A-LA-CARTE: A LAgrange interpolant using Chebyshev sampling for Accurately Representing Table modEls

Archit Gupta, Ahmet Gokcen Mahmutoglu, and Jaijeet Roychowdhury
Department of Electrical Engineering and Computer Sciences, University of California, Berkeley
Emails: {architgupta,amahmutoglu,jr}@berkeley.edu

***Abstract*—Recently, a method called STEAM, based on the MODSPEC modeling API, was published for constructing table-based representations from generic device models. With STEAM, using spline interpolants, device evaluation could be accelerated significantly, leading to somewhere between 8 to 15X speedup in various analysis, like DC, AC, and transient over the use of BSIM model for simulation. The accuracy for these analysis was reported to be less than 0.1% over the entire waveforms for circuits like inverters and differential pair.**

In this paper, we have explored the well established technology of Barycentric Lagrange Interpolation (BLI) with Chebyshev sample points for table-based modeling. This method allows us to use higher order polynomials in a numerically stable manner, approximating the core device functions to near machine-precision. The table-models thus created by our approach, dubbed A-LA-CARTE, are indistinguishable substitutes for the original compact model. These, at the same time, require far fewer mathematical operations to evaluate than most commonly used models like PSP, and BSIM. This makes it possible to now accelerate accuracy sensitive simulation algorithms like Harmonic Balance using table-based models by an order-of-magnitude.

Motivation

The key motivation for this paper comes from Table I, which shows a count of different mathematical operations required for running DC analysis on a CMOS inverter circuit using Penn State-Phillips (PSP), and Berkeley Short-channel IGFET Model (BSIM) models. This data was collected using the profiler in Octave while running a DC sweep using Berkeley Modelling and Prototyping Platform (MAPP). PSP, and BSIM models were translated from verilog-A using Verilog-A Parser and Processor (VAPP). Note that some of the most time consuming operations, involving calls to inbuilt mathematical functions, like *exp*, $\log, \ldots,$ *etc.*, have not been included in this table. The table simply lists the calls to the basic mathematical operations involved at the runtime of each device model.

For PSP model, each call involves roughly 11k multiplications, 5k additions, *etc.* The same numbers for BSIM end up being close to 2k multiplications, 800 additions *etc.*

When we use a table-based approximation for the PSP model, the operation count drops to roughly 50 multiplications, and 80 additions per model evaluation call. Approximation BSIM with a table based model yields a similar operation count and has not been described here.

Background and Previous Work

ModSpec

MODel SPECification (MODSPEC)[1] is a model specification format for describing devices. The key idea behind MODSPEC is to describe the behavior of a device (electronic or otherwise) using Differential-Algebraic Equation (DAE)s. In outlining the fundamental principals of the specification format, it is assumed that a device's *behavior* or *description* is comprised of: 1. "inputs", (labelled $\vec{x}$), 2. "internal unknowns", (labelled $\vec{y}$), and 3. "outputs" (labelled $\vec{z}$), which are explicitly computed using the "inputs" and "unknowns" using the following DAE relationship:

$$\begin{aligned}\vec{z} &= \frac{d}{dt}\, q_e(\vec{x},\vec{y}) + f_e(\vec{x},\vec{y}), \text{and}\\ \vec{0} &= \frac{d}{dt}\, q_i(\vec{x},\vec{y}) + f_i(\vec{x},\vec{y}).\end{aligned} \tag{1}$$

Readers are referred to [1, 2] for details and examples of MODSPEC, and specifically to [2] for its use and suitability for table-based device modeling. We build on the work from Spline-based Tables for Efficient and Accurate device Modeling (STEAM) to use Barycentric Lagrange Interpolation (BLI) with Chebyshev points in order to cut down memory requirements, while also increasing the accuracy at minimal loss in speedup.

Polynomial Interpolation

Since MODSPEC provides a complete description of a device using *four* functions, *i.e.*, f_e, q_e, f_i, and q_i, we can use several well-studied function-approximation techniques to approximate these functions. One of the most commonly used methods for function approximation is polynomial interpolation.

Lagrange Interpolation

We will consider the case studied previously in [2] with Spline interpolants, where, the value of a single-input, single-output function, $f(x)$, is known at a set of *knots*, labelled $a = x_0 < x_1 < x2 \ldots < x_n = b$. The Lagrange interpolation formula [3], interpolates between the known function values using the following polynomial expression:

$$L(x) = \sum_i \frac{f(x_i)\prod_{j\neq i}(x-x_j)}{\prod_{j\neq i} x_i - x_j} \tag{2}$$

Barycentric-Lagrange Interpolation

Barycentric Lagrange Interpolation formula, as described in [3], is a computationally efficient way of representing the Lagrange

Analysis	Model	Calls	Time Spent (%)	(*)	(+)	(-)	(/)	(**)
	PSP	7840	366.3s (77.5%)	85679787	37689915	18184544	11983440	5354720
DC	BSIM	5810	71.2s (71.9%)	10640142	4898368	2062882	2028271	888930
	STEAM (PSP)	7840	13.3 (46.8%)	412731	653472	223129	156825	0

TABLE I: Comparing the total number of mathematical operations for different transistor models in DC analysis

interpolant. The interpolant $B(x)$ is given by:

$$B(x) = \frac{\sum_i \frac{f(x_i)w_i}{(x-x_i)}}{\sum_i \frac{w_i}{(x-x_i)}}, \text{ where } w_i = \frac{1}{\prod_{j\neq i}(x_i - x_j)} \tag{3}$$

Chebyshev Points

Chebyshev points are defined as the roots or extremas of Chebyshev polynomials. Readers are referred to [4] for a clear and detailed write-up on this subject. As an illustrative example, the family of polynomials given by:

$$T_n(x) = \cos\left(n \cdot \cos^{-1}(x)\right), \tag{4}$$

is called Chebyshev polynomials of the first kind. Chebyshev points of the first kind of order n are the roots of the polynomial $T_n(x)$, given by:

$$r_i = \cos\left(\left(i + \frac{1}{2}\right)\frac{\pi}{n}\right) \tag{5}$$

Similarly, Chebyshev points of the second kind of order n are the extremas of the polynomial $T_n(x)$. These are given by the solutions of $T_n(x) = \pm 1$.

$$e_i = \cos\left(\frac{i \cdot \pi}{n}\right) \tag{6}$$

Chebyshev points of second kind are used in [5], and the tool described therein, called CHEBFUN because the set of Chebyshev points of second kind of order n are nested in the set of Chebyshev points of order $2n$. This makes incrementing the sample values of a function at the Chebyshev points computationally efficient. Besides, using Chebyshev points for interpolation with BLI has been shown to be numerically stable[5, 6].

A-LA-CARTE

In this section, we describe the main additions that we have made to the existing body of work in this area.

Applying BLI to device models

The continuity of derivatives of the functions describing a device model is essential for Newton-Raphson (NR) convergence in circuit simulation. In this context, cubic splines seem to be a natural choice for interpolating sample values of the device functions as they guarantee the continuity of not only the first, but also the second derivative across the entire domain. However, cubic splines or other higher order splines come with their own demerits.

Cubic splines suffer from numerical conditioning issues, the interpolation accuracy, for an instance degrades quadratically with the total number of sample points due to numerical conditioning. Moreover, cubic splines, or other low order polynomial interpolants, lack higher order derivatives, which are important for analyses relevant to Radio Frequency (RF) design, like Harmonic Balance (HB), Periodic Steady State (PSS), *etc.*. The lower order of polynomials also limits the accuracy of the overall interpolation scheme. We would, therefore, like to use a piecewise interpolant which is a polynomial with a reasonably high degree locally, and also has continuous derivatives across the entire domain. The requirement of our interpolant being a *piecewise polynomial* is important because, otherwise, a very high degree polynomial has to be fit to capture localized variations. This, is critical for several analyses, like AC, HB, *etc.*.

Multi-dimensional Piecewise BLI

Several approaches have been proposed to expand an interpolation algorithm to higher dimensions [7–9]. For our purpose, we have followed the approach of tensor products because of its simplicity, generalizability to multiple dimensions, and most importantly, speed. Note that for a tensor product approach to be feasible, the 1 dimensional interpolation algorithm must be linear in the sample values ($\vec{f}$ mentioned earlier). The algorithms for both Lagrange interpolant and BLI described in Sec. II are *linear*.

In order to extend the BLI interpolation algorithm, mentioned earlier in (3), consider a scalar function $f(x,y)$ of scalar variables x and y. Further, let us assume that we have sampled the function at a rectangular grid of values given by a cartesian product of points, $\vec{x} = [x_0, x_1, \ldots, x_n]$, and $\vec{y} = [y_0, y_1, \ldots, y_m]$. This gives us a total of mn sample values. The interpolant will essentially by the following polynomial expression:

$$B(x,y) = \frac{\sum_{i=0}^{n} \frac{g_i(y)w_i}{(x-x_i)}}{\sum_{i=0}^{n} \frac{w_i}{(x-x_i)}}, \text{ where } g_i(y) = \frac{\sum_{j=0}^{m} \frac{f(x_i,y_j)w_j}{(y-y_j)}}{\sum_{j=0}^{m} \frac{w_j}{(y-y_j)}} \tag{7}$$

Here, for a query point (x_q, y_q), most of the computation happens exactly once. The fractions $\frac{w_j}{(y-y_j)}$ have to be computed exactly once, making the computation of $g_i(y)$ a highly parallelizable matrix-multiplication. This is especially the case when we are interpolating device functions, which are not scalar, but a collection of multi-output functions $[\vec{f}_e, \vec{q}_e, \vec{f}_i, \vec{q}_i]^T$.

Results

We implemeted both BLI with Chebyshev sample points, and spline interpolants with uniform sample points in MAPP[10]. The interpolation algorithms themselves are quick to prototype and interface with the MODSPEC API. Generating a spline interpolant require some heavy-lifting. Unlike splines, generating a BLI interpolant doesn't require the solution to a matrix equation system, making the generation of the interpolant quite practically insignificant. It is important, however, to have extrapolation for BLI polynomials as they are very unstable

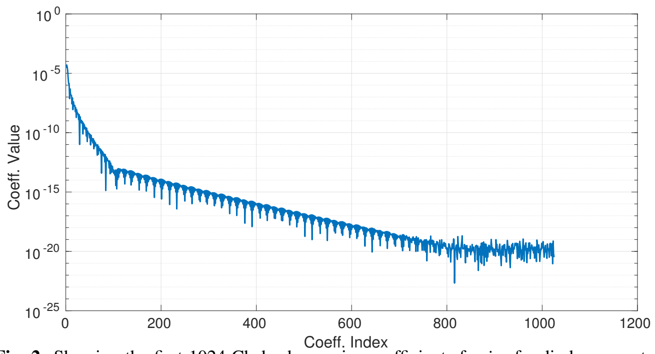


Fig. 2: Showing the first 1024 Chebyshev series coefficients for i_c of a diode-connected PSP model

Method	Pieces	Total Coefficients	Measured Error
Single Polynomial	1	257	5.24e-6
	1	2049	5.98e-13
Uniform Pieces	16	272	9.12e-7
	32	2080	4.89e-13
Hand-picked Split	8	70	4.92e-13

TABLE II: Required Polynomial Coefficients (or table size) for reconstructing PSP/f_e

outside the domain $[-1,1]$. We used linear extrapolation to maintain continuity and differentiability at the extrapolation boundary.

Evaluating Device Functions

We will now look at the applicability of BLI method for device models.

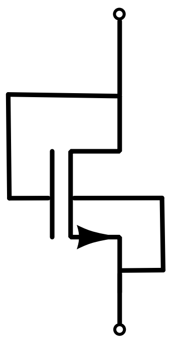


Fig. 1: Schematic of a diode connected MOSFET

Consider a diode connected Metal-Oxide Semiconductor Field Effect Transistor (MOSFET) as shown in Figure 1, modeled with BSIM and PSP models with only one input, called v_c. For simplicity, we have ignored terminal resistances from the model, making it a 1-port device. It can be completely characterized by providing, for any terminal voltage v_c, a DC current term i_c and a dynamic term q_c. A table-based model for the diode-connected MOSFET must reconstruct these two functions, $i_c(v_c)$, and $q_c(v_c)$.

?? shows the Chebyshev series coefficients for the input current i_c obtained from the PSP model. One can notice that after $\sim$ 800 indices, the coefficients are not numerically significant to contribute to any error in the table-based model. Therefore, if we were to construct a BLI with more than 800 Chebyshev sample points, it would reconstruct the model to near-machine precision.

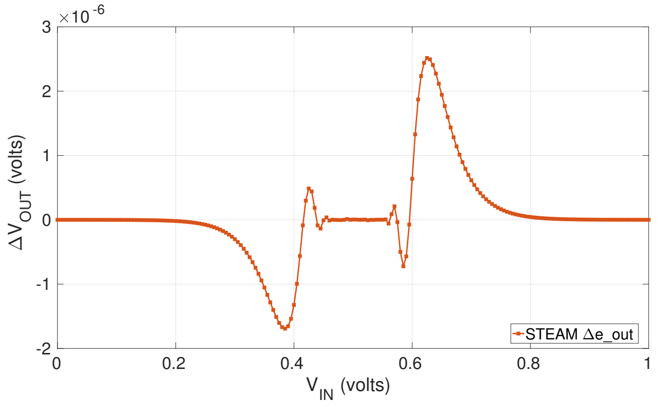
If one looks at the segment of Table II that refers to *single piece* as the method, one can notice that the mean point-wise relative error does indeed drop significantly when the number of sample points is dropped from 513 to 1025.

Elementary analyses algorithms

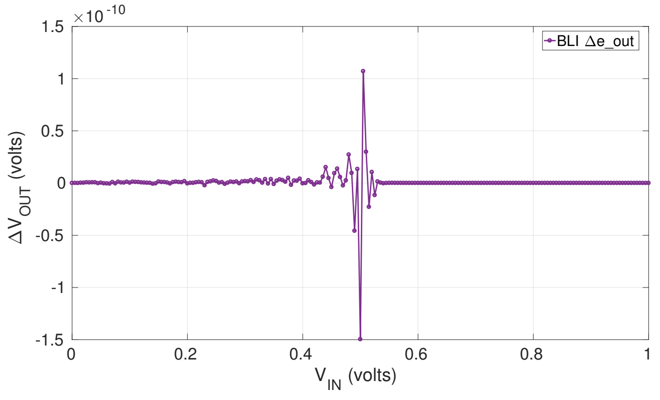
For the subsequent experiments involving analyses algorithms, we compare PSP and BSIM algorithms with table-based models constructed using spline interpolants and BLI. The 2-dimensional piecewise BLI interpolant was constructed with 32 pieces uniformly splitting the domain $[-1,1]$V. Each piece had a local polynomial of degree 33 in both the dimensions. The 2-dimensional spline interpolant was constructed on a uniform grid of 165x165 points for it to have the same number of sample points as the piecewise BLI interpolant. The accuracy and speedup for these interpolants *wrt*to the baseline models, PSP, and BLI, can be seen in **??** and **??** respectively.

Method	Pieces	Total Coefficients	Measured Error
Single Polynomial	1	257	3.81e-4
	1	2049	4.68e-6
Uniformly Spaced	16	272	6.59e-5
	32	2080	1.17e-7
Hand-picked Split	8	40	3.35e-14

TABLE III: Required Polynomial Coefficients (or table size) for reconstructing BSIM/f_e



(a) STEAM



(b) A-LA-CARTE

Fig. 3: Error in simulating a CMOS inverter for QSS using STEAM and A-LA-CARTE

Quiescent Steady State (QSS) and Transient Analysis

Having compared our approach with STEAM for evaluating device functions, *i.e.*, f_e, q_e, f_i, and q_i, we now analyze the overall accuracy of QSS (or DC analysis) and Transient analysis using the two approaches. We ran a QSS sweep for a **CMOS!** (**CMOS!**) inverter , and Gilbert Cell.

Figure 3 show simulation results for a CMOS Inverter comparing an approximation with spline interpolants and BLI with

CIRCUIT	Error BLI	Speedup BLI	Error STEAM	Speedup STEAM
CMOS Inverter	6.0e-12	2.6	6.2e-4	2.5
Ring Oscillator	5.0e-12	2.9	1.5e-3	2.7
Differential Pair[1]	3.8e-12	2.3	1.1e-8	2.2
Gilbert Cell	8.1e-12	2.8	7.9e-8	2.6

TABLE IV: Runtime and Error analysis for Transient Analysis with various circuits using BSIM Model

CIRCUIT	Error BLI	Speedup BLI	Error STEAM	Speedup STEAM
CMOS Inverter	1.1e-13	8.6	1.8e-9	13.2
Differential Pair	7.9e-16	11.0	1.4e-10	10.8
Gilbert Cell	5.1e-16	13.7	2.8e-10	12.9

TABLE V: Runtime and Error analysis for QSS with various circuits using PSP Model

Chebyshev sample points.

Accuracy Sensitive Applications

The error numbers obtained in previous sections (Sec. IV-A) motivate the use of our approach for table-based device modeling in these applications.

Some of the methods used for analyzing analog circuits that have periodic behavior include PSS (or Shooting) and HB. In this section, we compare the results from the approach used in [2] and A-LA-CARTE. Accuracy sensitive applications, like RF design, highlight the important differences in accuracy between the two methods. This is especially true for a method like HB that implicitly depends on the accuracy of higher order derivatives

Harmonic Balance

HB is a mixed time-frequency domain analysis that is used for analyzing oscillatory circuits. It can be applied to both autonomous systems, like oscillators, and systems driven by periodic signals like amplifiers. We applied both the approaches to accelerate HB for the kinds of simulations. For the former case, *i.e.*, analysis of oscillators, HB computes both the oscillation frequency and the amplitude of various harmonics corresponding the oscillation frequency.

References

[1] D. Amsallem and J. Roychowdhury. ModSpec: An open, flexible specification framework for multi-domain device modelling. In *Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on*, pages 367–374. IEEE, 2011.

[2] A. Gupta, T. Wang, A. G. Mahmutoglu, and J. S. Roychowdhury. Steam: Spline-based tables for efficient and accurate device modelling. In *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 463–468, Jan 2017.

[3] Jean-Paul Berrut and Lloyd N Trefethen. Barycentric lagrange interpolation. *SIAM review*, 46(3):501–517, 2004.

[4] A. Gil, J. Segura, and N. Temme. *Numerical Methods for Special Functions*. Society for Industrial and Applied Mathematics, 2007.

[5] Rodrigo B Platte and Lloyd N Trefethen. Chebfun: a new kind of numerical computing. *Progress in industrial mathematics at ECMI 2008*, pages 69–87, 2010.

[6] Nicholas J Higham. The numerical stability of barycentric lagrange interpolation. *IMA Journal of Numerical Analysis*, 24(4):547–556, 2004.

[7] Carl De Boor. *A practical guide to splines*, volume 27.

[8] Volker Barthelmann, Erich Novak, and Klaus Ritter. High dimensional polynomial interpolation on sparse grids. *Advances in Computational Mathematics*, 12(4):273–288, 2000.

[9] Alex Townsend and Lloyd N Trefethen. An extension of chebfun to two dimensions. *SIAM Journal on Scientific Computing*, 35(6):C495–C518, 2013.

[10] T. Wang, A.V. Karthik, B. Wu and J. Roychowdhury. MAPP: A platform for prototyping algorithms and models quickly and easily. In *IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization (NEMO)*, pages 1–3. IEEE, 2015.

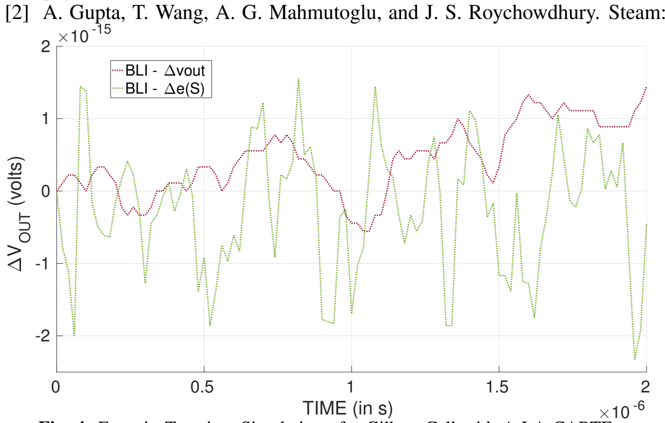
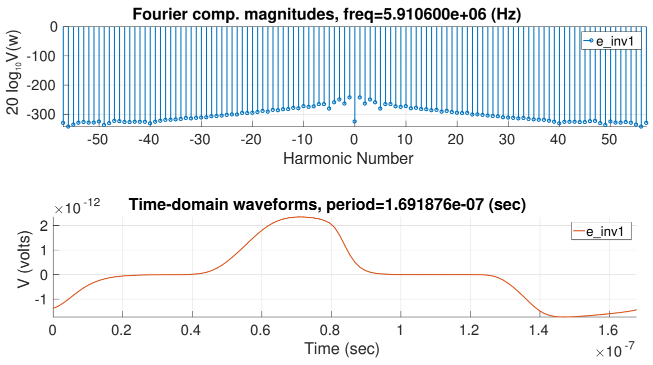
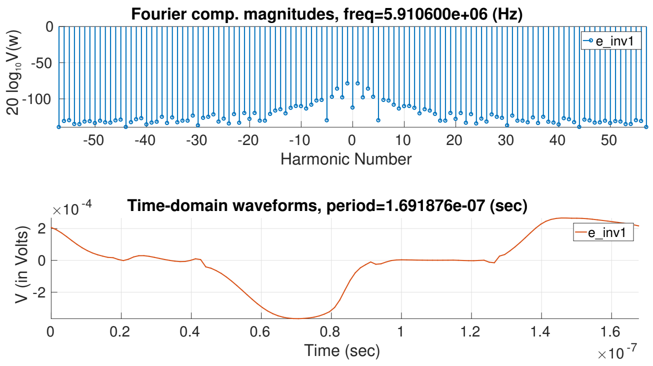


Fig. 4: Error in Transient Simulation of a Gilbert Cell with A-LA-CARTE



(a) BLI



(b) STEAM

Fig. 5: Time-Frequency domain error in Harmonic balance for a 3-stage ring oscillator circuit.