1}{h_k}\mathbf{L}\right\|_{sp} > \left\|\mathbf{A}_L^T(\mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n)^{-1}\mathbf{A}_L\right\|_{sp}$ **then**
    $\mathbf{S} \simeq \frac{1}{h_k}\mathbf{L}$
  else
    $\mathbf{S} \simeq \mathbf{A}_L^T(\mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n)^{-1}\mathbf{A}_L$
  end if
end function

$$||\mathbf{A}_L^T(\mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n)^{-1}\mathbf{A}_L||_{sp} \leq ||\mathbf{A}_L||_2||(\mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n)^{-1}||_{sp}||\mathbf{A}_L^T||_2. \tag{3.6}$$

Due to the specific structure of the interpolation matrix $\mathbf{A}_L$ the inequality (3.6) is fairly sharp and the estimate is getting better as h_k increases. With $||\mathbf{A}_L||_2 = \sqrt{2}$ we have

$$||\mathbf{A}_L^T(\mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n)^{-1}\mathbf{A}_L||_{sp} \leq 2||(\mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n)_h^{-1}||_{sp} = \frac{2}{\lambda_{min}(\mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n)}, \tag{3.7}$$

where $\lambda_{min}(\mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n)$ is the minimum eigenvalue of the matrix $\mathbf{G}_n + \frac{1}{h_k}\mathbf{C}_n$. Its estimate can be obtained efficiently by the inverse power iteration [45] where the matrix-vector product from the normal power method is replaced by the solution of a linear system.

3.4 Experimental Results

3.4.1 Experimental setup

For the experimental evaluation of the compression rates obtained by storing the inductance matrix $\mathbf{L}$ as a $\mathcal{H}$-matrix and the efficiency of the proposed preconditioning method we consider a set of parallel interconnect RLC models with inductive and capacitive coupling. The inductance matrix, with all mutual inductances, was assembled with FastHenry [46], while we assume that there is a capacitive path from all nodes to the ground and a capacitive coupling between adjacent nodes. Structural details of the benchmarks are summarized in Table 3.1. The simulation of the RLC interconnect models was performed with a driver resistance of 30Ω, load capacitance of 20fF, total wire self-capacitance of 40fF and total coupling capacitance between adjacent wires of 20fF. A 1V 20ps ramp voltage source was applied to one of the wires while the remaining were kept inactive. Our experimental framework was developed in C/C++ and all our experiments were performed on a system with a 3.60GHz Intel Core i7 CPU and 16GB memory. It is noted that since we used an off-the-shelf RLC extractor like FastHenry (rather than develop a custom extraction tool), we were forced to store the whole dense inductance matrix in memory before converting it to $\mathcal{H}$-matrix format, and thus were restricted in the largest circuit that could be handled

by the memory of the target machine.

Table 3.1: Structural details of the test problem.

Bench	**number of RL branches m**	**number of nodes n**	**Resistance (Ω) of each RL seg.**
bus1	256	544	3.3156
bus2	1024	2080	0.8289
bus3	4096	8208	0.1036
bus4	8192	16416	0.1036
bus5	16384	32832	0.1036

3.4.2 Efficiency of the compression of inductance matrix by $\mathcal{H}$-matrices

The $\mathcal{H}$-matrix approximation requires information on the geometrical coordinates of the interconnect branches, but it is pointed out that the physical structure of the circuit model can be arbitrary and is not restricted to specific configurations like parallel buses of conductors. In order to compress the dense inductance matrix by $\mathcal{H}$-matrices we adopt in this work the HLIBpro library [47][48][49]. HLIBpro offers the approximation routines to store a dense matrix in the $\mathcal{H}$-matrix format and perform a set of matrix operations with $\mathcal{H}$-matrices. The inputs to HLIBpro are the geometrical coordinates of the nodes of the physical structure. HLIBpro identifies the node indices that correspond to closer interconnects and performs a segmentation of $\mathbf{L}$ into blocks. For the clustering process of the coordinates into groups and, in turn, the segmentation of $\mathbf{L}$ into a number of blocks of certain size we used the default set of parameters of HLIBpro (as suggested in HLIBpro's manual [50]). Each block $\mathbf{A}_i$ was approximated by low rank product $\widetilde{\mathbf{A}}_i = \mathbf{Z}_i\mathbf{Y}_i^T$ such that $\|\mathbf{A}_i - \widetilde{\mathbf{A}}_i\|_F \leqslant 10^{-4}$. We tested the quality of matrix compression of $\mathbf{L}$ with hierarchical matrices (denoted as $\mathbf{L}_{\mathcal{H}}$) by solving a linear system $\mathbf{L}\overline{\mathbf{z}}_2 = \overline{\mathbf{y}}_2$ for multiple random right hand sides $\overline{\mathbf{y}}_2$ by Krylov-iterative method, preconditioned by $\mathbf{L}_{\mathcal{H}}$. In all our tests the Krylov-iterative method converged in 1-2 iterations meaning that $\mathbf{L}_{\mathcal{H}}$ is a close approximation of $\mathbf{L}$ and can be used instead of it in all subsequent calculations without a significant loss

of accuracy. Table 3.2 shows the memory savings, the speedups of matrix-vector multiplication required in the Krylov loop, as well as the time required to perform $\mathcal{H}$-matrix compression of $\mathbf{L}$. It is apparent that the larger the benchmark, the more memory savings is recorded because increasingly larger blocks can be approximated by low rank products. Moreover, the time to sparsify the dense inductance matrix in each benchmark with $\mathcal{H}$-matrices is sub-linear with respect to the total number of elements in $\mathbf{L}$, e.g the sparsification of inductance matrix for bus4 that has 7 times more elements than the inductance matrix in bus3 is only 2.55 times longer.

Table 3.2: Computational impact of the approximation of dense inductance matrix $\mathbf{L}$ with $\mathcal{H}$-matrix $\mathbf{L}_{\mathcal{H}}$.

Bench.	**Compr. Time to $\mathcal{H}$ matrix**	**Storage**			**Mat-vec Product Time**		
		Dense L	**$\mathcal{H}$ matrix**	**Mem. save**	**Dense L**	**$\mathcal{H}$ matrix**	**Speed up**
bus1	20ms	512.17kB	144.26kB	71.8%	0.04ms	0.03ms	1.33×
bus2	30ms	8MB	746.65kB	90.8%	0.65ms	0.6ms	10.8×
bus3	180ms	128MB	4.89MB	96.1%	11ms	0.32ms	34.4×
bus4	470ms	512MB	15.96MB	96.8%	50ms	1.7ms	29.5×
bus5	520ms	2GB	50.64MB	97.5%	225ms	8ms	28.1×

3.4.3 Preconditioner efficiency analysis

The efficiency of the proposed preconditioner depends on the timestep h_k used in the simulation. Table 3.3 reports the iteration count to solve (3.2) with the proposed preconditioner, which uses two different approximations of the Schur complement ($\mathbf{S} = \mathbf{S}_1$ and $\mathbf{S} = \mathbf{S}_2$), and with no preconditioner for the range of time step sizes that is of practical interest. Also, Table 3.4 reports the spectral norm of $\mathbf{S}_1$ and $\mathbf{S}_2$ used to approximate $\mathbf{S}$ in (3.4) for a given timestep, as in Algorithm 7. We adopt $h_k \in \{$10fs, 1ps, 100ps$\}$, where the lower range is used to compute detailed waveforms and the upper range is used to either examine the general trend in waveforms or to compute steady state. From the iteration count we can observe that the for small steps sizes $h_k \sim 10$ fs the choice $\mathbf{S} = \mathbf{S}_1$ is superior, and for $h_k \sim$ [1,100]ps the choice $\mathbf{S} = \mathbf{S}_2$

leads to smaller iteration count. This is in agreement with the discussion in Subsection 3.3.3 on different weightings of the two terms $\mathbf{S}_1$ and $\mathbf{S}_2$ to the Schur complement of $\mathbf{S}$. The cross-over point between the two preconditioners occurs approximately in the interval [10fs 1ps]. In the case with no preconditioning, the Krylov-iterative method failed to converge for larger step sizes.

Table 3.3: Iteration count to solve the transient system (3.2) with the proposed preconditioner and without preconditioner for timesteps 10fs, 1ps, 100ps at one time instant.

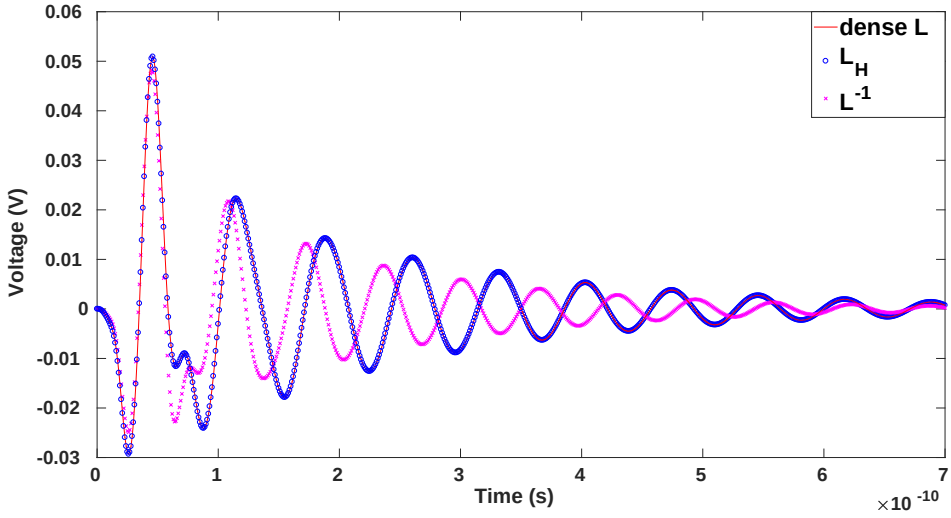
Bench.	**# iter with prec. $\mathbf{U_J}$**			**# iter w/o prec.**		
	h_k = **10fs**	h_k = **1ps**	h_k = **100ps**	h_k = **10fs**	h_k = **1ps**	h_k = **100ps**
bus1	2	21	3	2021	2043	NA
bus2	2	16	3	1848	3961	NA
bus3	7	13	4	1326	4476	NA
bus4	7	18	4	2427	5563	NA
bus5	7	17	4	1876	6158	NA

Table 3.4: The spectral norms of the Schur complement terms for timesteps 10fs, 1ps, 100ps.

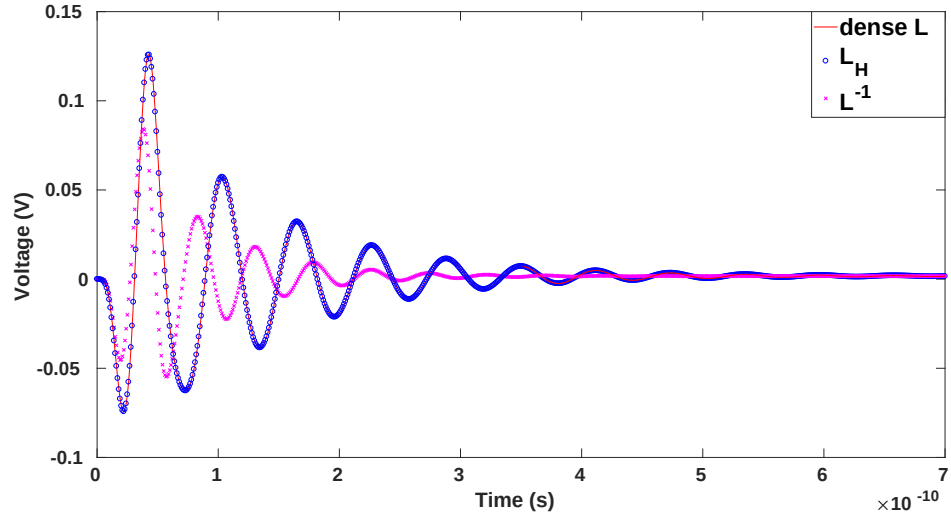
Bench.	h_k=**10fs**		h_k=**1ps**		h_k=**100ps**	
	$\|\|\mathbf{S}_1\|\|_{sp}$	$\|\|\mathbf{S}_2\|\|_{sp}$	$\|\|\mathbf{S}_1\|\|_{sp}$	$\|\|\mathbf{S}_2\|\|_{sp}$	$\|\|\mathbf{S}_1\|\|_{sp}$	$\|\|\mathbf{S}_2\|\|_{sp}$
bus1	**3.7e+5**	20	**3.7e+3**	1.6e+3	37.4	**1.6e+5**
bus2	**1e+5**	64.6	1e+3	**6.3e+3**	10.4	**6.3e+5**
bus3	**7.1e+3**	494.8	71.5	**4.9e+4**	0.7	**4.9e+6**
bus4	**1.3e+4**	505.8	129.9	**5e+4**	1.3	**5e+6**
bus5	**2.5e+4**	509	250	**5.1e+4**	2.5	**5.1e+6**

3.4.4 Transient analysis results

Combining the benefits of the storage of the dense inductance matrix as an $\mathcal{H}$-matrix and the reduced number of Krylov iterations through the proposed preconditioner, Table 3.5 reports the speedups of the simulation of RLC benchmarks for h_k=10ps and 30 time points, in comparison to standard SPICE simulation (dense $\mathbf{L}$) and simulation with $\mathcal{H}$-matrix $\mathbf{L}$ but without preconditioner. For this simulation scenario, the proposed methodology leads to speedups up to 2139× for the largest circuit. Regarding



(a)



(b)

Figure 3-5: Voltage response at randomly chosen nodes of bus1 (a) and bus3 (b) with h_k =1ps (the choice of h_k = 1ps shows all the details in the response waveform) obtained by full SPICE, $\mathcal{H}$-matrix approximation with preconditioning, and sparse reluctance simulation with same memory as $\mathcal{H}$-matrix. The proposed approach is indistinguishable from SPICE, while reluctance-based simulation exhibits significant deviation.

Table 3.5: Runtime results of the whole simulation of benchmark RLC circuits with timestep 1ps.

Bench.	**Size of Matrix $\mathbf{J}_k$**	**SPICE Time**	**Simulation with $\mathcal{H}$-matrix w/o preconditioner**		**Simulation with $\mathcal{H}$-matrix with preconditioner**	
			Time	**Speedup**	**Time**	**Speedup**
bus1	800	1.1s	1s	3.75×	0.28s	3.9×
bus2	3104	61s	9.75s	6.21×	1.57s	38.8×
bus3	12304	4989s	110s	19.19×	5.72s	872×
bus4	24608	31917s	515s	22.4×	23s	1388.3×
bus5	49216	194694s	2545s	76.5×	91s	2139×

the accuracy of the $\mathcal{H}$-matrix approximation, the relative rms error of the simulation compared to exact SPICE was less than 0.01 in all nodes of every benchmark. Simulation waveforms for random nodes of two benchmark circuits are graphically displayed in Fig. 3-5, where it can be observed that the waveforms for $\mathcal{H}$-matrix with the proposed preconditioner are indistinguishable from exact SPICE. Superimposed in the same figures are waveforms from sparse reluctance-based simulation, obtained by inversion and truncation of $\mathbf{L}$ (with sparsity ratio leading to same memory footprint as the H-matrix approximation), which can be observed to exhibit a clear deviation from both exact SPICE and the proposed methodology. Note that for the largest benchmark circuit, the inversion of the dense $\mathbf{L}$ (with size 16.3K) took an additional 111s.

Chapter 4

Conclusion and Future Work

4.1 Conclusion

In this dissertation we present effective techniques for the sparsification and handling dense matrices in circuit simulation. We firstly present a rigorous mathematical approach for the sparsification of the dense matrices resulting in MOR. In the proposed methodology, we derive a second-order formulation of the original model, so that the resulting MOR matrices are close to DD matrices, and then exploit a rigorous sparsification methodology, which entails the computation of the nearest Laplacian matrices of the reduced model matrices under the Frobenius norm and a graph sparsification algorithm, to sparsify these matrices. Moreover, we propose the hierarchical matrices as a suitable compression mechanism for the dense inductance matrix, that arises after modelling all mutual inductive couplings between the interconnects in SoCs, and we present an efficient preconditioner for the system to be solved during the time integration. The contributions of our work in this dissertation can be summarized in the following two results:

- The sparsification of the dense MOR matrices leads to faster simulation times with negligible degradation in accuracy, that is introduced due to the approximations of the ROM matrices with their nearest circuit-type matrices. Actually, we end up with an even stronger result in the sense that the sparsified ROMs

can be synthesized straightforwardly with only positive RLC elements.

- By compressing the fully dense inductance matrix with hierarchical matrices and applying the proposed preconditioner we can tackle much larger problems than previously possible on a given computing platform, both in terms of storage and wall clock time. Our experimental results indicate that a very good compression ratio of inductance matrix can be attained without compromising accuracy, while the proposed preconditioner reduces the iterations count of Krylov iterative method in simulation effectively leading to a significant speedup of the simulation.

4.2 Future Directions

Our future plans on the research presented in this dissertation is to extend it towards the following two directions:

- The approximation of the ROM matrices with the nearest SDD matrices introduces error in simulation. In order to avoid that approximation we suggest to re-think the MOR problem, instead of as a projection problem, but as an optimization problem. The objective of the optimization would be the minimization of the infinite norm (H_∞) of the difference between initial model's transfer function and the corresponding reduced model's transfer function under the constraints that the ROM matrices belong in the class of SDD matrices.

- The proposed preconditioner in Chapter 4 for the simulation of general RLC circuits deviates from the optimal preconditioner, which is the Schur complement of the system to be solved in transient analysis. Recall that we made the appropriate simplifications, on top of the Schur complement preconditioner, in order to solve the preconditioner solve step efficiently. Therefore, a better approach would be to build an application-specific preconditioner that exploits the special structure of the individual problem, such as the structure of a Power Grid, and arrive in a more efficient solution of the preconditioner solve step.

Publications

The results of our research in this dissertation led to the following conference and journal publications:

Journal Publications

- Charalampos Antoniadis, Nestor Evmorfopoulos, Georgios Stamoulis. "On the Sparsification of Dense Matrices in MOR and their Realization in RC/RLC Circuits".IEEE Transactions on COMPUTER-AIDED DESIGN of Integrated Circuits and Systems (TCAD). (Under review)

Peer-Reviewed Conference Publications

- Charalampos Antoniadis, Nestor Evmorfopoulos, Georgios Stamoulis. "On the Sparsification of the Reluctance Matrix in RLCk Circuits Simulation". Conference of Synthesis, Modeling, Analysis and Simulation methods and Applications to Circuits Design (SMACD), July 2-5 2018, Prague, Czech Republic

- Charalampos Antoniadis, Nestor Evmorfopoulos, Georgios Stamoulis. "Efficient Sparsification of Dense Matrices in Model Order Reduction". Asia and South Pacific Design Automation Conference (ASP-DAC), January 21-24 2019, Tokyo, Japan

- Charalampos Antoniadis, Nestor Evmorfopoulos, Georgios Stamoulis. "A Rigorous Approach for the Sparsification of Dense Matrices in Model Order Reduction of RLC Circuits". Design Automation Conference (DAC), June 2-6 2019, Las Vegas, USA

- Dimitrios Garyfallou, Charalampos Antoniadis, Nestor Evmorfopoulos, Georgios Stamoulis. "A Sparsity-Aware MOR Methodology for Fast and Accurate Timing Analysis of VLSI Interconnects". Conference of Synthesis, Modeling, Analysis and Simulation methods and Applications to Circuits Design (SMACD), July 15-18 2019, Laussane, Switzerland

- Charalampos Antoniadis, Milan D. Mihajlovic, Nestor Evmorfopoulos, Georgios Stamoulis and Vasilis F. Pavlidis. "Efficient Linear System Solution Techniques in the Simulation of Large Dense Mutually Inductive Circuits" (short paper). International Conference on Computer Design (ICCD), November 17-20, 2019, Abu Dhabi, United Arab Emirates

Also, our research led to the following publications that is not related with the content of this dissertation

- Charalampos Antoniadis, Georgios Karakonstantis, Nestor Evmorfopoulos, Andreas Burg, George Stamoulis. "On the statistical memory architecture exploration and optimization". Design, Automation & Test in Europe (DATE), March 9-13 2015, Grenoble, France

- Dimos Ntioudis, Christos Kalonakis, Panagiotis Giannakou, Charalampos Antoniadis, Georgios Stamoulis, Panagiota Tsompanopoulou, Nestor Evmorfopoulos, John Moondanos, Georgios Dimiriu. "CCSOpt: a continuous gate-level resizing tool", International Conference on Modern Circuits And Systems Technologies (MOCAST), May 14-15 2015, Thessaloniki, Greece

- Panagiotis Giannakou, Charalampos Antoniadis, Christos Kalonakis, Dimos Dioudis, Georgios Stamoulis, Panagiota Tsompanopoulou, Nestor Evmorfopoulos, John Moondanos, Georgios Dimitriu. "GDS2trim: Physical Layout Manipulation Utility for continuous transistor sizing", International Conference on Modern Circuits And Systems Technologies (MOCAST), May 14-15 2015, Thessaloniki, Greece

- Charalampos Antoniadis, Dimitrios Garyfallou, Nestor Evmorfopoulos and Georgios Stamoulis. "EVT-based Worst Case Delay Estimation under Process Variation". Design, Automation & Test in Europe (DATE), March 19-23 2018, Dresden, Germany

Appendix A

Model Order Reduction Algorithms

In this section we provide "near-to-implementation" (MATLAB-like) pseudocodes of SAPOR and PACT model order reduction algorithms that we use in chapter 2. Beginning with SAPOR (see Algorithm 8), its list of input parameters contains the initial model matrices, as they are shown in (2.5), the frequency (s_0) around which we are interested in approximating the initial model with a ROM, the size of the initial model (n), the desired size of the ROM (r) and the number of input ports (p).

Algorithm 8 SAPOR

function $\widehat{\mathbf{C}}_n$, $\widehat{\mathbf{G}}_n$, $\widehat{\mathbf{\Gamma}}$, $\widehat{\mathbf{B}}_u$, $\widehat{\mathbf{E}}_u$ = SAPOR($\mathbf{C}_n$, $\mathbf{G}_n$, $\mathbf{\Gamma}$, $\mathbf{B}_u$, $\mathbf{E}_u$, s_0, n, r, p)
  $\mathbf{F} = 2 * s_0 * \mathbf{C}_n + \mathbf{G}_n$
  $\mathbf{K} = s_0^2 * \mathbf{C}_n + s_0 * \mathbf{G}_n + \mathbf{\Gamma}$
  $\mathbf{B}_0 = s_0 * \mathbf{B}_1$
  $\mathbf{B}_1 = \mathbf{B}$
  $\mathbf{A} = \begin{bmatrix} -\mathbf{K}^{-1} * \mathbf{F} & \mathbf{K}^{-1} \\ -\mathbf{C}_n & \mathbf{0} \end{bmatrix}$
  $\mathbf{Q}_0 = \mathbf{K}^{-1} * \mathbf{B}_0$
  $\mathbf{P}_0 = \mathbf{B}_1$
  $\mathbf{Q}$ = blkSOAR($\mathbf{A}$, $\mathbf{Q}_0$, $\mathbf{P}_0$, n, r, p)
  $\widehat{\mathbf{C}}_n = \mathbf{Q}^T\mathbf{C}_n\mathbf{Q}$, $\widehat{\mathbf{G}}_n = \mathbf{Q}^T\mathbf{G}_n\mathbf{Q}$, $\widehat{\mathbf{\Gamma}} = \mathbf{Q}^T\mathbf{\Gamma}\mathbf{Q}$
  $\widehat{\mathbf{B}}_u = \mathbf{Q}^T\mathbf{B}_u$, $\widehat{\mathbf{E}}_u = \mathbf{Q}^T\mathbf{E}_u$
end function

On the other hand, the list of input parameters of PACT (see Algorithm 9) consist of the initial model matrices (but now because PACT is a MOR algorithm specifically for RC circuits, the matrix $\mathbf{\Gamma}$ is absent), the number of ports (N), the frequency

band of operation (ω_c) and an error control parameter (ϵ_c), that both determine the extra columns and rows, except for the first N, that have to be added to the ROM after the transformation (2.9). Recall that before applying PACT the equations of Kirchhoff's Current Law (KCL) for every node have to be rearranged so that the first N correspond to port nodes while the rest of them to internal nodes.

Algorithm 9 PACT

function $\widehat{\mathbf{C}}_n$, $\widehat{\mathbf{G}}_n$, $\widehat{\mathbf{B}}_u$, $\widehat{\mathbf{E}}_u$ = PACT($\mathbf{C}_n$, $\mathbf{G}_n$, $\mathbf{B}_u$, $\mathbf{E}_u$, N, ω_c, ϵ_c)
  $n = \mathbf{B}_u$.cols
  $n_i = n - N$
  $\mathbf{G}_i = \mathbf{G}_n(N+1 : end, N+1 : end)$
  $\mathbf{G}_c = \mathbf{G}_n(N+1 : end, 1 : N)$
  $\mathbf{L} = \text{chol}(\mathbf{G}_i)$
  $\mathbf{X} = \begin{bmatrix} \mathbf{I}^{N\times N} & \mathbf{0}^{N\times n_i} \\ -\mathbf{G}_i^{-1}\mathbf{G}_c & \mathbf{L}^{-T} \end{bmatrix}$
  $\widehat{\mathbf{C}}_n = \mathbf{X}^T\mathbf{C}_n\mathbf{X}, \quad \widehat{\mathbf{G}}_n = \mathbf{X}^T\mathbf{G}_n\mathbf{X}, \quad \widehat{\mathbf{B}}_u = \mathbf{X}^T\mathbf{B}_u, \quad \widehat{\mathbf{E}}_u = \mathbf{X}^T\mathbf{E}_u$
  $\widehat{\mathbf{C}}_i = \widehat{\mathbf{C}}_n(N+1 : end, N+1 : end)$
  $[\mathbf{U}, \mathbf{S}] = \text{eig}(\widehat{\mathbf{C}}_i)$
  Sort the diagonal of $\mathbf{S}$ in descending order and perfrom the column swaps in $\mathbf{U}$ so as to preserve the initial eigenvalue/eigenvector correspondence
  Solve the equation $\omega_c * \lambda_c + (\omega_c * \lambda_c)^3 - \epsilon_c = 0$ for λ_c
  $k = 0$
  for $s_i = 1$ `to` n_i **do**
    if $\mathbf{S}(s_i, s_i) > \lambda_c$ **then**
      $k = k + 1$
    else
      break
    end if
  end for
  $\widehat{\mathbf{G}}_n = \widehat{\mathbf{G}}_n(1 : N+k, 1 : N+k)$
  $\widehat{\mathbf{C}}_n = \widehat{\mathbf{C}}_n(1 : N+k, 1 : N+k)$
  $\widehat{\mathbf{B}}_u = \widehat{\mathbf{B}}_u(1 : N+k, :)$
  $\widehat{\mathbf{E}}_u = \widehat{\mathbf{E}}_u(1 : N+k, :)$
end function

Algorithm 10 Block Arnoldi for Second Order Systems

function $\mathbf{Q}$ = BLKSOAR($\mathbf{A}$, $\mathbf{Q}_0$, $\mathbf{P}_0$, n, p, r)
  $k = \frac{r}{p}$
  $\mathbf{Q} = \mathbf{P} = \mathbf{0}^{n \times r}$
  $[\mathbf{Q}_1, \mathbf{P}_1]$ = SOrth($\mathbf{Q}_0$, $\mathbf{P}_0$, p)
  $\mathbf{Q}(:, 1:p) = \mathbf{Q}_1, \quad \mathbf{P}(:, 1:p) = \mathbf{P}_1$
  $\mathbf{A}_1 = \mathbf{A}(1:n, 1:n), \quad \mathbf{A}_2 = \mathbf{A}(1:n, n+1:end), \quad \mathbf{A}_3 = \mathbf{A}(n+1:end, 1:n)$
  for $i = 1$ `to` $k-1$ **do**
    $\mathbf{Qi} = \mathbf{Q}(:, (i\text{-}1)*p+1{:}i*p), \quad \mathbf{Pi} = \mathbf{P}(:, (i\text{-}1)*p+1{:}i*p)$
    $\mathbf{Q}_i = \mathbf{A}_1 * \mathbf{Qi} + \mathbf{A}_2 * \mathbf{Pi}, \quad \mathbf{P}_i = \mathbf{A}_3 * \mathbf{Qi}$
    for $j = 1$ `to` i **do**
      $\mathbf{Qj} = \mathbf{Q}(:, (j\text{-}1)*p+1{:}j*p), \quad \mathbf{Pj} = \mathbf{P}(:, (j\text{-}1)*p+1{:}j*p)$
      $\mathbf{H}_{ji} = \mathbf{Q}_j^T * \mathbf{Q}_i$
      $\mathbf{Q}_i = \mathbf{Q}_i$ - $\mathbf{Qj} * \mathbf{H}_{ji}, \quad \mathbf{P}_i = \mathbf{P}_i$ - $\mathbf{Pj} * \mathbf{H}_{ji}$
    end for
    $[\mathbf{Q}(:, i*p+1:(i+1)*p), \mathbf{P}(:, i*p+1:(i+1)*p)]$ = SOrth($\mathbf{Q}_i, \mathbf{P}_i, p$);
  end for
end function

Algorithm 11 Matrix Orthonormalization

function $\mathbf{Q}, \mathbf{P}$ = SORTH($\mathbf{Q}_m$, $\mathbf{P}_m$, p)
  $\mathbf{Q} = \mathbf{0}^{size(\mathbf{Q}_m)}, \quad \mathbf{P} = \mathbf{0}^{size(\mathbf{P}_m)}$
  for $i = 1$ `to` p **do**
    $\mathbf{q}_i = \mathbf{Q}_m(:, i), \quad \mathbf{p}_i = \mathbf{P}_m(:, i)$
    for $j = 1$ `to` $i-1$ **do**
      $\mathbf{R}_{ji} = \mathbf{Q}(:, j)^T * \mathbf{q}_i$
      $\mathbf{q}_i = \mathbf{q}_i - \mathbf{R}_{ji} * \mathbf{Q}(:, j), \quad \mathbf{p}_i = \mathbf{p}_i - \mathbf{R}_{ji} * \mathbf{P}(:, j)$
    end for
    $R_{ii} = ||\mathbf{q}_i||_2$
    if $R_{ii} == 0$ **then**
      print("deflation")
      break
    else
      $\mathbf{q}_i = \frac{1}{R_{ii}} * \mathbf{q}_i, \quad \mathbf{p}_i = \frac{1}{R_{ii}} * \mathbf{p}_i$
    end if
    $\mathbf{Q}(:, i) = \mathbf{q}_i, \quad \mathbf{P}(:, i) = \mathbf{p}_i$
  end for
end function

Appendix B

Compression and Routines with $\mathcal{H}$-matrices

In this section, we offer a pseudocode (see Algorithm 12) of how to use HLIBpro library, which is rather straightforward, in order to compress a dense matrix. Moreover, we provide an algorithm (see Algorithm 13) for the multiplication of the system matrix in (3.2), that consists of different matrix formats, with $\mathbf{C}_n$, $\mathbf{G}_n$ and $\mathbf{A}_L$ being in sparse matrix format and $\mathbf{L}$ being in $\mathcal{H}$-matrix format, with a vector.

Algorithm 12 Compression of a dense matrix with HLIBpro

1: **function** $\mathbf{A}_{\mathcal{H}}$ = HLIBpro($\mathbf{A}$)
2: Associate every i, j entry of dense matrix $\mathbf{A}$ with a (x,y,z) coordinate in Euclidean space.
3: Provide the above relation between matrix entries and coordinates, the minimum size of matrix blocks to be approximated by low-rank products, the accuracy of approximation of matrix blocks by low-rank products and a distance criterion between the coordinates, to HLIBpro library.
4: **end function**

Algorithm 13 Matrix multiplication of system matrix in (3.2) with a vector

1: **function** $\mathbf{q}$ = beMatrixMul($\mathbf{J}_k$, $\mathbf{p}$)
2: $\mathbf{p}_1 = \mathbf{p}(1:n), \quad \mathbf{p}_2 = \mathbf{p}(n+1:end)$
3: $\mathbf{q}_1 = (\frac{1}{h_k}\mathbf{C}_n + \mathbf{G}_n) * \mathbf{p}_1 + \mathbf{A}_L * \mathbf{p}_2, \quad \mathbf{q}_2 = -\mathbf{A}_L^T * \mathbf{p}_1 + \frac{1}{h_k}\mathbf{L} * \mathbf{p}_2$
4: $\mathbf{q} = \begin{bmatrix}\mathbf{q}_1 & \mathbf{q}_2\end{bmatrix}^T$
5: **end function**

Bibliography

[1] M. Abadir, *https://semiengineering.com/is-it-time-to-take-inductance-and-electromagnetic-effects-on-socs-seriously/*, Semiconductor Engineering June, 2018.

[2] Z. He, M. Celik, and L. T. Pileggi, *SPIE: Sparse Partial Inductance Extraction*, Design Automation Conference (DAC), 1997.

[3] A. Devgan, H. Ji, and W. Dai, *How to Efficiently Capture On-Chip Inductance Effects: Introducing a New Circuit Element K*, IEEE/ACM International Conference on Computer-Aided Design, 2000.

[4] Y. Tanji, T. Watanabe and H. Asai, *Generating stable and sparse reluctance/inductance matrix under insufficient conditions*, Asia and South Pacific Design Automation Conference (ASP-DAC), 2008.

[5] If. Apostolopoulou, K. Daloukas, N. Evmorfopoulos and G. Stamoulis *Selective inversion of inductance matrix for large-scale sparse RLC simulation*, Design Automation Conference (DAC), 2014.

[6] G. Zhong, C. Koh, and K. Roy, *On-Chip Interconnect Modeling by Wire Duplication*, IEEE/ACM International Conference on Computer Aided Design, 2002, pp. 341-346.

[7] T. Chen, C. Luk, and C. Chen, *Inductwise: Inductance-Wise Interconnect Simulator and Extractor*, IEEE Trans. Computer-Aided Design, vol. 22, no. 7, pp. 884-894, 2003.

[8] M. Beattie and L. Pileggi, *Modeling Magnetic Coupling for On-Chip interconnect*, IEEE/ACM Design Automation Conf.(DAC), 2001.

[9] C. Antoniadis, N. Evmorfopoulos and G. Stamoulis *On the Sparsification of the Reluctance Matrix in RLCk Circuit Transient Analysis*, International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design, 2018 (SMACD), 2018.

[10] L.T. Pillage and R.A. Rohrer *Asymptotic waveform evaluation for timing analysis*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 9, no. 4, pp. 352-366, April 1990

[11] P. Feldmann and R.W. Freund, *Efficient linear circuit analysis by Pade approximation via the Lanczos process*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 14, pp. 639-649, 1995.

[12] R.W. Freund and P. Feldmann, *Reduced-Order Modeling of Large Linear Subcircuits via a Block Lanczos Algorithm*, Design Automation Conference, 1995

[13] R.W. Freund P. Feldmann, *Reduced-order modeling of large passive linear circuits by means of the SyPVL algorithm*, International Conference on Computer-aided Design, 1996

[14] A. Odabasioglu, M. Celik and L.T. Pileggi, *PRIMA: passive reduced-order interconnect macromodeling algorithm*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 17, pp. 645-654, 1998.

[15] J. R. Phillips, L. Daniel and L. M. Silveira, *Guaranteed passive balancing transformations for model order reduction*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 22, no. 8, pp. 1027-1041, Aug. 2003.

[16] J. Phillips and L. M. Silveira, *Poor man's TBR: a simple model reduction scheme*, Design, Automation and Test in Europe Conference and Exhibition, 2004

[17] Z. Ye, D. Vasilyev, Z. Zhu and J. R. Phillips, *Sparse Implicit Projection (SIP) for reduction of general many-terminal networks*, IEEE/ACM International Conference on Computer-Aided Design, 2008.

[18] P. Miettinen, M. Honkala, J. Roos, M. Valtonen, *PartMOR: Partitioning-Based Realizable Model-Order Reduction Method for RLC Circuits*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 30, pp. 374-387, 2011.

[19] H. Yu, C. Chu, Y. Shi, D. Smart L. He and S. X.-D. Tan *Fast Analysis of Large Scale Inductive Interconnect by Block Structure Preserved Macromodeling*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 18, pp. 1399-1411, 2009.

[20] F. N. Najm, *Circuit Simulation*, Wiley-IEEE Press, 2010.

[21] A. E. Ruehli, *Circuit Analysis, Simulation and Design, Part 1*, New York: North-Holland, 1986.

[22] C. Antoniadis, N. Evmorfopoulos and G. Stamoulis, *Efficient Sparsification of Dense Circuit Matrices in Model Order Reduction*, Asia and South Pacific Design Automation Conference, 2019.

[23] R. Horn and C. Johnson, *Matrix Analysis*, Cambridge University Press, 1985.

[24] R. Merris, *Laplacian matrices of a graph: A survey*, Linear Algebra and its Applications, vol. 197, pp. 143-176, 1994.

[25] Freud R. W., *SPRIM: structure-preserving reduced-order interconnect macromodeling.*, IEEE/ACM International Conference on Computer-Aided Design, 2004.

[26] B.N. Sheehan, *ENOR: Model order reduction of RLC circuits using nodal equations for efficient factorization*, IEEE/ACM Design Automation Conference, 1999.

[27] H. Zheng, L. T. Pileggi, *Robust and passive model order reduction for circuits containing susceptance elements*, IEEE/ACM International Conference on Computer Aided Design, 2002.

[28] Y. Shi, H. Yu, and L. He, *SAMSON: A generalized second-order arnoldi method for reducing multiple source linear network with susceptance*, ACM International Symposium on Physical Design, 2006

[29] Y. Su, J. Wang, X. Zeng, Z. Bai, C. Chiang, D. Zhou *SAPOR: second-order Arnoldi method for passive order reduction of RCS circuits*, IEEE/ACM International Conference on Computer Aided Design, 2004.

[30] J.P. Boyle and R.L. Dykstra, *A Method for Finding Projections onto the Intersection of Convex Sets in Hilbert Spaces*, Springer, 1986.

[31] M. Mendoza, M. Raydan and P. Tarazaga, *Computing the nearest diagonally dominant matrix*, Numerical Linear Algebra with Applications, vol. 5, pp. 461-474, 1998.

[32] M. Monsalve, J. Moreno, R. Escalante and M. Raydan, *Selective alternating projections to find the nearest SDD+ matrix*, Applied Mathematics and Computation, vol. 145, pp. 205-220, 2003.

[33] K. J. Kerns and A. T. Yang, *Stable and efficient reduction of large, multiport rc networks by pole analysis via congruence transformations*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1997.

[34] D. Oyaro, P. Triverio, *TurboMOR: an Efficient Model Order Reduction Technique for RC Networks with Many Ports*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2015.

[35] D. Spielman and N. Srivastava, *Graph Sparsification by Effective Resistances*, SIAM Journal on Computing, vol. 40, pp. 1913-1926, 2011.

[36] D. Achlioptas, *Database-friendly Random Projections*, ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, 2001.

[37] D. Spielman and S.H. Teng, *Nearly-linear Time Algorithms for Graph Partitioning, Graph Sparsification, and Solving Linear Systems*, ACM Symposium on Theory of Computing, 2004.

[38] I. Koutis, G.L. Miller and R. Peng, *A nearly-m logn solver for SDD linear systems*, IEEE Symposium on Foundations of Computer Science, 2011.

[39] F. Yang, X. Zeng, Y. Su, D. Zhou, *RLCSYN: RLC Equivalent Circuit Synthesis for Structure-Preserved Reduced-order Model of Interconnect*, IEEE International Symposium on Circuits and Systems, 2007.

[40] R. Ionutiu, J. Rommes and W. H. A. Schilders, *SparseRC: Sparsity Preserving Model Reduction for RC Circuits With Many Terminals*, IEEE Transactions on Integrated Circuits and Systems, vol. 30, pp. 1828-1841, Dec. 2011.

[41] R. Barrett et al., *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, Society for Industrial and Applied Mathematics, 1994.

[42] Saad Y., *Iterative Methods for Sparse Linear Systems*, Society for Industrial and Applied Mathematics, 2003.

[43] C. Eckart. and G. Young, *The approximation of one matrix by another of lower rank*, Psychometrika, vol. 1, pp. 211-218, 1936.

[44] W. Hackbush, *Hierarchical Matrices: Algorithms and Analysis*, Springer Publishing Company, 2015.

[45] G. W. Stewart, *Matrix Algorithms: Volume II: Eigensystems*, Society for Industrial and Applied Mathematics, 2001.

[46] M. Kamon, M. J. Tsuk and J. K. White, *FASTHENRY: a multipole-accelerated 3-D inductance extraction program*, IEEE Transactions on Microwave Theory and Techniques, vol. 42, pp. 1750-1758, Sept. 1994.

[47] S. Borm, L. Grasedyck and W. Hackbusch, *Introduction to Hierarchical Matrices with Applications*, Engineering Analysis with Boundary Elements, vol. 27, pp. 405-422, 2003.

[48] R. Kriemann, *Parallel H-Matrix Arithmetics on Shared Memory Systems*, Computing vol. 74, pp. 273-297, 2005.

[49] R. Kriemann, L. Grasedyck and S. L. Borne, *Parallel blackbox H-LU preconditioning for elliptic boundary value problems*, Computing and Visualization in Sciences, vol. 11, pp. 273-291, 2007.

[50] Max Planck institute for Mathematics in the Sciences, *https://www.hlibpro.com/doc/2.7/pages.html*, 2019.