

A Robust and Efficient Harmonic Balance (HB) using Direct Solution of HB Jacobian

Amit Mehrotra
Berkeley Design Automation
amit.mehrotra@berkeley-da.com

Abhishek Somani
Berkeley Design Automation
asomani@berkeley-da.com

ABSTRACT

In this paper we introduce a new method of performing direct solution of the harmonic balance Jacobian. For examples with moderate number of harmonics and moderate to strong nonlinearities, we demonstrate that the direct solver has far superior performance with a moderate increase in memory compared to the best preconditioned iterative solvers. This solver is especially suited for Fourier envelope analysis where the number of harmonics is small, circuits are nonlinear and Jacobian bypass can be used for additional speed. For examples with large number of harmonics and moderate to strong nonlinearities, the performance advantage is maintained but the memory requirements increase. We propose efficient preconditioners based on direct solution of harmonic balance matrices which provide the user with a memory-speed trade-off.

Categories and Subject Descriptors

B.7.2 [Integrated Circuits]: Design Aids—simulation

General Terms

Algorithms, Performance

Keywords

Simulation, Harmonic Balance, Preconditioning

1. INTRODUCTION

Harmonic balance analysis (HB) has been the preferred simulation method for RF and microwave applications for a long time [18, 9, 7]. For circuits where the nonlinearities are moderate and the signal transitions are not very sharp, harmonic balance is the preferred method of simulation over time-domain techniques such as shooting Newton because it gives unmatched performance and accuracy [17, 5, 13]. Furthermore, RF and microwave applications rely heavily on measured frequency domain data such as *s*-parameters of linear networks. These can be directly simulated in frequency domain without requiring expensive convolutions or fitting. The drawback of traditional HB implementations is that the underlying Jacobian is large and not very sparse and it is very inefficient to factor and solve directly. Practical HB implementations tried to circumvent the problem by separating the linear and nonlinear portions of the circuit and formulating the problem so as to achieve balance between the harmonics of the waveforms of the two portions (hence the name harmonic balance). Since there is no coupling between harmonics in the linear portion, it can be arbitrarily large and

the limitation of this approach was the size and nonlinearity of the portion which contained all the nonlinear elements [18].

Krylov subspace methods addressed these limitations of HB [5, 12]. These iterative methods do not require the matrix to be explicitly formed, let alone factored. Instead they require the matrix-vector product to be formed at each iteration. As shown in Section 2, matrix-vector product with the HB Jacobian can be formed very efficiently using a series of FFTs and sparse matrix-vector multiplications with circuit matrices. The relative success of iterative methods with increasing circuit sizes resulted in restricting the use of traditional direct solution method to small circuits only.

However, the effectiveness of these iterative methods critically depends on selecting appropriate preconditioners, otherwise the number of iterations becomes prohibitively large. In addition, as the complexity and nonlinearity of circuits increases, Krylov subspace based HB engines require more advanced preconditioners [11] and still require a large number of iterations to reliably solve the system of linear equations, and in many cases, are unable to solve the system (See Section 3 for a brief description of existing preconditioners for HB Jacobian).

In this paper, we develop a direct method for solving linear systems that have the HB Jacobian as the coefficient matrix. Unlike previous such attempts which suffer from the problems mentioned earlier, we exploit the unique structure of the HB Jacobian and develop a highly specialized direct solver specifically for the problem (Section 4). We show that compared to the best preconditioned Krylov subspace methods, this direct solver gives competitive performance on circuits with mild nonlinearities and far superior performance and robustness for circuits with moderate to strong nonlinearities. We also show that this specialized direct solver gives far superior performance compared to a generic sparse direct solver. Moreover, for Fourier envelope applications, the direct solver lends itself very naturally to Jacobian bypass and therefore gives additional performance advantage as demonstrated in Section 5.

As the number of harmonics increases, as in three or more tone HB problems or users request a larger number of harmonics, the memory requirement of direct method based HB solvers increases rapidly. To address this issue, we develop new preconditioners for iterative methods based on the direct solver for HB Jacobian for problems which have a large number of harmonics (Section 6). These preconditioners provide a direct trade-off between performance and memory. In some cases they provide both speed and memory improvements.

2. HB PRELIMINARIES

We first review the basic theory of Harmonic Balance and introduce the notations we will use in this paper. Consider a non-autonomous circuit whose equations are given by

$$\int_{-\infty}^t y(t-s)x(s)ds + \frac{dq(x(t))}{dt} + f(x(t)) + b(t) = 0 \quad (1)$$

where $x(t) \in \mathbb{R}^n$ represents the state variables in the circuit, n is the circuit size, y represents the matrix-valued impulse response

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

DAC'09, July 26-31, 2009, San Francisco, California, USA
Copyright 2009 ACM 978-1-60558-497-3/09/07....10.00

function of frequency-domain linear elements (such as s parameters), $q : \mathbb{R}^n \rightarrow \mathbb{R}^n$ represents the nonlinear charge and flux storage in the circuit, $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ represents the memoryless nonlinearities and $b : \mathbb{R} \rightarrow \mathbb{R}^n$ represents the time-dependent excitations in the circuit which are assumed to be periodic with period T . Since the circuit is nonautonomous, the circuit steady-state response $x(t)$ will also be periodic with period T and its functions $q(x)$ and $f(x)$ are also T -periodic. Therefore all these waveforms can be expanded in Fourier series as follows:

$$b(t) = \sum_{i=-\infty}^{\infty} B_i \exp(j2\pi i f_0 t), \quad x(t) = \sum_{i=-\infty}^{\infty} X_i \exp(j2\pi i f_0 t)$$

$$f(t) = \sum_{i=-\infty}^{\infty} F_i \exp(j2\pi i f_0 t), \quad \text{and} \quad q(t) = \sum_{i=-\infty}^{\infty} Q_i \exp(j2\pi i f_0 t)$$

where $f_0 = \frac{1}{T}$.

(1) can be written in frequency-domain as

$$\sum_{i=-\infty}^{\infty} [Y_i X_i + j2\pi i f_0 Q_i + F_i + B_i] \exp(j2\pi i f_0 t) = 0$$

where $Y(f) = \int_{-\infty}^{\infty} y(t) \exp(-j2\pi f t) dt$ is the Fourier transform of $y(t)$ and $Y_i = Y(i f_0)$. Since $\exp(j2\pi i f_0 t)$ are orthogonal, it follows that

$$Y_i X_i + j2\pi i f_0 Q_i + F_i + B_i = 0$$

$\forall i$. In practice the infinite summations are truncated to a finite number of harmonics k . Collocating the above equation at $i \in [-k, k]$, we have

$$Y_{-k} X_{-k} + j2\pi(-k) f_0 Q_{-k} + F_{-k} + B_{-k} = 0$$

$$\dots$$

$$Y_0 X_0 + j2\pi 0 f_0 Q_0 + F_0 + B_0 = 0$$

$$\dots$$

$$Y_k X_k + j2\pi k f_0 Q_k + F_k + B_k = 0$$

In matrix form

$$F_{hb} = \mathcal{Y} \mathcal{X} + j2\pi f_0 \Omega \mathcal{Q} + \mathcal{F} + \mathcal{B} = 0 \quad (2)$$

where, $\Omega = \text{diag}([-kI \quad \dots \quad 0 \quad \dots \quad kI])$ and

$$\mathcal{Q} = [Q_{-k}, \dots, Q_0, \dots, Q_k]^T$$

\mathcal{B} , \mathcal{F} and \mathcal{X} are similarly defined and

$$\mathcal{Y} = \text{diag}[Y_{-k} \quad \dots \quad 0 \quad \dots \quad Y_k]$$

(2) represents a system of $m(2k+1)$ nonlinear equations in $m(2k+1)$ unknowns \mathcal{X} which can be solved using Newton's method. The Jacobian for (2) is given by

$$J_{hb} = \mathcal{Y} + j2\pi f_0 \Omega \frac{\partial \mathcal{Q}}{\partial \mathcal{X}} + \frac{\partial \mathcal{F}}{\partial \mathcal{X}}$$

It can be easily shown that

$$J_{hb} = \mathcal{Y} + j2\pi f_0 \Omega \Gamma \mathcal{C} \Gamma^{-1} + \Gamma \mathcal{G} \Gamma^{-1} \quad (3)$$

where

$$\mathcal{C} = \begin{bmatrix} C(t_1) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & C(t_{2k+1}) \end{bmatrix}$$

where $C(t_i) = \left. \frac{dq}{dx} \right|_{x(t_i)}$, \mathcal{G} is also similarly defined. Γ represents time to frequency translation and consists of a series of permutations and Fourier transforms. This Jacobian is large and not very sparse and therefore a generic dense or sparse solver will not be able to solve this system very efficiently. Note that

$$\Gamma \mathcal{C} \Gamma^{-1} = \begin{bmatrix} C_0 & C_1 & \dots & & C_{-k} \\ C_{-k} & C_0 & C_1 & \dots & \\ & & \ddots & & \\ C_1 & C_2 & \dots & C_{-k} & C_0 \end{bmatrix} \quad (4)$$

where C_i is the i th Fourier coefficient of $C(t)$. I.e., the matrix block-circulant.

However, Krylov subspace methods only require a subroutines to perform matrix vector multiplication with J_{hb} which can be achieved using permutations (no cost), Fourier transforms ($\mathcal{O}(n(2k+1) \log(2k+1))$) and sparse matrix vector multiplications $\mathcal{O}(n(2k+1))$. Therefore *if the problem is properly preconditioned*, Krylov subspace methods converge quickly to the solution.

3. EXISTING PRECONDITIONERS FOR HB JACOBIAN

In this section we briefly review various preconditioners we will be using to compare with our approach. A more elaborate and detailed review can be found in [4].

3.1 Averaging preconditioner

If $G(t_i)$ s and $C(t_i)$ s can be assumed constant then (3) reduces to

$$J_{hb_{avg}} = \mathcal{Y} + \begin{bmatrix} j2\pi(-k) f_0 C_{avg} + G_{avg} & \dots & 0 \\ & \ddots & \\ & & \dots & j2\pi k f_0 C_{avg} + G_{avg} \end{bmatrix}$$

which is a block diagonal matrix and each block has the sparsity structure of the circuit transient matrix, i.e., $J_{hb_{avg}}$ can be very easily inverted. For problems with mild nonlinearities, this preconditioner works extremely well and is probably the best preconditioner for such problems.

3.2 Averaging preconditioner with one step correction

As the circuit becomes more nonlinear, off-diagonal entries in J_{hb} become more dominant and $J_{hb_{avg}}$ becomes less effective. A one-step correction scheme includes them in the following manner [4]

$$J_{hb} = J_{hb_{avg}} + (j2\pi f_0 \Omega \Gamma \mathcal{C}_{diff} \Gamma^{-1} + \Gamma \mathcal{G}_{diff} \Gamma^{-1})$$

where $\mathcal{G}_{diff} = \mathcal{G} - \mathcal{G}_{avg}$. J_{hb} is expanded in Taylor series to find a better approximation of J_{hb}^{-1} as follows:

$$J^{-1} = \left[I + J_{hb_{avg}}^{-1} (j2\pi f_0 \Omega \Gamma \mathcal{C}_{diff} \Gamma^{-1} + \Gamma \mathcal{G}_{diff} \Gamma^{-1}) \right]^{-1} J_{hb_{avg}}^{-1}$$

$$\approx \left[I - J_{hb_{avg}}^{-1} (j2\pi f_0 \Omega \Gamma \mathcal{C}_{diff} \Gamma^{-1} + \Gamma \mathcal{G}_{diff} \Gamma^{-1}) \right] J_{hb_{avg}}^{-1}$$

3.3 Averaging preconditioner with super diagonals

Due to nonlinearities, if a few of the harmonics are large, then they can also be included in $J_{hb_{avg}}$. However, inspecting the form of the circulant matrix in (4) it is obvious that if all entries of a given harmonic are included, the matrix becomes difficult to invert. Therefore, we only include entries in the super-diagonal and discard the entries in the subdiagonal. The resulting matrix is block upper triangular and therefore easier to invert. However, if the harmonic index is large, then the corresponding entries are further away from the block-diagonal and more entries of that harmonic are discarded from the lower diagonal part of the matrix and the preconditioner becomes less effective. This is specially true for multi-tone problems, where the artificial frequency map may place waveforms with significant harmonic content, far away from the diagonal.

3.4 One-step correction on averaging preconditioner with super diagonals

This is similar in spirit to averaging preconditioner with one-step correction where the difference matrix is generated by subtracting the averaging preconditioner along with the superdiagonals from the full harmonic balance preconditioner. One-step correction is performed using this modified difference matrix.

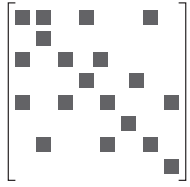


Figure 1: Sparsity structure of HB Jacobian

3.5 Finite-difference Jacobian

If the circuit is highly nonlinear then the preconditioner of the finite difference Jacobian can be used. The form of this preconditioner is

$$\begin{bmatrix} \frac{C(t_1)}{h_1} + G(t_1) & 0 & \dots & 0 \\ -\frac{C(t_1)}{h_2} & \frac{C(t_2)}{h_2} + G(t_2) & & \\ & \ddots & & \\ \dots & 0 & -\frac{C(t_{n-1})}{h_n} & \frac{C(t_n)}{h_n} + G(t_n) \end{bmatrix}$$

where $h_i = t_i - t_{i-1}$. The limitation of this method is that it cannot be used for harmonic balance problems with more than one tones. Another option is to use the full finite-difference Jacobian as a preconditioner.

$$\begin{bmatrix} \frac{C(t_1)}{h_1} + G(t_1) & 0 & \dots & -\frac{C(t_n)}{h_1} \\ -\frac{C(t_1)}{h_2} & \frac{C(t_2)}{h_2} + G(t_2) & & \\ & \ddots & & \\ \dots & 0 & -\frac{C(t_{n-1})}{h_n} & \frac{C(t_n)}{h_n} + G(t_n) \end{bmatrix}$$

This works better for more nonlinear problems compared to the preconditioner of finite-difference but is more expensive to apply since every solve with this requires another set of Krylov subspace iterations.

3.6 Schur-complement preconditioner

This preconditioner is based on the assumption that the matrix can be permuted such that the number of columns (c) containing the nonlinear entries are small $c \ll n$ and these columns can be permuted to the right of the matrix. The first $n - c$ columns are factored using a block diagonal solver, though a rectangular LU decomposition is required. The resulting Schur complement matrix can be factored either exactly or using incomplete LU [11]. This preconditioner relies on the fact that the number of columns with nonlinear elements is very small compared to the entire circuit and also that permuting these columns to the right of the matrix does not cause significant fills. Both these assumptions are not generally true.

4. DIRECT METHOD FOR HB JACOBIAN

4.1 Basic procedure

Note that the structure of J_{hb} is similar to (4) except that it is not block circulant as $\Gamma C \Gamma^{-1}$. In this formulation, all variables of a single harmonic are grouped together. If we apply a permutation to J_{hb} such that all the harmonics of a single variable are grouped together then the matrix structure will look as in Figure 1 [5]. At the top level, the matrix has the same sparsity structure as the transient matrix but the difference is that in the transient matrix, each entry is a number whereas in this matrix, each “entry” is a block matrix of size $(2k + 1) \times (2k + 1)$.

This matrix can be efficiently factored in the following manner. We first form a test matrix which has the same sparsity structure as the transient matrix and place *representative* entries and factor it using a modern sparse LU solver. The solver, along with factoring the matrix will determine a row and column permutation and a row and column scaling scheme which will minimize the number of fills in factoring the test matrix. Moreover, it will also provide the locations of the fills in the L and U matrices.

Given this information, it is trivial to write a matrix factor routine for J_{hb} as follows:

For each column j

1. identify the permuted column, perform row permutation and row and column scaling

2. for each row i

(a) if there is a structural nonzero entry in U

$$U_{ij} = A_{ij} - \sum_{k=1}^{i-1} L_{ik} U_{kj} \text{ if } i \leq j$$

(b) if there is a structural nonzero entry in L

$$L_{ij} = \left(A_{ij} - \sum_{k=1}^{j-1} L_{ik} U_{kj} \right) U_{jj}^{-1} \text{ if } i > j$$

Here A_{ij} , U_{ij} , L_{ij} are the dense matrices of size $(2k+1) \times (2k+1)$ located at the $(i, j)^{th}$ entry of J_{hb} , it's upper (U) and lower (L) triangular factors respectively. These operations can be performed very efficiently using aggressively optimized BLAS [3] and LAPACK [1] implementations for the target CPU architecture. The test matrix should be constructed so that it mimics the properties of J_{hb} . For instance, if there is an entry in the transient matrix at $(i, j)^{th}$ location where only C matrix entry is present, then the corresponding block in J_{hb} will be $j2\pi f_0 \Omega C_{i,j}$ which is singular. This entry should not be a choice of pivot for the test matrix otherwise the matrix factor of J_{hb} will fail. This can be avoided by introducing numerical zeroes in such locations which will prevent these entries to be used as pivots when factoring the test matrix.

Furthermore, these BLAS and LAPACK routines are also available in parallel form and multi-threaded versions of these routines can be used to further improve performance as the performance of these basic routines scales almost linearly with the number of processors. However, in this paper all the results are reported with the sequential implementations of BLAS and LAPACK since we are comparing performance with sequential implementation of preconditioners in Krylov subspace methods.

4.2 Complexity analysis

We now calculate the storage and computational complexity of direct solve and compare it with iterative solvers. Note that J_{hb} need not be constructed explicitly and only the factors L and U are required. During the factorization, the block matrices of J_{hb} can be constructed on the fly and each block is only needed once. Therefore the storage requirement of the direct solver is

$$(nnzl + nnzu)(2k + 1)^2$$

where $nnzu$ and $nnzl$ are the nonzeros in U and L respectively. For preconditioned Krylov subspace methods, typically C and G matrices are stored in time domain for efficient matrix vector multiplication with J_{hb} and the storage requirement is $2nn_t$ where n_t is the number of time points. For multitone problems the artificial frequency map can result in large n_t . However, for the direct solver, the storage requirement is quadratic in k and therefore can be quite steep for multitone problems.

The computation complexity of the direct solver is

$$\mathcal{O}((nnz^{\alpha_l} + nnzu^{\alpha_u})(2k + 1)^3)$$

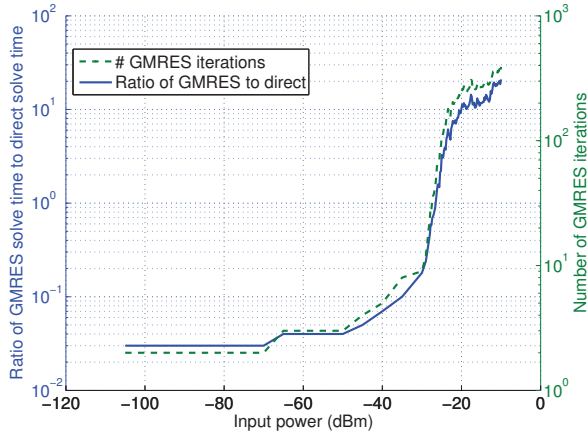
where $\alpha_l, \alpha_u > 1$ but are close to 1. The complexity of a Krylov subspace based method is somewhat less straight-forward to express. If a Krylov subspace method converges in l iterations, then the complexity of HB matrix vector multiply is

$$l\mathcal{O}((nnzc + nnzg)(2k + 1) \log(2k + 1))$$

where $nnzc$ and $nnzg$ are the number of nonzeros in C and G respectively. For an Arnoldi based Krylov subspace method such as GMRES [16] (preferred over Lanczos based methods such as

Table 1: Run time comparison of HB using direct solver and best preconditioned Krylov subspace solver

Example	circuit size	number of tones	number of harmonics	problem size	direct solver		best Krylov solver		
					time (s)	memory (GB)	time (s)	memory (GB)	iterations
mixer	58	1	40	4698	0.45	0.086	0.068	0.063	15
LNA + mixer	1135	1	30	69235	23.4	0.45	15.4	0.16	52
		3	125	284885	991	6.5	602	1.31	92
receiver 1	2158	1	15	66898	28.5	0.28	23.5	0.16	33
		3	125	541658	3297	12.7	962	1.66	58
receiver 2	8254	3	140	2319374	20917	64.4	DNF	–	–
LNA + mixer + filter	8444	1	20	346204	104.6	1.42	242.1	0.56	124
		3	80	1359484	3661	20	DNF	–	–
PA	341592	1	7	5123880	46582	25.6	DNF	–	–

**Figure 2: Ratio of GMRES solve time to the direct solve time as a function of input power for an LNA.**

TFQMR [6] due to superior convergence and numerical properties), the complexity of running l iterations is $\mathcal{O}(l^2 n(2k+1))$. The complexity of applying the preconditioner per iteration is typically $\mathcal{O}(nnz^\alpha(2k+1))$.

The key observation is that if the number of Krylov subspace iterations l required to achieve convergence is greater than $2k+1$, then the computational complexity of a Krylov subspace based method is similar to the direct method! I.e., for such problems, the direct method can be competitive or significantly superior in performance compared to Krylov subspace methods.

Specifically for problems with moderate to strong nonlinearities, the number of Krylov subspace iterations can be very large even though the number of harmonics requested is moderate and it is *precisely* these situations where the direct method outperforms its more decorated Krylov subspace siblings.

4.3 A representative example

Before we present a comprehensive set of numerical results, it is instructive to compare the direct and the preconditioner Krylov solver performance on a simple example. The circuit is a low noise amplifier with two tones of equal power applied at the input. The frequency separation of these tones (Δf) is much smaller than their respective frequencies (i.e., $f_1, f_2 \gg \Delta f$). Two-tone harmonic balance is run with all harmonic mixes which add up to ten or less. The input power is swept from -105 dBm to -10 dBm.

Figure 2 shows the relative time for solving the harmonic balance Jacobian using the best preconditioned GMRES to the CPU for the direct solver, along with the number of GMRES iterations. For low powers, we used the averaging preconditioner but for high powers, we needed to switch to the 1-step correction preconditioner. It is obvious from the figure that for small input power levels, the LNA operates almost linearly and GMRES is much faster than direct solver. However, as the input power increases which causes the circuit to become more nonlinear, the number of GMRES iterations and therefore the CPU time

of GMRES become much worse compared to the direct solver. Note that since the number of harmonics required over the entire power sweep is constant, direct solver CPU time is constant.

4.4 Numerical results

The above procedure was implemented in our experimental HB simulator along with most of the above mentioned preconditioners. In subsequent tables, all examples which required less than 2.4 GB memory for direct solver were run on an Intel 32 bit machine and used Intel Math Kernel Library (MKL) [8] for LAPACK and BLAS. All examples requiring more than 2.4 GB were run on an AMD Opteron machine and use Automatically Tuned Linear Algebra Subroutines (ATLAS) [2]. Both direct solver and best preconditioned Krylov subspace methods were run on the same unloaded machine.

Table 1 shows the CPU time and memory comparisons between the direct solver and the best preconditioned Krylov subspace solver (in terms of CPU time), for various circuits with increasing size and harmonics along with the number of Krylov subspace iterations for the iterative methods. It is clear that direct solver is slower (but not excessively slow) for mildly nonlinear problems compared to Krylov subspace methods but becomes markedly superior as the circuit complexity and nonlinearity increases. In many examples, Krylov subspace methods exceeded the generous iteration limit of 500 and are reported as DNFs. The mixer example is a four quadrant Gilbert's cell mixer. Averaging preconditioner was the fastest on this example. For the LNA + mixer example, the averaging preconditioner with 1-step correction (Section 3.4) was the fastest. For the receiver 1 example, averaging preconditioner with super-diagonals corresponding to the first and second harmonic (Section 3.3) was the fastest. The LNA + mixer + filter example converged in 1-tone HB using only the finite difference preconditioner and this could not be used in the 3-tone HB and no other preconditioner could be used for this circuit for 3-tone HB.

Table 1 also points out a very desirable characteristic of the direct solver: robustness. If the process fits in the CPU memory and the test matrix is formed correctly, HB Jacobian solve will succeed for all the Newton iterations. The same cannot be claimed for Krylov subspace solvers which may exceed the iteration limit as Newton is progressing and may require a preconditioner change in the middle or may abort all together.

4.5 Comparison with general purpose sparse LU solvers (GLU)

It is instructive to compare the factor time of the proposed specialized solver for HB Jacobian with the factor time of a GLU for this matrix. Table 2 shows the ratio of the factor time of the general purpose sparse solver with the factor time of our proposed direct solver. It is clear that the proposed solver is much more efficient, and this efficiency increases with the example size. Further, for many larger examples in Tables 1 and 3, GLU ran out of memory even on a machine with 64 GB RAM.

5. FOURIER ENVELOPE ANALYSIS

We will first briefly review the Fourier envelope analysis in the

Table 3: Run time comparison of Fourier envelope using direct solver and best preconditioned Krylov subspace solver

Examples	circuit size	number of harmonics	problem size	direct solver			best Krylov solver		
				time (sec)	memory (GB)	bypass %	time (sec)	memory (GB)	iterations
LNA + mixer	1135	15	35185	4835	0.22	31	9522	0.20	60
receiver 1	2158	15	66898	1951	0.62	63	3628	0.35	34
LNA + mixer + filter	8444	20	346204	5547	1.45	49	27467	0.68	102
receiver 3	14685	10	308385	17467	1.0	47	90734	0.756	86
GSM transmitter	29053	13	784431	16570	2.57	76	109540	1.42	52
extracted receiver	32466	10	681786	6590	1.84	58	59024	1.34	42
LNA + mixer + filter	44150	20	1810150	360250	15.36	51	311040	3.63	102

Table 2: Ratio of factor time of a general purpose sparse solver to the factor time of the proposed direct solver

Example	circuit size	problem size	ratio
LNA + mixer	1135	69235	3.5
receiver 1	2158	66898	7.0

framework of multi-rate signals. When circuits are driven by or respond with signals of widely separated time scales, it is advantageous to view the circuit equations in more than one time scales [14, 15]. The resulting partial differential equation can be efficiently solved and the circuit response can be easily derived from the multi-rate solution. The multi-rate PDE version of (1) is given by

$$\frac{\partial q(\hat{x}(t_1, t_2))}{\partial t_1} + \frac{\partial q(\hat{x}(t_1, t_2))}{\partial t_2} + f(\hat{x}(t_1, t_2)) + \hat{b}(t_1, t_2) = 0$$

where the convolution term has been omitted. If $x(t_1, t_2)$ is the solution of the above equation then it can be shown that $x(t, t)$ is the solution of the original DAE [15]. Fourier envelope assumes that the signals are periodic in t_2 and therefore can be expanded in Fourier series as before. After following essentially the same steps outlined above we get

$$\frac{\partial Q(t_1)}{\partial t_1} + 2\pi j f_2 \Omega Q(t_1) + \mathcal{F}(t_1) + \mathcal{B}(t_1) = 0$$

The above DAE is discretized using an appropriate linear multi-step method (such as Backward Euler) and integrated in t_1

$$\frac{Q_i - Q_{i-1}}{h_i} + 2\pi j f_2 \Omega Q(t_{1_i}) + \mathcal{F}(t_{1_i}) + \mathcal{B}(t_{1_i}) = 0$$

The Jacobian for this is of the form

$$\frac{1}{h_i} C_i + \mathcal{G}_i + 2\pi j f_2 \Omega C_i$$

Note that the Fourier envelope Jacobian is almost identical to HB Jacobian except for an additional term $\frac{1}{h_i} C_i$. Therefore direct solve can be used for Fourier envelope as well.

In fact, direct solve is ideally suited for this analysis because it is typically performed with one large periodic tone (usually the LO in an RF transceiver), the nonlinearities are strong and at every envelope step, it usually takes 2-3 Newton iterations to converge.

Since direct solve lends itself very naturally to Jacobian bypass, we can take advantage of this. The Jacobian matrix is not factored (bypassed) and the previous factors are used to perform triangle solves with L and U. As the number of bypassed Jacobians increases, the total number of Newton iterations increases but the time taken per Newton iteration reduces because of time savings due to Jacobian bypass. Jacobian bypass can result in 2-3× performance improvement over non-bypassed implementation of the direct solver. Table 3 compares the CPU time and memory requirement for various examples. Many of these examples are also present in Table 1. Note that even for those

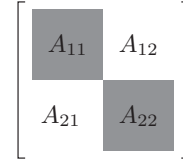


Figure 3: Splitting HB Jacobian into two segments

examples where 1- or 3-tone HB with direct solver was slower than the best preconditioned Krylov subspace method, envelope analysis is faster with the direct solver. This is entirely due to the use of Jacobian bypass with the direct solver which does not give any speed improvement in the case of Krylov subspace methods.

6. PRECONDITIONERS BASED ON DIRECT SOLVE

Li and Pillage [10, 4] introduced a new preconditioner of HB Jacobian. They segmented the HB matrix into two parts of approximately equal number of harmonics, as shown in Figure 3, solved the two portions independently, and used the combined solution as the result of a preconditioner solve. By recursively applying this method on each sub-problem they developed efficient methods for solving the hierarchical problem. Our proposed preconditioner is inspired by their approach with two crucial modifications

1. we use the direct solver for solving each of the segments
2. we also include A_{21} in the preconditioner, i.e., for two segments we are including 75% of the matrix in the preconditioner.

I.e., if we need to solve for $[b_1, b_2]^T$ using this preconditioner, we first solve $A_{11}x_1 = b_1$, and then solve $A_{22}x_2 = b_2 - A_{21}x_1$. This requires an extra matrix vector multiply with J_{hb} . $A_{21}x_1$ can be obtained by multiplying J_{hb} with $[x_1, 0]^T$ and taking the second block vector of the product. To generalize this, if the matrix is divided into l segments then for the i^{th} block row, we need to form $\sum_{j=1}^{i-1} A_{ij}x_j$, which can be formed by multiplying J_{hb} with $[x_1, \dots, x_{i-1}, 0, \dots]^T$ and taking the result of the i th block row. Note that the inclusion or non-inclusion of the A_{ij} blocks applies only to the preconditioner matrix, and the accuracy of the solution of HB Jacobian is *unaffected* in both the cases.

The advantage of using direct solve of A_{11} and A_{22} is that these matrices need to be factored only *once* and in each subsequent Krylov subspace iteration, we only need to perform *tri-angle solves* with L and U factors of these matrices which gives significant speed advantage over the preconditioner where A_{11} and A_{22} are also solved using Krylov subspace methods.

6.1 Complexity analysis

Assuming that the matrix is divided into l segments of approximately equal number of harmonics, the storage requirement of each segment is reduced by the factor l^2 but we need

Table 4: Memory and CPU time for HB Jacobian using various values of segments

Example	direct		Preconditioned by direct					
	time(s)	mem(GB)	segment=2			segment=3		
			time(s)	mem(GB)	iterations	time(s)	mem(GB)	iterations
LNA + mixer	990	6.5	1161	3.7	32	1325	2.8	67
receiver 1	3297	12.7	1905	7.0	15	2202	5.2	26
LNA + mixer + filter	3361	20	8052	15.7	50	12450	12.6	80

Table 5: Memory and CPU time for HB Jacobian using various values of segments without using A_{21}

Example	direct		Preconditioned by direct					
	time(s)	mem(GB)	segment=2			segment=3		
			time(s)	mem(GB)	iterations	time(s)	mem(GB)	iterations
LNA + mixer	990	6.5	1487	3.8	61	1479	2.8	94
receiver 1	3297	12.7	2328	7.1	28	2340	5.3	44
LNA + mixer + filter	3361	20	9513	16.0	99	10551	12.6	151

to store l of these. Hence the overall memory requirement reduces by a factor of l . Similarly the matrix factor time is reduced by l^2 but we need to perform l triangle solves at each Krylov iteration. If m iterations are required for Krylov subspace method to converge to the solution, then the total solve time is $\mathcal{O}\left(ml(nnz u_u^\alpha + nnz l_l^\alpha) \frac{(2k+1)^2}{l}\right)$. We need $m-1$ matrix vector multiplies with the HB Jacobian. Therefore the overall cost of this is

$$\begin{aligned} & \mathcal{O}\left((nnz u_u^\alpha + nnz l_l^\alpha) \frac{(2k+1)^3}{l^2}\right) \\ & + \mathcal{O}\left(m(nnz u_u^\alpha + nnz l_l^\alpha) \frac{(2k+1)^2}{l}\right) \\ & + \mathcal{O}(ml(nnzc + nnzu)(2k+1) \log(2k+1)) + \mathcal{O}(m^2 n(2k+1)) \end{aligned}$$

This provides a direct trade-off between memory and speed. As l is increased, the memory consumption goes down but m increases which may increase the HB solve time depending how steep the increase in m is.

6.2 Results

Table 4 shows the memory and CPU time for various segment values for 3-tone HB examples from Table 1. For comparison, the same examples are run for various segment values but without including A_{21} (Table 5). This saves the matrix vector multiply with J_{hb} but increases the number of Krylov subspace iterations. It is clear that whether CPU time increases or decreases with increasing number of segments is heavily dependent on the problem.

7. CONCLUSION

In this paper we introduced a new method of performing direct solve of the harmonic balance Jacobian. The proposed block LU decomposition is novel as it takes advantage of the unique structure of the HB Jacobian which has the same block-sparsity pattern as the Jacobian formed in transient analysis. The resulting matrix and LU data-structures are more compact than is possible with a general purpose sparse matrix solver and allow handling much larger problems. This solver also results in far superior performance as the operations on the dense blocks are performed efficiently using LAPACK/BLAS routines.

For examples with moderate number of harmonics and moderate to strong nonlinearities, we demonstrated that the proposed direct solver is significantly faster than the best preconditioned iterative solvers with a moderate increase in the memory. We also showed that this solver is especially suited for Fourier envelope analysis where the number of harmonics is small, circuits are nonlinear and the direct solver is very well suited for using Jacobian bypass. For examples with large number of harmonics and moderate to strong nonlinearities, the performance advantage of the new solver is maintained but the memory requirements increase. To address this we proposed preconditioners

based on direct solution of harmonic balance matrices which provide the user with a memory-speed trade-off. Above all, the proposed method robustly solves problems with strong nonlinearities, which were often found to be beyond the reach of traditional iterative methods with comparable computing resources.

8. REFERENCES

- [1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide, 3rd ed.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999.
- [2] ATLAS. <http://math-atlas.sourceforge.net/>.
- [3] L. S. Blackford, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, M. Heroux, L. Kaufman, A. Lumsdaine, A. Petitot, R. Pozo, K. Remington, and R. C. Whaley. An updated set of Basic Linear Algebra Subprograms (BLAS). *ACM Transactions on Mathematical Software*, 28:135–151, 2002.
- [4] W. Dong and P. Li. Hierarchical harmonic-balance methods for frequency-domain analog-circuit analysis. *IEEE Trans. Computer-Aided Design*, 26:2089–2101, Dec. 2007.
- [5] P. Feldmann, B. Melville, and D. Long. Efficient frequency domain analysis of large nonlinear analog circuits. In *Proceedings of the IEEE 1996 Custom Integrated Circuits Conference*, pages 461–464, 1996.
- [6] R. Freund, G. H. Golub, and N. M. Nachtigal. Iterative solutions of linear systems. *Acta Numerica*, pages 57–100, 1991.
- [7] R. J. Gilmore and M. B. Steer. Nonlinear circuit analysis using the method of harmonic balance, a review of the art. Part I. Introductory concepts. *International Journal on Microwave and Millimeter Wave Computer Aided Engineering*, 1, Jan. 1991.
- [8] Intel. <http://www.intel.com/cd/software/products/asm-na/eng/307757.htm>.
- [9] K. S. Kundert, J. K. White, and A. L. Sangiovanni-Vincentelli. *Steadystate methods for simulating analog and microwave circuits*. Kluwer, Boston, MA, 1990.
- [10] P. Li and L. T. Pillegi. Efficient harmonic balance simulation using multi-level frequency decomposition. In *IEEE/ACM International Conference on Computer Aided Design*, pages 677–682, 2004.
- [11] D. Long, R. Melville, K. Ashby, and B. Horton. Full-chip harmonic balance. In *Proceedings of the IEEE 1997 Custom Integrated Circuits Conference*, pages 379–382, 1997.
- [12] R. C. Melville, P. Feldmann, and J. Roychowdhury. Efficient multi-tone distortion analysis of analog integrated circuits. In *Proceedings of the IEEE 1995 Custom Integrated Circuits Conference*, pages 241–244, 1995.
- [13] O. Nastov, R. Telichevesky, K. Kundert, and J. White. Fundamentals of fast simulation algorithms for RF circuits. *Proceedings of the IEEE*, 93:600–621, Mar. 2007.
- [14] J. Roychowdhury. Efficient methods for simulating highly nonlinear multi-rate circuits. In *Proceedings 34th Design Automation Conference*, pages 269–274, 1997.
- [15] J. Roychowdhury. Analyzing circuits with widely separated time scales using numerical PDE methods. *IEEE Trans. Circuits Sys. I*, 48:578–594, May 2001.
- [16] Y. Saad. *Iterative methods for sparse linear systems*. Society for Industrial and Applied Mathematics, 1996.
- [17] R. Telichevesky, K. Kundert, I. Elfadel, and J. K. White. Fast simulation algorithms for RF circuits. In *Proceedings of the IEEE 1996 Custom Integrated Circuits Conference*, pages 437–444, 1996.
- [18] V. Rizzoli, C. Cecchetti, A. Lipparini, and F. Matri. General-purpose harmonic balance analysis of nonlinear microwave circuits under multitone excitation. *IEEE Trans. Microwave Theory Tech*, 1, Jan. 1991.