View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by HKU Scholars Hub

The HKU Scholars Hub The University of Hong Kong 香港大學學術庫

Title	Globally stable, highly parallelizable fast transient circuit simulation via faber series
Author(s)	Li, YC; Chen, Q; Weng, SH; Cheng, CK; Wong, N
Citation	The IEEE 10th International New Circuits and Systems Conference (NEWCAS 2012), Montreal, QC., 17-20 June 2012. In Conference Proceedings, 2012, p. 177-180
Issued Date	2012
URL	http://hdl.handle.net/10722/174256
Rights	Annual IEEE Northeast Workshop on Circuits and Systems (NEWCAS) Proceedings. Copyright © IEEE.

Globally Stable, Highly Parallelizable Fast Transient Circuit Simulation via Faber Series

Ying-Chi Li[1], Quan Chen[1], S. H. Weng[2], C. K. Cheng[2] and Ngai Wong[1]
[1]Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong
Email: tycli@eee.hku.hk, quanchen@eee.hku.hk, nwong@eee.hku.hk
[2]Department of Computer Science and Engineering, The University of California, San Diego, USA
Email: s2weng@eng.ucsd.edu, ckcheng@ucsd.edu

***Abstract*—Time-domain circuit simulation based on matrix exponential has attracted renewed interested, owing to its explicit nature and global stability that enable millionth-order circuit simulation. The matrix exponential is commonly computed by Krylov subspace methods, which become inefficient when the circuit is stiff, namely, when the time constants of the circuit differ by several orders. In this paper, we utilize the truncated Faber Series for accurate evaluation of the matrix exponential even under a highly stiff system matrix arising from practical circuits. Experiments have shown that the proposed approach is globally stable, highly accurate and parallelizable, and avoids excessive memory storage demanded by Krylov subspace methods.**

I. Introduction

The magnitudes of parasitic components in modern VLSI circuits can differ by more than several orders, which making the resulting ordinary differential equation (ODE) system usually stiff. Efficient numerical simulation of stiff systems has long been a challenge. Excessively small time steps are required to maintain the stability of solution when explicit methods (e.g. forward Euler) are used. Implicit methods like trapezoidal and Gear methods have been widely used to attack the stiff issue, but the requirement of solving linear systems limits their scalability. Recently, a third category of integration method featuring global stability and explicit nature has been developed [1]. The new matrix exponential method (MEXP) exploits the analytical solution of ODE in the matrix exponential form, and uses the Krylov subspace method to compute the product $e^A v$ efficiently [1]. The Krylov-based MEXP has been proved to be absolute-stable and involves only sparse matrix-vector products, rendering it a highly parallelizable technique.

Despite overcoming the stability limitation, the Krylov-based MEXP still faces certain difficulties when dealing with stiff circuits. The approximation of matrix exponential for non-Hermitian matrices, which is usually the case for modified nodal analysis (MNA) representation, involves the storing all the basis vectors of the Krylov subspace. Memory constraint is often an issue especially when the system matrix A is stiff, calling for a large number of basis vectors. The high memory requirement also limits the implementation of Krylov-based methods on state-of-the-art parallel computing architectures such as Graphics Processing Uint (GPU) and Field Programmable Gate Array (FPGA), with scarce local memory resource. A common remedy to alleviate the memory bottleneck is by restarting the algorithm each time the Krylov subspace has reached a maximum dimension. Nevertheless, the restarted Krylov methods suffer from a degraded convergence and even diverge under some conditions.

Compared with the Krylov subspace method, computing matrix exponential via polynomial approximation has been less explored. Chebyshev series expansion can be used to compute matrix exponential for large and symmetric matrices. The performance of the Krylov subspace method and the Chebyhev series expansion is compared in [6], showing that the latter has significant memory saving and can outperform the former when handling stiff systems. However, the Chebyshev method is not available for asymmetric matrices. The Faber polynomial and series were first proposed by Georg Faber in 1903 [12]. Curtiss has summarized the development of the Faber polynomial and series in [11]. The approximation of the matrix exponential with non-stiff matrix based on the Faber series has been studied [4]. In special cases, The Faber polynomial can be reduced to the Chebyshev polynomial [8] and the Taylor Series [4] for matrix exponential approximation.

In this paper, we extend the Faber series machinery to practical situations with stiff system matrices. The results are contrasted with the Krylov subspace method. Advantages of Faber series method include: 1) Only a smaller number of vectors need to be stored because of a recurrence relation; 2) It guarantees convergence if the spectrum of A is included in a bounded set Ω where e^{At} is analytic everywhere; 3) The computational cost for each iteration is constant [5]; 4) Each recursion step involves only one matrix-vector multiplication which is highly parallelizable to further decrease the computation time. The parallelization is realized on a GPU with CUDA architecture. In the following, the theoretical background of Faber series is reviewed in Section II. The application of the Faber series in approximating matrix exponential will be detailed in Section III, followed by numerical experiments in Section IV. Section V then draws the conclusion.

II. Faber polynomial and Faber series

Let $\Omega \subset \mathbb{C}$ be a compact set containing more than one point and bounded by a Jordan curve Γ_Ω. $\overline{\mathbb{C}}\backslash\Omega$ denotes the complement of Ω which is simply connected in the extended complex plane. Then, by the Riemann mapping theorem, a

978-1-4673-0859-5/12/$31.00 ©2012 IEEE

unique function which has the Laurent expansion

$$z = \phi(\omega) = c\omega + c_0 + \frac{c_1}{\omega} + \frac{c_2}{\omega^2} + ... \tag{1}$$

conformally maps the exterior of a unit disk in the ω-plane, i.e. $\{\omega : |\omega| > 1\}$ onto $\overline{\mathbb{C}}\backslash\Omega$ in the z-plane, where c is the logarithmic capacity of Ω, $\phi(\infty) = \infty$ and $\phi'(\infty) = c$.

There exists an inverse function $\psi(z) = \phi^{-1}(z)$ from z exterior to a sufficiently large circle and has a Laurent expansion

$$\omega = \psi(z) = dz + d_0 + \frac{d_1}{z} + \frac{d_2}{z^2} + ... \tag{2}$$

which maps $\overline{\mathbb{C}}\backslash\Omega$ back onto $\{\omega : |\omega| > 1\}$ and $d = 1/c$. Then the n-th Faber polynomial $F_n(z)$, for $n = 1, 2, ...$ with respect to Ω is defined as the principle part of the Laurent expansion at ∞ of $[\psi(z)]^n$ with $F_0(z) = 1$ [11]. The generating function of the Faber polynomial is then (cf.[11])

$$\frac{\omega\phi'(\omega)}{\phi(\omega) - z} = 1 + F_1(z)\frac{1}{\omega} + F_2(z)\frac{1}{\omega^2} + ..., \tag{3}$$

where $|\omega| > 1$ and $z \in \overline{\mathbb{C}}\backslash\Omega$. By multiplying $\phi(\omega) - z$ on both sides of (3) and expanding $\phi(\omega)$ and $\phi'(\omega)$ in their Laurent series, a recursion formula for Faber polynomial $F_n(z)$ is easily deduced,

$$\begin{gathered} F_0(z) = 1,\ F_1(z) = \frac{z - c_0}{c}, \\ F_{n+1}(z) = \frac{(z - c_0)F_n - [c_1F_{n-1} + ... + c_nF_0] - nc_n}{c}, \end{gathered} \tag{4}$$

for $n = 1, 2, ...$ (cf. [8]). where $c, c_0, c_1, ...$ are the coefficients of (1). Given any function $f(z)$ which is analytic everywhere inside Ω, by the exterior mapping function (1) with $\mathbf{C_R} = \{z : z = \phi(\omega), |\omega| > 1\}$ and the Cauchy Integral Formula, for any $z_0 \in \Omega$, we obtain

$$\begin{aligned} f(z_0) &= \frac{1}{2\pi i}\int_{\mathbf{C_R}} \frac{f(z)}{z - z_0}dz \\ &= \frac{1}{2\pi i}\int_{|\omega|=1} \frac{f(\phi(\omega))\phi'(\omega)}{\phi(\omega) - z_0}d\omega. \end{aligned} \tag{5}$$

Dividing ω in (3) and substituting into (5), the Faber series expansion reads

$$\begin{aligned} f(z_0) &= \frac{1}{2\pi i}\int_{|\omega|=1} f(\phi(\omega))\left[\sum_{n=0}^{\infty}\frac{F_n(z)}{\omega^{n+1}}\right]d\omega \\ &= \sum_{n=0}^{\infty} a_nF_n(z), \end{aligned} \tag{6}$$

where

$$a_n = \frac{1}{2\pi i}\int_{|\omega|=1} \frac{f(\phi(\omega))}{\omega^{n+1}}d\omega \tag{7}$$

is the Faber coefficient of $f(z_0)$ and $F_n(z)$ is the Faber polynomial for $n = 1, 2, ...$

III. Matrix exponential in the Faber series

In MNA, a circuit is represented by a system of ODEs (here we only consider linear circuits)

$$C\dot{x}(t) = Gx(t) + Bu(t), \tag{8}$$

where C is the capacitance/inductance matrix, G denotes the conductance matrix and $u(t)$ the input. The analytic solution of (8) is given by

$$x(t) = e^{At}x(0) + \int_0^t e^{A(t-\tau)}b(\tau)d\tau, \tag{9}$$

where $A = C^{-1}G$ (explicit computation for C^{-1} is not needed [1]), and $b(t) = C^{-1}Bu(t)$. Although C is possibly singular, previous work [2] has proposed a pragmatic means to regularize C with affordable cost and sparsity preservation. The computation of (9) boils down to evaluating the action of the matrix exponential on a vector [1]

$$y = f(At)\mathbf{v} = e^{At}\mathbf{v}, \tag{10}$$

where $A \in \mathbb{C}^{N\times N}$ which is stable (Re$(\sigma(A)) < 0$,where $\sigma(A)$ denotes the spectrum of A) and asymmetric, t is the time step and $\mathbf{v} \in \mathbb{C}^N$. For $\sigma(A) \subseteq \Omega$. If the conformal mapping from $\{\omega : |\omega| > 1\}$ onto $\overline{\mathbb{C}}\backslash\Omega$ is (1), then for $\sigma(At) \subseteq \Omega_t$, the conformal mapping from $\{\omega : |\omega| > 1\}$ onto $\overline{\mathbb{C}}\backslash\Omega_t$ is

$$\phi_t(\omega) = \phi(\omega)t = ct\omega + c_0t + \frac{c_1t}{\omega} + \frac{c_2t}{\omega^2} + ... \tag{11}$$

By (5) and (6), the Faber series expansion of this matrix exponential is

$$\begin{aligned} f(At)\mathbf{v} &= \frac{1}{2\pi i}\int_{\mathbf{C_R}} f(z)(zI - At)^{-1}\mathbf{v}dz \\ &= \sum_{n=0}^{\infty} a_nF_n(At)\mathbf{v}, \end{aligned} \tag{12}$$

where I is the identity matrix. By (7) and substituting $\omega = e^{i\theta}$, a_n is changed to a closed-loop integral:

$$\begin{aligned} a_n &= \frac{1}{2\pi i}\int_{|\omega|=1} \frac{f(\phi_t(\omega))}{\omega^{n+1}}d\omega \\ &= \frac{1}{2\pi}\int_0^{2\pi} [e^{ce^{i\theta} + c_0 + c_1e^{-i\theta} + c_2e^{-2i\theta)} + ...}]^t e^{-ni\theta}d\theta, \end{aligned} \tag{13}$$

and $F_n(At)$ is obtained by substituting At into (4). Since t in $F_n(At)$ cancels out, $F_n(At) = F_n(A)$. The scaling factor t only changes the value of the Faber coefficient a_n and has no effect on the Faber polynomial F_n.

Instead of computing F_n from (4), we compute p_n which has the form of

$$\begin{gathered} p_0 = \mathbf{v},\ p_1 = \frac{A - c_0I}{c}\mathbf{v}, \\ p_{n+1} = \frac{(A - c_0I)p_n - [c_1p_{n-1} + ... + c_np_0] - nc_n\mathbf{v}}{c}. \end{gathered} \tag{14}$$

Equation (12) becomes:

$$f(At)\mathbf{v} = \sum_{n=0}^{\infty} a_np_n. \tag{15}$$

Thus, in each recursion iteration, only one matrix-vector multiplication (the most costly step when A is large) is needed and the rest is just vector summation. Note that $A\mathbf{v}$ is computed by $C^{-1}(G\mathbf{v})$ which involves only the solution of a linear sparse system. With this structure, each recursion is highly parallelizable in terms of the matrix-vector multiplication and vector-vector summation, whereas only parallelization of matrix-vector multiplication can be done in the Krylov method. In our implementation, GPU computation is used to speed up the iteration process.

It is obvious that the coefficients in $z = \phi(\omega)$ are important for computing both the Faber coefficients and the Faber polynomial. And they depend on the distribution of $\sigma(A)$. Since A is large, it is prohibitive to compute all the eigenvalues of A. However, if Ω is a polygon and its vertices are known, the coefficients of $\phi(\omega)$ can be calculated conveniently. To estimate the vertices of Ω, we adopt the Arnoldi-Faber iterative method in [8] to estimate $\sigma(A)$ and compute the coefficients of $z = \phi(\omega)$ by the methods proposed in [8] and [10].

In practice, we truncate the Faber series expansion up to m terms to approximate (15)

$$y_m = \sum_{n=0}^{m-1} a_n F_n(A)\mathbf{v} = \sum_{n=0}^{m-1} a_n p_n \approx f(A)\mathbf{v}. \tag{16}$$

Thus, a reliable error estimate is needed to terminate the expansion. We adapt the method in [5]. In addition, by experiment, when a large time step is applied (e.g. $\max(\sigma(At)) > 10^3$), the estimated error is out of the acceptable range. Scaling technique (modified from the 2^N algorithm) is applied to overcome this undesirable situation, namely, we scale the exponential by a factor s as follows

$$f(At) = e^{At}\mathbf{v} = (e^{\frac{At}{s}})^s\mathbf{v}. \tag{17}$$

Then, the truncated Faber series approximation by (16) is

$$y_m = \left[\sum_{n=0}^{m-1} a_n F_n(\frac{At}{s})\right]^s \mathbf{v}, \tag{18}$$

and the error is given as

$$\text{err}(At) = \text{err}\left(\frac{At}{s}\right) s, \tag{19}$$

which lies in a acceptable range of err $< 10^{-5}$.

IV. Numerical Examples

Two examples are tested with the Faber series expansion method. The results are compared with the Krylov subspace method to evaluate the performance. We take $\mathbf{v} \in \mathbb{C}^N$ to be an all-one vector. The algorithm is implemented in Matlab. Both methods are also implemented on GPU (NVIDIA GeForce GTX 570 with 1.25GB RAM) with Jacket v2.0 [13] to demonstrate the speedup from parallelization.

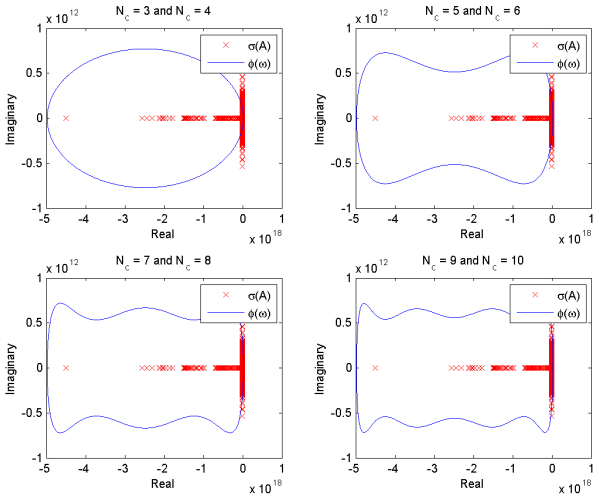

Fig. 1. Relation between N_c and the shape of Ω.

A. Example 1

The first matrix is of size 972×972. Fig.1 shows the spectrum of A, $\min(\text{Re}(\sigma(A))) = -4.506 \times 10^{18}$ while $\max(\text{Re}(\sigma(A))) = -2.0309 \times 10^5$. The upper and lower bounds of the imaginary axis are 5.3191×10^{11} and -5.3191×10^{11}, respectively. This is an extremely stiff system with the maximum and minimum magnitudes of eigenvalues differing by 10^{13}. Given the extreme points of the spectrum, the compact set Ω can be constructed by the method in [8]. For simplicity, rectangle $[-4.506 \times 10^{18}, 0] \times [-6 \times 10^{11}i, 6 \times 10^{11}i]$ is used to enclose $\sigma(A)$. The number of coefficients N_c of $\phi(\omega)$ will affect the bounded region of the eigenvalue. Fig. 1 also shows that the more coefficients are used, the tighter Ω bounds the spectrum. In addition, the smaller is the logarithmic capacity c (first coefficient in $\phi(\omega)$), the faster is the convergence. The Faber coefficient a_n in (13) is computed by the trapezoidal numerical integration $trapz$ in Matlab. Since the computation involves only scalars, the time for computing a_n is negligible compare with the matrix-vector multiplication when A is large.

Table I shows the results of the Faber series approximation of $y = e^{At}\mathbf{v}$ with $N_c = 5$, which are compared with the Krylov subspace method in terms of the number of matrix-vector products (mvps) and the total runtime for the same error estimation. In general, the Faber series approach requires more mvps than the Krylov method, which is expected since the Arnoldi process is nearly optimal in terms of error reduction. The number of iterations also increase for the Faber approximation when stiffness grows. Nevertheless, the total runtime of the Faber series method is higher than the Krylov method by only $1.4\times$ with a nearly doubled number of mvps. This suggests the performance of the two methods may not solely depend on the number of mvps, as will be evident in the next example.

B. Example 2

The second example is a large sparse matrix of 479201×479201. The extreme eigenvalues are $\min(\text{Re}(\sigma(A))) =$

TABLE I
RESULT OF y_m: EXAMPLE 1

	Faber series		Krylov-Arnoldi		
t	#itr m	CPU(ms)	#itr m	CPU(ms)	err
1E-18	10	16.854	8	15.594	10^{-8}
5E-18	18	20.851	12	18.224	10^{-7}
1E-17	24	23.803	15	20.576	10^{-6}
5E-17	51	37.965	32	27.419	10^{-6}
1E-16	70	46.800	45	34.358	10^{-6}
5E-16	148	86.452	90	63.201	10^{-6}
1E-15	207	116.372	110	81.842	10^{-5}

-9.2272×10^{12} while max(Re($\sigma(A)$)) = -2.4673×10^{8}. A rectangle set $\Omega = [-1 \times 10^{13}, 0] \times [-5.8 \times 10^{4} i, 5.8 \times 10^{4} i]$ is applied. Tables II and III show respectively the performance of the Faber series with $N_c = 5$ and the Krylov method for different time steps (i.e., different degrees of resulted stiffness). Both tables use an estimated error of 10^{-6}. sf denotes the scaling factor used in the Faber series method. According

TABLE II
RESULT OF y_m BY THE FABER SERIES: EXAMPLE 2

t	#itr m	sf	CPU-T (ms)	GPU-T (ms)	Speedup
1E-13	6	1	92.46	54.89	1.68X
5E-13	11	1	181.85	65.19	2.79X
1E-12	14	1	221.50	71.83	3.08X
5E-12	30	1	475.68	108.98	4.36X
1E-11	41	1	645.07	128.59	5.02X
5E-11	104	1	1678.76	261.591	6.41X
1E-10	213	2	3387.18	502.723	6.74X
5E-10	957	8	15152.53	2079.66	7.29X

TABLE III
RESULT OF y_m BY THE KRYLOV-ARNOLDI METHOD: EXAMPLE 2

t	#itr m	CPU-T(ms)			GPU-T (ms)	Speedup
		mvps	orth	total		
1E-13	5	30.22	40.34	155.29	157.04	0.989X
5E-13	6	36.04	56.15	186.71	185.99	1.004X
1E-12	6	36.18	56.03	187.11	186.83	1.001X
5E-12	8	48.75	95.265	256.33	254.47	1.007X
1E-11	11	66.38	175.66	400.82	376.99	1.063X
5E-11	20	120.01	556.08	931.46	891.858	1.044X
1E-10	30	179.73	1250.83	1839.85	1752.9	1.050X
5E-10	96	589.35	12428.00	14224.07	13847.81	1.027X

to the result, the number of iterations needed for the Faber series are still larger than the Krylov method, up to 10 times for the stiffest case. However, the total computation times of the two methods do not scale in the same manner. The Faber series method runs comparably fast with the Krylov method on CPU for the last case. The reason of this mismatch lies in the cost of orthogonalization (orth) process in the Krylov subspace construction, whose complexity grows quadratically with the number of iterations. When the matrix is very sparse and mvp is relatively inexpensive, the orthogonalization tends to dominate the total computation. The increasing portion of time that orthogonalization takes is evident in Table III, which reaches 87% in the last case. In addition, when a large number of basis vectors is required for stiffer problems, the Krylov method actually has to be restarted, for instance every 20 iterations, due to the memory constraint for ordinary machines. This will further downgrade the performance of Krylov method. On the other hand, the Faber series method only needs to store a small, constant number of vectors in memory (4 vectors for $N_c = 5$), and is therefore particularly suitable for memory-limited scenarios.

The advantage of Faber series method is more prominent for GPU implementation. The major computation in the Faber approximation involves only mvps and vector summation, which can benefit most from parallelization. The GPU implementation therefore can bring a significant speedup (up to 7 times in the test) compared with the CPU implementation. Since the orthogonalization is less straightforward for parallelization, the improvement from GPU implementation is relatively small for the Krylov method. The Faber series approach is about 3 to 6 times faster than the Krylov method on the GPU platform.

V. CONCLUSION

We have developed an explicit, globally stable time-domain circuit simulation method based on the matrix exponential formulation. The Faber series is employed to compute the matrix-exponential-vector product efficiently. Numerical result have shown that the Faber series method is advantageous in terms of memory usage and ease of parallelization, and can be faster than the Krylov-based methods in practice even using more iterations. The Faber series method can be a valuable complement with the Krylov subspace method when dealing with large and stiff circuit simulation problems.

REFERENCES

[1] S. H. Weng, Q. Chen, and C. K. Cheng, "Time-domain analysis of large-scale circuits by matrix exponential method with adaptive control," IEEE Trans. on CAD, 2012, in press.

[2] Q. Chen, S. H. Weng, and C. K. Cheng, "A practical regularization technique for modified nodal analysis in large-scale time-domain circuit simulation," IEEE Trans. on CAD, 2012, in press.

[3] C. Moler and C. V. Loan, "Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later," *SIAM Review*, vol. 45, no. 1, pp. 3-C49, Mar 2003.

[4] I. Moret and P. Novati, "An interpolatory approximation of the matrix exponential based on Faber polynomials," *Journal of Computational and Applied Mathematics*,vol. 131 no. 1-2, 2001.

[5] I. Moret and P. Novati, "The computation of functions of matrices by truncated Faber series", *Numerical Functional Analysis and Optimization*, vol. 22, no. 5-6, pp. 697-719, 2001.

[6] L. Bergamaschi and M. Vianello, "Efficient computation of the exponential operator for large, sparse, symmetric matrices," *Numerical Linear Algebra with Applications*, vol. 7, no. 1, pp. 27-45, 2000.

[7] M. Hochbruck and C. Lubich, "On Krylov subspace approximations to the matrix exponential operator," *SIAM Journal on Numerical Analysis*, vol. 34, no. 5, pp. 19111925, 1997.

[8] G. Starke, R. S. Varga, "A hybrid Arnoldi-Faber iterative method for nonsymmetric systems of linear equations," *Numerische Mathematik*, vol. 64, no. 1, pp. 213-240, 1993.

[9] Y. Saad, "Analysis of some Krylov subspace approximations to the matrix exponential operator,"*SIAM Journal on Numerical Analysis*, 1992.

[10] L. Trefethen, "Numerical computation of the Schwarz-Christoffel transformation," *SIAM J. Sci. Stat. Compt.*, vol. 1, pp. 82-102, 1980.

[11] J. H. Curtiss, "Faber polynomials and the Faber series," *The American Mathematical Monthly*, vol. 78, no. 6, pp. 577-596, 1971.

[12] G. Faber, "*Über* polynomische entwicklungen," *Math. Ann.*, 57, pp398-408, 1903.

[13] Accelereyes. [Online]. Available: http://www.accelereyes.com/