A Parallel Hamiltonian Eigensolver for Passivity Characterization and Enforcement of Large Interconnect Macromodels

L. Gobbato, A. Chinea, S. Grivet-Talocia
Dip. Elettronica, Politecnico di Torino
e-mail stefano.grivet@polito.it

Abstract—**The passivity characterization and enforcement of linear interconnect macromodels has received much attention in the recent literature. It is now widely recognized that the Hamiltonian eigensolution is a very reliable technique for such characterization. However, most available algorithms for the determination of the required Hamiltonian eigenvalues still require excessive comoputational resources for large-size macromodels with thousands of states. This work intends to break this complexity by introducing the first parallel implementation of a specialized Hamiltonian eigensolver, designed and optimized for shared memory multicore architectures. Our starting point is a multi-shift restarted and deflated Arnoldi process. Excellent parallel efficiency is obtained by running different Arnoldi iterations concurrently on different threads. The numerical results show that macromodels with several thousands states are characterized in few seconds on a 16-core machine, with close to ideal speedup factors.**

I. Introduction

The analysys and design flow of electrical interconnected systems is more and more relying on reduced-order macromodels. Such macromodels are often identified from tabulated frequency responses, typically available from a full-wave solver or from direct measurement, using rational curve fitting. Many algorithms have been proposed, most providing some generalization or optimization of the core method known as Vector Fitting [1]. See [2]–[5] and references therein for an overview.

The main concern for a reliable use of rational macromodels within standard circuit solvers is passivity. Non-passive macromodels do not guarantee the global stability of transient simulations, due to their ability to amplify the energy they are fed with [6]. Therefore, it is of paramount importance to check and enforce passivity during macromodel generation. This is the reason why this problem has been extensively studied over the last few years [8]– [23].

One of the most attractive techniques for passivity characterization is based on the so-called Hamiltonian matrix [7], [8]. Extraction of the imaginary eigenvalues of this matrix provides an algebraic method for full passivity characterization. Unfortunately, a standard full eigensolution scales as the third power of the problem size. This fact prevents an efficient characterization for large-size macromodels, such as those arising in signal and power integrity analyses for electronic packaging applications. These macromodels may have up to hundred interface ports, and several thousand states.

Although some efforts have been attempted towards more efficient Hamiltonian eigensolutions [9], [17], [20], [21], [24], there is still significant potential for improvement. The main direction that we investigate in this work is parallelization. High-performance computing systems with many CPU cores are becoming widespread, even at the desktop level. It is thus necessary to port existing algorithms to such platforms, so that optimal use of the computational resources is achieved.

We introduce a parallel multi-shift restarted Arnoldi scheme for the identification of all imaginary Hamiltonian eigenvalues. A dedicated thread scheduling strategy has been devised and implemented using the OpenMP paradigm. Several benchmarks were run on a 16-core machine, leading to excellent parallel efficiency and speedup factors with respect to a serial implementation. After reviewing some background material in Sec. II and Sec. III, we present the main algorithm in Sec. IV and the numerical results in Sec. V.

II. Background and Notation

Throughout this paper x, $\boldsymbol{x}$, and $\mathbf{X}$ denote a generic scalar, vector (lowercase and boldface), and matrix (uppercase and boldface), respectively. Superscripts *, ^{T}, and ^{H} will stand for the complex conjugate, transpose, and conjugate (Hermitian) transpose, respectively.

We consider linear macromodels in state-space form

$$\mathbf{H}(s) = \mathbf{D} + \mathbf{C}\,(s\mathbf{I} - \mathbf{A})^{-1}\,\mathbf{B} \tag{1}$$

where s is the Laplace variable, $\mathbf{H}(s)$ is the $p \times p$ transfer matrix of the macromodel, and $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$ are the state-space matrices of some realization associated to $\mathbf{H}(s)$. All poles, or equivalently the eigenvalues of $\mathbf{A}$, are assumed to be strictly stable, so that the transfer matrix $\mathbf{H}(s)$ is nonsingular for $\mathbb{R}\mathrm{e}\,\{s\} \geq 0$.

Several techniques are available for obtaining a macromodel in the form (1). Standard approaches involve some fitting, approximation, or identification process from tabulated responses. A common scenario is the availability of raw data in form of frequency samples of the scattering matrix $\{\mathbf{S}(j\omega_k),\, k = 1, \ldots, K\}$ of a linear component over a given bandwidth, either via electromagnetic simulation or direct measurement. The four state-space matrices are found by fitting the model equations to this data. Among others, the well-known *Vector Fitting* scheme in its various implementations [1]–[5] has gained much polularity thanks to its simplicity and robustness.

In this work, we are interested in large-size macromodels, both in terms of electrical interface ports p and overall dynamic order n, i.e., the size of state-space matrix $\mathbf{A}$. In order to reduce both storage requirements and processing time, we will assume a state-space realization with the same structure as in [9], more precisely

$$\begin{aligned} \mathbf{A} &= \text{blkdiag}\{\mathbf{A}_k,\, k=1,\dots,p\} \\ \mathbf{B} &= \text{blkdiag}\{\boldsymbol{u}_k,\, k=1,\dots,p\} \\ \mathbf{C} &= [\mathbf{C}_1, \mathbf{C}_2, \dots, \mathbf{C}_p] \end{aligned} \quad (2)$$

where $\mathbf{A}_k \in \mathbb{R}^{m_k \times m_k}$ stores in its diagonal the m_k poles of the k-th column of $\mathbf{H}(s)$, $\boldsymbol{u}_k$ is a $m_k \times 1$ array with all entries equal to one, and $\mathbf{C}_k \in \mathbb{R}^{p \times m_k}$ stores the residues of the k-th column of $\mathbf{H}(s)$. If complex pole pairs are present, the transformation in [9] can be applied to the relevant blocks of (2) in order to recover a real realization. Therefore, we will assume a real-valued realization without loss of generality. The above structure corresponds to a multiple SIMO (Single-Input Multiple-Output) configuration. The maximum number nonvanishing elements of $\mathbf{A}$ and $\mathbf{B}$ is $2n$ and n, respectively, thus enabling sparse storage and optimized processing.

In the following, we will restrict our attention to scattering input-output representations, although the same derivations can be performed for the impedance, admittance, and hybrid cases. Due to the asymptotic stability assumption, macromodel passivity holds when all the singular values of the transfer matrix are uniformly bounded by one at any frequency

$$\sigma_i \le 1\,, \quad \forall \sigma_i \in \sigma(\mathbf{H}(j\omega))\,, \quad \forall \omega\,. \quad (3)$$

We will assume strict asymptotic passivity by assuming

$$\sigma_i < 1\,, \quad \forall \sigma_i \in \sigma(\mathbf{D}), \quad (4)$$

a condition that can be easily enforced during the macromodel generation [9].

The Hamiltonian matrix associated to the state-space realization (1) provides an effective tool for deriving the passivity conditions in a purely algebraic form, without resorting to sampling. In the scattering case, the Hamiltonian matrix reads

$$\mathcal{M} = \begin{bmatrix} \mathbf{A} - \mathbf{B}\mathbf{R}^{-1}\mathbf{D}^T\mathbf{C} & -\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T \\ \mathbf{C}^T\mathbf{S}^{-1}\mathbf{C} & -\mathbf{A}^T + \mathbf{C}^T\mathbf{D}\mathbf{R}^{-1}\mathbf{B}^T \end{bmatrix}, \quad (5)$$

with $\mathbf{R} = (\mathbf{D}^T\mathbf{D} - \mathbf{I})$ and $\mathbf{S} = (\mathbf{D}\mathbf{D}^T - \mathbf{I})$. The set Ω of purely imaginary eigenvalues of $\mathcal{M}$ correspond to the frequencies where one or more singular values cross or touch the unit threshold. If Ω is empty, the model is passive thanks to condition (4). If Ω is not empty and at least one imaginary eigenvalue has odd multiplicity (usually all eigenvalues are simple), then the model is not passive. Therefore, knowledge of Ω allows a full qualification of the model passivity [7], [8]. Once this characterization is available, one of the standard perturbation methods for passivity enforcement [8]– [21] can be applied. No further details on passivity enforcement will be provided here, since this subject is well developed in the above literature. We will rather focus on the efficient determination of the set Ω.

III. Single-shift iterations

The set Ω of imaginary Hamiltoniam eigenvalues can be determined by postprocessing the full set of eigenvalues computed by a full eigensolver. This option is however unacceptable for large-size macromodels, due to the excessive computational cost. The Hamitonian matrix has size $2n \times 2n$ and is full, regardless of the special structure of the adopted state-space realization. Consequently, the cost for the full eigensolution scales as $O(n^3)$, which prevents good scaling to dynamic orders of several thousands.

Alternative approaches have been proposed to reduce this cost. One technique [9] is based on a shifted and restarted Arnoldi process, aimed at the extraction of few eigenvalues in a disk centered on the imaginary axis. Iteration of this process via bisection on the desired bandwidth leads to the identification of all imaginary eigenvalues, disregarding the remaining part of the spectrum. A second approach, based on the Laguerre's method [20], [21], allows the computation of the full eigenspectrum in $O(n^2)$ operations. In this work, we focus on the former approach, due to its superior potential for efficient parallelization.

We recall that, given an arbitrary shift $\vartheta \in \mathbb{C}$, the shifted and inverted Hamiltonian matrix can be represented as

$$\begin{aligned} (\mathcal{M} - \vartheta\mathcal{I})^{-1} = &\begin{bmatrix} \mathbf{A}_\vartheta & \\ & -\mathbf{A}^T_{-\vartheta} \end{bmatrix} \\ - &\begin{bmatrix} \mathbf{A}_\vartheta & \\ & -\mathbf{A}^T_{-\vartheta} \end{bmatrix} \begin{bmatrix} \mathbf{B} & \\ & -\mathbf{C}^T \end{bmatrix} \\ \times &\begin{bmatrix} -\mathbf{H}_\vartheta & \mathbf{I} \\ \mathbf{I} & -\mathbf{H}^T_{-\vartheta} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{C} & \\ & \mathbf{B}^T \end{bmatrix} \begin{bmatrix} \mathbf{A}_\vartheta & \\ & -\mathbf{A}^T_{-\vartheta} \end{bmatrix} \end{aligned} \quad (6)$$

by application of the Shermann Morrison Woodbury Lemma [9], [25], where $\mathbf{A}_{\pm\vartheta} = (\mathbf{A} \pm \vartheta\mathbf{I})^{-1}$ and $\mathbf{H}_{\pm\vartheta} = \mathbf{D} - \mathbf{C}\mathbf{A}_{\pm\vartheta}\mathbf{B}$. The number of operations required by the above computations has a leading term which is linear in the number of macromodel states n. This fact enables using Krylov-subspace techniques to find selected eigenvalues. Starting from a random initial vector $\boldsymbol{v}_1$, a d-dimensional orthogonal basis

$$\mathbf{V}_d = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_d] \quad (7)$$

of the Krylov subspace

$$\text{span}\left\{\boldsymbol{v}_1, (\mathcal{H} - \vartheta\mathcal{I})^{-1}\boldsymbol{v}_1, \dots, (\mathcal{H} - \vartheta\mathcal{I})^{-d+1}\boldsymbol{v}_1\right\} \quad (8)$$

is found by constructing one vector at the time and orthonormalizing it with respect to previous vectors. The Galerkin projection of the original Hamiltonian eigenvalue problem onto this subspace provides an approximation of few eigenvalues that are closest to the shift (the convergence rate improves the closer is the eigenvalue to the shift).

The implementation of the above process needs some special tricks in order to avoid missing eigenvalues and converge quickly. In particular, explicit restarts and incremental deflation are adopted in our implementation, according to the guidelines of [9]. A few remarks are in order

- For best efficiency, the dimension of the Krylov subspace d is chosen to be much smaller than the matrix order $2n$. For all examples we used a maximum size $d = 60$.
- For a given shift ϑ, only a small number n_ϑ of eigenvalues are sought for, typically 4–6. In fact, we have the requirement that $n_\vartheta \ll d$ in order to guarantee good eigenvalue stabilization. Moreover, we require a fine granularity in view of the parallelization effort, to be described below. The best implementation is thus achieved with a very fast determination of very few eigenvalues closest to ϑ. Calculation of all desired eigenvalues in Ω will be achieved by repeating the process using many shifts, which will be allocated to different computing threads.
- In addition to the eigenvalue estimates, we will need the guarantee that no other eigenvalues are present within a disk $\mathcal{C}_{\vartheta,\rho} = \{s \in \mathbb{C} : |s - \vartheta| < \rho\}$. Obviously, the radius of this disk depends on the eigenvalue distribution around the shift. Therefore, we start with an initial guess of the radius ρ_0 and we update it during the restarted iterations by monitoring the convergence of all eigenvalues. If more eigenvalues than n_ϑ are converging within the current disk, the radius is reduced so that only n_ϑ eigenvalues are enclosed, and the remaining ones are discarded. If some of the n_ϑ converging eigenvalues are outside the initial disk, the radius is redefined to the maximum distance of a converging eigenvalue from the shift. Therefore, the final radius can be $\rho \gtreqless \rho_0$.

The above observations are summarized by what we call *single-shift* Arnoldi iteration $\mathcal{S}$, which can be formally described in functional form as

$$(\{\lambda_k\}, \rho) \leftarrow \mathcal{S}(\vartheta, \rho_0) \tag{9}$$

where the input parameters are the shift and the initial radius, and the output parameters are the complete set of eigenvalues $\{\lambda_k\}$ that are included in the disk $\mathcal{C}_{\vartheta,\rho}$. Note that this eigenvalue set may be empty if the initial radius is small. A graphical illustration of the single-shift iteration results is outlined in Fig. 1. Additional details and derivations can be found in [9].

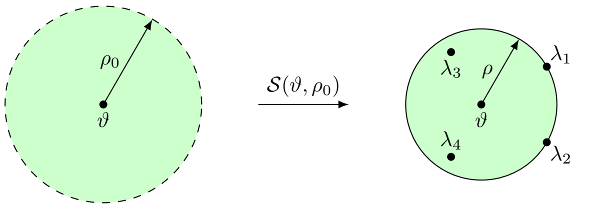

Fig. 1. Schematic representation of the single-shift operator $\mathcal{S}$.

IV. Parallel Hamiltonian Eigensolver

The $\mathcal{S}$-iteration (9) finds few eigenvalues that are localized around a prescribed point ϑ. As described in [9], placing a set of multiple shifts $\{\vartheta_k, k \in \mathcal{K}\}$ on the imaginary axis via a bisection process allows covering the entire bandwidth of interest $[\omega_{\min}, \omega_{\max}]$ with the union of the corresponding disks $\mathcal{C}_{\vartheta_k,\rho_k}$, thus guaranteeing the detection of all imaginary Hamiltonian eigenvalues.

The main issue with a standard bisection process is the strong dependency of each iteration step on the completion of previous steps, since the convergence radius of each shift is not known. We illustrate this difficulty with reference to Fig. 2. The first two shifts ϑ_1, ϑ_2 are placed at the edges of the search band and processed. The resulting radii ρ_1, ρ_2 are generally different. Since the optimal location of the third shift is in the midpoint

$$\vartheta_3 = \frac{(\vartheta_1 + \rho_1) + (\vartheta_2 - \rho_2)}{2}, \tag{10}$$

processing of ϑ_3 requires full completion of the single-shift iterations associated to ϑ_1 and ϑ_2. This data dependency is obviously found at any level of the bisection tree. One could neglect this dependency and predistribute the shifts on a regular grid, to be processed by successive binary subdivision. However, in a multi-threaded implementation where individual shifts are processed by individual threads, it is very likely that the work performed on some preallocated shifts will be useless, since they could be included in the convergence disks associated to nearby disks. Although this strategy is acceptable for a serial implementation, there is no potential for good scalability to a multi-threaded implementation deployed on a machine with many computational cores. This poor scalability was indeed verified experimentally.

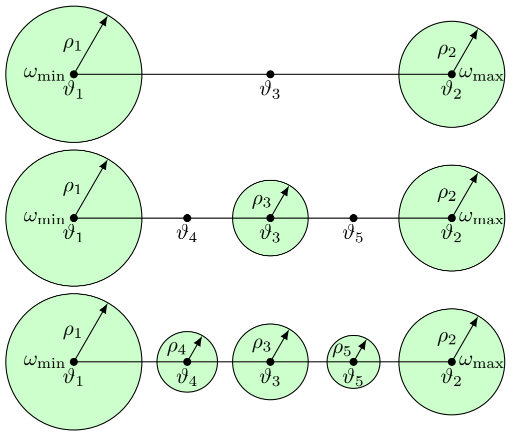

Fig. 2. Bisection process used to cover the bandwidth $[\omega_{\min}, \omega_{\max}]$.

We now introduce the adopted parallelization strategy, which is aimed at

- allocating individual single-shift iterations to individual threads;
- making sure that the work of all concurrent threads is independent on each other;

- making sure that no thread performs a single-shift iteration that is not strictly required.

The above requirements are addressed by a dedicated scheduling strategy, illustrated below.

A. *Data organization and initialization*

We assume that the search area is represented by the frequency interval $[\omega_{\min}, \omega_{\max}]$. Usually, the lower bound corresponds to zero frequency $\omega_{\min} = 0$, whereas the upper bound is precomputed as the magnitude of the largest Hamiltonian eigenvalue, which can be otbained with a single-shift iteration on the Hamiltonian matrix $\mathcal{M}$ without applying any shift-and-invert operation.

The algorithm is parameterized by the number of threads T that will run concurrently. We first subdivide the search area into N adjacent and non-overlapping intervals $\widetilde{I}_\nu = [\widetilde{I}_{L,\nu}, \widetilde{I}_{U,\nu}]$, where

$$\begin{aligned} \widetilde{I}_{L,\nu} &= \omega_{\min} & \nu &= 1\,, \\ \widetilde{I}_{L,\nu} &= I_{U,\nu-1} & 1 < \nu &\le N\,, \\ \widetilde{I}_{U,\nu} &= \omega_{\max} & \nu &= N\,. \end{aligned}$$

The number of these intervals is at least double the number of threads, $N = \kappa T$ with $\kappa \ge 2$. A tentative shift $\widetilde{\vartheta}_\nu$ is then attributed to each interval according to the rule

$$\begin{aligned} \widetilde{\vartheta}_\nu &= \widetilde{I}_{L,\nu} & \nu &= 1\,, \\ \widetilde{\vartheta}_\nu &= (\widetilde{I}_{L,\nu} + \widetilde{I}_{U,\nu})/2 & 1 < \nu &< N\,, \\ \widetilde{\vartheta}_\nu &= \widetilde{I}_{U,\nu} & \nu &= N\,. \end{aligned}$$

These tentative shifts are collected into a set

$$\widetilde{\Theta}_0 = \{\widetilde{\vartheta}_\nu,\, \nu \in \mathcal{I}_0\} \tag{11}$$

where $\mathcal{I}_0$ is an index set that at the initialization step coincides with the set of consecutive integers $\{1, \ldots, N\}$. We also define a set $\Theta_0 = \emptyset$, that will include all shifts that are being processed (denoted as $\widehat{\vartheta}_\mu$) or have been processed (denoted as ϑ_μ) by single-shift iterations. The same notation will be propagated to the intervals surrounding each shift.

B. *Algorithm startup*

In the startup phase, a number of T threads must start processing at the same time. Each thread will then pick one of the tentative shifts from $\widetilde{\Theta}_0$ and will perform the corresponding single-shift iteration. These tentative shifts that are selected for processing are removed from the set $\widetilde{\Theta}_0$ and "promoted" to the set Θ_0. We will use a global index k in order to keep track of individual threads and shifts. After this startup phase, we have $k = T$ shifts in the processing state

$$\Theta_k = \{\widehat{\vartheta}_1, \ldots, \widehat{\vartheta}_k\} \tag{12}$$

with

$$\widehat{\vartheta}_1 = \widetilde{\vartheta}_1\,, \tag{13}$$

$$\widehat{\vartheta}_2 = \widetilde{\vartheta}_N\,, \tag{14}$$

$$\widehat{\vartheta}_\nu = \widetilde{\vartheta}_{\nu-1}, \quad 3 \le \nu \le T \tag{15}$$

so that the extrema of the search bandwidth are processed first, followed by as many shifts as allowed by the available threads, see Fig. 3 for an illustration. The set of tentative shifts is also deflated as

$$\widetilde{\Theta}_k = \widetilde{\Theta}_0 \setminus \{\widetilde{\vartheta}_1, \widetilde{\vartheta}_N, \widetilde{\vartheta}_2, \ldots \widetilde{\vartheta}_{T-1}\}\,. \tag{16}$$

At successive iterations the set of tentative shifts will be best described by a compact notation

$$\widetilde{\Theta}_k = \{\widetilde{\vartheta}_\nu \mid \nu \in \mathcal{I}_k\}\,, \tag{17}$$

where $\mathcal{I}_k$ is an index set (possibly with non-consecutive elements) that depends on the iteration count, and whose number of elements will be denoted as M_k,

$$\mathcal{I}_k = \{i_1^{(k)}\,,\ \ldots\ ,\ i_{M_k}^{(k)}\}\,. \tag{18}$$

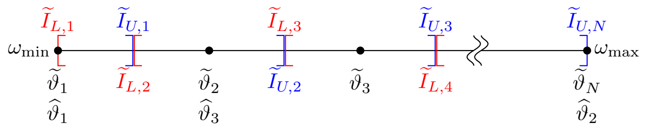

Fig. 3. Shifts in the processing state after the startup phase for the case $T = 3$.

C. *Shift selection and processing*

The above process can be applied anytime a new thread is idle and needs to select and process a shift. Assuming a global iteration count $k \ge T$, a new shift is added to the processing list as

$$\Theta_{k+1} = \Theta_k \cup \{\widehat{\vartheta}_{k+1}\}\,, \tag{19}$$

where

$$\widehat{\vartheta}_{k+1} = \widetilde{\vartheta}_\mu\,, \quad \mu \in \mathcal{I}_k\,, \quad \nexists \widehat{\vartheta}_\nu, \widetilde{\vartheta}_\nu \in \left[\widetilde{I}_{\mu,L}, \widetilde{I}_{\mu,U}\right]\,. \tag{20}$$

In other words, an equivalence is established between the index μ in the tentative queue and the index $k+1$ in the processing queue, see Fig. 4. Conditions (20) ensure that a "free" interval will be processed, i.e., not including any shift that is being processed or has the potential to be processed by a successive thread. Of course, the set of tentative shifts and the corresponding index set will be deflated accordingly

$$\widetilde{\Theta}_{k+1} = \widetilde{\Theta}_k \setminus \{\widehat{\vartheta}_{k+1}\} = \widetilde{\Theta}_k \setminus \{\widetilde{\vartheta}_\mu\}\,. \tag{21}$$

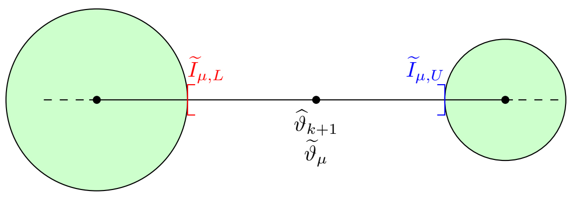

Fig. 4. Choice of the center to be processed.

Once the shift $\widehat{\vartheta}_{k+1} = \widetilde{\vartheta}_\mu$ is promoted for processing as in (20), the corresponding single-shift iteration

$$(\Lambda_{k+1}, \rho_{k+1}) \leftarrow \mathcal{S}(\widehat{\vartheta}_{k+1}, \rho_{k+1,0}) \tag{22}$$

is assigned to a computing thread and executed. The initial radius is initialized as

$$\rho_{k+1,0} = \alpha \frac{I_{U,k+1} - I_{L,k+1}}{2}\,, \tag{23}$$

where $\alpha \gtrsim 1$ in order to allow a small overlap with adjacent intervals.

D. *Completion of single-shift iteration and update*

As soon as a thread has completed processing a single-shift iteration, the corresponding shift is promoted $\widetilde{\vartheta}_k \to \vartheta_k$ and the corresponding radius ρ_k is used to update all data structures. Two cases may occur

- If the convergence disk covers the interval associated to the current shift $2\rho_k > I_{U,k} - I_{L,k}$, then I_k is simply deleted, since all eigenvalues within this interval have been found. It may be the case that, when the starting radius (23) is updated by the single-shift iteration to a larger value, the convergence disk includes some of the tentative shifts. Such shifts will be useless and are immedately deleted

$$\widetilde{\Theta}_k \leftarrow \widetilde{\Theta}_k \setminus \{\widetilde{\vartheta}_\nu \in \widetilde{\Theta}_k : \widetilde{\vartheta}_\nu \in [\vartheta_k - \rho_k, \vartheta_k + \rho_k]\} \tag{24}$$

- If the radius has been redefined to a smaller value by the single-shift iteration (see Fig. 5), the embedding interval is split into three portions

$$\begin{aligned}[I_{L,k}, I_{U,k}] = &[I_{L,k}, \vartheta_k - \rho_k] \cup \\ &[\vartheta_k - \rho_k, \vartheta_k + \rho_k] \cup [\vartheta_k + \rho_k, I_{U,k}]\end{aligned} \tag{25}$$

The inner portion does not need further processing, whereas the two outer portions will define respectively new tentative intervals $\widetilde{I}_{\mu_{k,1}}$, $\widetilde{I}_{\mu_{k,2}}$ with associated shifts

$$\widetilde{\vartheta}_{\mu_{k,1}} = \frac{I_{L,k} + (\vartheta_k - \rho_k)}{2} \tag{26}$$

$$\widetilde{\vartheta}_{\mu_{k,2}} = \frac{I_{U,k} + (\vartheta_k + \rho_k)}{2} \tag{27}$$

with $\mu_{k,i} = \max\{\mathcal{I}_k\} + i$, which in turn are added to the tentative set

$$\widetilde{\Theta}_{k+1} = \widetilde{\Theta}_k \cup \{\widetilde{\vartheta}_{\mu_{k,1}}, \widetilde{\vartheta}_{\mu_{k,2}}\} \tag{28}$$

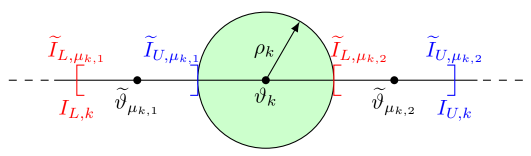
$\widetilde{I}_{L,\mu_{k,1}}$
$\widetilde{I}_{U,\mu_{k,1}}$
ρ_k
$\widetilde{I}_{L,\mu_{k,2}}$
$\widetilde{I}_{U,\mu_{k,2}}$
$I_{L,k}$
$\widetilde{\vartheta}_{\mu_{k,1}}$
ϑ_k
$\widetilde{\vartheta}_{\mu_{k,2}}$
$I_{U,k}$

Fig. 5. Quantities involved in the update of the data structures when $2\rho_k < I_{U,k} - I_{L,k}$.

E. *Termination*

The algorithm stops at the iteration K as soon as condition $\widetilde{\Theta}_K = \emptyset$ occurs, and at the same time there are no shifts $\widetilde{\vartheta}_\nu$ in the processing state, i.e.

$$\Theta_K = \{\vartheta_1, \ldots, \vartheta_K\}\,. \tag{29}$$

TABLE I
SUMMARY OF MAIN RESULTS. THE TABLE REPORTS DYNAMIC ORDER n, NUMBER OF PORTS p, AND NUMBER OF IMAGINARY HAMILTONIAN EIGENVALUES N_λ FOR EACH ANALYZED TEST CASE. THE LAST FOUR COLUMNS REPORT THE CPU TIME IN SECONDS FOR RUNNING THE SINGLE-THREAD SERIAL CODE ($\bar{\tau}_1$), THE PARALLEL CODE USING 16 THREADS (MEAN $\bar{\tau}_{16}$ AND WORST-CASE $\tau_{16}^{\max}$), AND THE CORRESPONDING AVERAGE SPEEDUP FACTOR $\bar{\eta}$.

Case #	n	p	N_λ	$\bar{\tau}_1$	$\bar{\tau}_{16}$	$\tau_{16}^{\max}$	$\bar{\eta}_{16}$
Case 1	1000	20	6	13.763	0.655	0.844	21.028
Case 2	1000	20	42	10.911	0.521	0.579	20.957
Case 3	1000	20	40	11.729	0.565	0.639	20.745
Case 4	1980	18	0	81.193	5.020	5.208	16.175
Case 5	2240	56	22	33.972	1.950	2.121	17.420
Case 6	1728	18	0	46.735	3.022	3.109	15.463
Case 7	1734	83	10	22.836	1.518	1.563	15.040
Case 8	1792	56	104	50.933	3.627	3.736	14.044
Case 9	1702	56	115	14.206	0.976	1.055	14.554
Case 10	4150	83	114	64.396	5.171	6.024	12.453
Case 11	1792	56	125	54.470	3.809	3.911	14.301
Case 12	2432	83	46	27.842	1.955	2.043	14.242

V. Results and Discussion

This section illustrates the performance of the proposed parallel solver on several test cases. The first four columns of Table I summarize the main features of each benchmark, namely the dynamic order n, the number of ports p, and the number of imaginary Hamiltonian eigenvalues N_λ to be computed. All numerical results have been produced on a IBM LS42 Blade server with four AMD Opteron quad-core processors running at 2.3 GHz, thus summing to 16 total computing cores, and with 2 GB RAM per core. This allows us to investigate the scalability, efficiency and speedup of the parallel solver with up to 16 parallel threads.

The proposed scheme relies on the iterative random selection of the starting vectors for each individual restart of the single-shift Arnoldi processes. Therefore, we expect some statistical variation in the code efficiency, since different choices of starting vectors might lead to a different number of restarts and/or iterations. This was in fact verified experimentally, as depicted in Fig. 6 for Case 5. The graph depicts $\bar{\eta}_t \pm \delta_t$, where t is the number of concurrent threads, $\bar{\eta}_t$ is the average and δ_t is the standard deviation of the speedup factor $\eta_t = \tau_t / \bar{\tau}_1$, computed over 20 independent runs, and τ_t is the CPU time.

The last four columns of Table I compare the average and the worst-case CPU time of the 16-thread parallel solver with the corresponding CPU time of the one-thread serial implementation, together with the average speedup factor. This table demonstrates a close-to-ideal speedup. In some cases (e.g., Cases 7–12) the speedup is slightly below expectation, reaching 78% of parallel efficiency in the worst case. For other cases (e.g., Cases 1–5), the speedup exceeds the ideal behavior and exhibits a superlinear trend. This variation and speedup bonus is a byproduct of the dynamic shift allocation process, which may eliminate from the tentative queue some shifts before they enter the processing queue. The occurence of such conditions depends on the problem itself and on the number of running threads, and is affected by additional

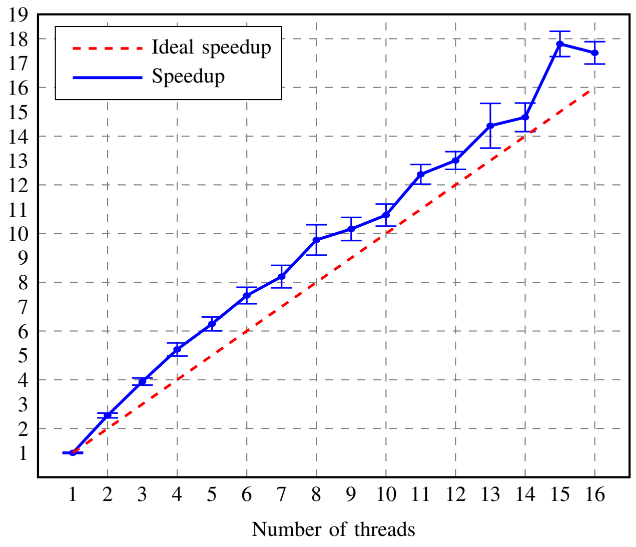

Fig. 6. Speedup factor vs number of threads for Case 5, compared to the ideal speedup (red dashed line). The dots and the vertical bars depict the average speedup and the associated standard deviation computed on a number of 20 independent runs (see text).

statistical variations due to the random selection of the starting vectors. However, the overall impact of such variability does not significantly affect the parallel efficiency, as demonstrated by the worst-case CPU time results.

VI. Conclusions

We have presented a parallel eigensolver for structured Hamiltonian matrices associated to interconnect state-space macromodels. The results demonstrate that the selective extraction of all purely imaginary eigenvalues can be performed in very few seconds even for large-scale problems, leading to an almost real-time algebraic characterization of macromodel passivity. The scalability and parallel efficiency achieved by our implementation on a 16-core machine are remarkable, showing good promise for the deployment of high-performance macromodeling tasks on next-generation computing platforms.

VII. Acknowledgement

The Authors are grateful to Dr. Katopis and Dr. Becker (IBM) for supporting this activity through an IBM Shared University Research (SUR) Grant. This work was supported in part by the Italian Ministry of University (MIUR) under a Program for the Development of Research of National Interest (PRIN grant #2008W5P2K) and in part by IdemWorks s.r.l.

References

[1] B. Gustavsen, A. Semlyen, “Rational approximation of frequency responses by vector fitting”, *IEEE Trans. Power Delivery*, Vol. 14, N. 3, July 1999, pp. 1052–1061.

[2] B. Gustavsen, ”Computer code for rational approximation of frequency dependent admittance matrices”, *IEEE Trans. Power Delivery*, Vol. 17, N. 4, October 2002, pp. 1093–1098.

[3] B. Gustavsen, A. Semlyen, “A robust approach for system identification in the frequency domain”, *IEEE Trans. Power Delivery*, Vol. 19, N. 3, July 2004, pp. 1167–1173.

[4] D. Deschrijver, B. Haegeman, T. Dhaene, ”Orthonormal Vector Fitting: A Robust Macromodeling Tool for Rational Approximation of Frequency Domain Responses”, *IEEE Transactions on Advanced Packaging*, vol. 30, n. 2, May 2007, pp. 216–225.

[5] S. Grivet-Talocia, M. Bandinu, “Improving the Convergence of Vector Fitting in Presence of Noise”, *IEEE Transactions on Electromagnetic Compatibility*, vol. 48, n. 1, pp. 104-120, February, 2006.

[6] B.D.O. Anderson and S.Vongpanitlerd, *Network Analysis and Synthesis*, Prentice-Hall, Englewood Cliffs, NJ, 1973.

[7] S. Boyd, V. Balakrishnan, P. Kabamba, “A bisection method for computing the H_∞ norm of a transfer matrix and related problems”, *Math. Control Signals Systems*, Vol. 2, 1989, pp. 207–219.

[8] S. Grivet-Talocia, “Passivity enforcement via perturbation of Hamiltonian matrices”, *IEEE Trans. CAS-I*, pp. 1755-1769, vol. 51, n. 9, September, 2004

[9] S. Grivet-Talocia, A. Ubolli “On the Generation of Large Passive Macromodels for Complex Interconnect Structures”, *IEEE Trans. Adv. Packaging*, vol. 29, No. 1, pp. 39–54, Feb. 2006

[10] D. Saraswat, R. Achar and M. Nakhla, ”Global Passivity Enforcement Algorithm for Macromodels of Interconnect Subnetworks Characterized by Tabulated Data”, *IEEE Transactions on VLSI Systems*, Vol. 13, No. 7, pp. 819–832, July 2005.

[11] C.P.Coelho, J.Phillips, L.M.Silveira, “A Convex Programming Approach for Generating Guaranteed Passive Approximations to Tabulated Frequency-Data”, *IEEE Trans. Computed-Aided Design of Integrated Circuits and Systems*, Vol. 23, No. 2, February 2004, pp. 293–301.

[12] H. Chen, J. Fang, “Enforcing Bounded Realness of S parameter through trace parameterization”, in *12th IEEE Topical Meeting on Electrical Performance of Electronic Packaging*, October 27–29, 2003, Princeton, NJ, pp. 291–294.

[13] B. Dumitrescu, ”Parameterization of Positive-Real Transfer Functions With Fixed Poles”, *IEEE Trans. CAS-I*, vol. 49, n. 4, April 2002, pp. 523-526.

[14] B. Gustavsen, A. Semlyen, “Enforcing passivity for admittance matrices approximated by rational functions”, *IEEE Trans. Power Systems*, Vol. 16, N. 1, Feb. 2001, pp. 97–104.

[15] B. Gustavsen, “Computed Code for Passivity Enforcement of Rational Macromodels by Residue Perturbation,” *IEEE Trans. Adv. Packaging*, vol. 30, pp. 209–215, May 2007.

[16] D. Saraswat, R. Achar and M. Nakhla, ”A Fast Algorithm and Practical Considerations For Passive Macromodeling Of Measured/Simulated Data”, *IEEE Transactions on Components, Packaging and Manufacturing Technology*, Vol. 27, N. 1, pp. 57–70, Feb. 2004.

[17] S. Grivet-Talocia, “An adaptive sampling technique for passivity characterization and enforcement of large interconnect macromodels,” *IEEE Trans. Adv. Packaging*, vol. 30, pp. 226–237, May 2007.

[18] A Lamecki and M. Mrozowski, “Equivalent SPICE Circuits With Guaranteed Passivity From Nonpassive Models,” IEEE Transactions on Microwave Theory And Techniques, Vol. 55, No. 3, March 2007, pp. 526–532.

[19] S. Grivet-Talocia, A. Ubolli “Passivity Enforcement with Relative Error Control”, *IEEE Trans. on Microwave Theory and Techniques*, vol. 55, No. 11, November 2007, pp. 2374–2383.

[20] Z. Ye, L. M. Silveira, and J. R. Phillips, “Fast and Reliable Passivity Assessment and Enforcement with Extended Hamiltonian Pencil,” in *International Conference on Computer Aided Design*, 2009, pp. 774–778.

[21] Z. Ye, L. M. Silveira, and J. R. Phillips, “Extended Hamiltonian Pencil for Passivity Assessment and Enforcement for S-parameter Systems,” in *DATE 2010 Conference*, pp. 1148–1152.

[22] Z. Zhang, C. U. Lei, and N.Wong, “GHM: A generalized Hamiltonian method for passivity test of impedance/admittance descriptor systems,” in *Proc. Int. Conf. Comput.-Aided Design, San Jose, CA*, Nov. 2009, pp. 767–773.

[23] Z. Zhang and N. Wong, “Passivity test of immittance descriptor systems based on generalized Hamiltonian methods,” *IEEE Trans. Circuits Syst. II: Express Briefs*, vol. 57, no. 1, pp. 61–65, Jan. 2010.

[24] V. Mehrmann and D. Watkins, “Structure-preserving methods for computing eigenpairs of large sparse skew-hamiltonian/hamiltonian pencils,” *SIAM J. Sci. Comput*, vol. 22, pp. 1905–1925, 2000.

[25] G. H. Golub, C. F. van Loan, *Matrix computations*, 3rd ed., Baltimore: Johns Hopkins University Press, 1996