Dynamic Analysis of Power Delivery Network with Nonlinear Components Using Matrix Exponential Method

Hao Zhuang, Ilgweon Kang, Xinan Wang, Jeng-Hau Lin, and Chung-Kuan Cheng
Department of Computer Science & Engineering, University of California, San Diego, CA, 92093
{zhuangh, i2kang, xinan, jel252, ckcheng}@ucsd.edu

Abstract—In this work, we propose a matrix exponential-based time-integration algorithm for dynamic analysis of power delivery network (PDN) with nonlinear components. The presented method is an explicit method and maintains very competitive aspects compared to traditional low order approximation methods, such as backward Euler method with Newton-Raphson iterations (BENR). The proposed method takes comparable number of time steps to complete the whole simulation. Second, the method takes only one LU decomposition per time step while BENR requires at least two LU decompositions for the convergence check of solutions of nonlinear system. Moreover, our method does not need to repeat expensive LU decomposition operations when the length of time steps are adjusted for error controls. The experimental results validate our method's efficiency. We observe the reductions of total LU operation number and the simulation runtime.

Categories and Subject Descriptors: B.7.2 [Integrated Circuits]: Design Aids—simulation; G.1.3 [Numerical Analysis]: Numerical Linear Algebra; J.6 [Computer-Aided Engineering]

General Terms: Algorithms, Design, Theory

Keywords: Power delivery network, matrix exponential method, circuit simulation, Krylov subspace.

I. INTRODUCTION

VLSI design verification relies on the dynamic analysis of power delivery network (PDN) to estimate power supply noises. Currently, PDN is often modeled as a large-scale linear circuit system with voltage supplies and time-varying current sources [1], [2]. However, modern designs also contain nonlinear components for their PDNs, such as power gated transistors, or voltage regulators [3]. To ensure power and timing integrity of the systems, it is crucial to analyze large dimensional interconnect network together with nonlinear devices [4]. Therefore, previous formulations and solvers [1], [5]–[7] may not be sufficient for the purpose. It is important to design efficient yet accurate nonlinear PDN time-domain simulation algorithms.

Conventional time-domain simulation has two types of formulation, explicit and implicit methods, for solving differential equations numerically. The explicit formulation has very small stable region so that it constrains the time step choice [8]. It may require special treatment to obtain the unconditional stability [9]. Otherwise, the time step would be extremely small in the stiff circuit system, and then results to very bad runtime performance. In contrast, implicit methods, such as backward Euler (BE) and trapezoidal methods [1], [5]–[7], are regarded as stable approaches, and adopted to solve PDN time-domain simulation in academic research and industry. Nonlinear system is linearized and solved iteratively, such as Newton-Raphson method (BENR), until the solutions are converged. To solve resulting linear system, direct solvers (LU, Cholesky matrix decompositions) [10], [11] are favored due to its robustness. Therefore, for each time step, it requires at least two LU decompositions until the solutions are converged. Besides, in PDN simulation, long time span/large number of time steps always accompanies with such time-consuming LU operations, making the simulation very expensive. Therefore, it is crucial to devise efficient PDN simulation algorithms with less number of expensive LU operations.

Beyond conventional method, matrix exponential time-integration kernel has been applied for time-domain circuit simulation [12]–[15]. This kernel can be used to derive analytical solution for linear differential equation system. Combined with Krylov subspace for matrix exponential and vector product (MEVP) computation, this method provides high order approximation for solving differential equations than conventional methods, and very flexible time-step controlling due to its explicit nature and scaling invariant property of Krylov subspace. The drawbacks of previous works is the slow convergence rate for matrix exponential and vector product (MEVP) by standard Krylov subspace [12]. The works [14], [15] use variant Krylov subspace schemes to accelerate MEVP's convergence rate. Another problem is previous matrix exponential based simulation still requires iterations due to its formulation [12], [13], [16], which includes expensive LU operations, to converge the solutions.

In this work, we design exponential Rosenbrock-Euler method based on [17], [18] to leverage matrix exponential time-integration kernel for nonlinear PDN simulation. MEVP is computed via inverted Krylov subspace [15]. The contributions of this paper are as follows.

- The proposed method is more suitable for nonlinear PDN dynamic analysis, which takes only one LU decomposition per time step, while conventional Newton-Raphson backward Euler method requires at least two operations.
- The method does not need to repeat LU decomposition when we adjust the length of time steps for error controls. It is due to the matrix exponential formulation and Krylov subspace (time-step) scaling-invariant property [19]. On the contrary, the low order approximation requires time step embedded in the linear matrix formulation for matrix factorizations. Therefore, once the time steps are adjusted, LU decompositions are unavoidable.
- The method completes the simulation using comparable number of time steps, which solves time step constraint problem in matrix exponential-based nonlinear simulation [12]. Besides, the proposed method avoids costly regularization process [12] when inverted Krylov subspace method is utilized for MEVP.

The remainder of this paper is organized as follows. Sec. II states the formulation of nonlinear PDN simulation. Sec. III presents our method using one-stage exponential Rosenbrock-Euler approach. Sec. IV illustrates Krylov subspace method to approximate MEVP with invert Krylov subspace method. Sec. V shows the experimental results. In Sec. VI, we concludes this paper.

II. BACKGROUND

Time-domain simulation of PDN with nonlinear components is based on following differential equation system,

\[ \mathbf{F}(\mathbf{x}(t)) = \frac{d}{dt} \mathbf{q}(\mathbf{x}(t)) + \mathbf{f}(\mathbf{x}(t)) - \mathbf{Bu}(t) = \mathbf{0} \]  

(1)
where \( x(t) \) denotes nodal voltages and branch currents at time \( t \); \( q \) and \( f \) represent the charge/flux and current/voltage terms, respectively, \( u(t) \) is the external input at \( t \); \( B \) is an incident matrix mapping the input to the circuit system. To start the flow of PDN’s dynamic analysis, the DC solution (i.e., \( x(0) \)) is first obtained. Then, at each time step, implicit method, such as BE or TR, is applied to approximate the differential equations in current conventional numerical time-integration kernel [20]. In order to solve the above nonlinear system, Newton-Raphson method is often applied to obtain a converged solution. First, the nonlinear system is linearized at a given solution \( x_{n+1}^{(i)} \) (the following time-integration scheme is based on BE formulation):

\[
\left( \frac{C(x_{n+1}^{(i)})}{h_n} + G(x_{n+1}^{(i)}) \right) (x_{n+1}^{(i+1)} - x_{n+1}^{(i)}) = -F(x_{n+1}^{(i)}) \tag{2}
\]

where \( x_{n+1}^{(i)} \) is the numerical solution at \( t_{n+1} \) in the \( i \)-th iteration, and \( h_n = t_{n+1} - t_n \). \( C(x) \) and \( G(x) \) are linearized capacitance matrix and conductance matrix at \( x \), respectively. \( \frac{C(x_{n+1}^{(i)})}{h_n} + G(x_{n+1}^{(i)}) \) is the Jacobian matrix. Then, the corresponding linear system is solved in each iteration until solution \( x_{n+1}^{(i)} \) are converged to \( x_{n+1} \). During each iteration, the direct solver, e.g. LU factorization, is applied to solve linear [20]. In addition, when LTE is larger than error threshold, we need to change time step in order to reduce LTE. New iterations are required until the criteria of error budget is met.

A. Matrix Exponential-Based Circuit Simulation

Matrix exponential-based linear circuit simulation is another kind of time-integration kernel [8]. Weng, et al. [12] applied this framework to nonlinear circuit simulation. The system Eq. (1) is linearized to

\[
C(x(t)) \frac{dx(t)}{dt} = -G(x(t))x(t) + J_n(x(t)) + B(u(t)), \tag{3}
\]

where \( J_n(x) \) denotes the nonlinear dynamics at \( x \) (e.g. current vector from devices). The solution \( x_{n+1} \) at \( t_{n+1} \) is obtained via

\[
x_{n+1} = e^{h_n A(x_n)} x_n + \int_0^{h_n} e^{h_n - \tau} A(x_n) C^{-1}(x_n) (J_n(x_n + \tau)) + B(u(t_n + \tau)) d\tau \tag{4}
\]

where \( A(x_n) = -C(x_n)^{-1} G(x_n) \). Assume the external input \( u \) is piecewise-linear. The equation is written as

\[
x_{n+1} = e^{h_n A(x_n)} x_n + \frac{1}{2} (e^{h_n A(x_n)} - I) A(x_n)^{-1} (J_n(x_n) - \frac{1}{2} I n(x_n + 1)) + \frac{1}{h_n} (e^{h_n A(x_n)} - I) J_n x_n + B(u(t_n + h_n - u(t_n)) + e^{h_n A(x_n)} x_n. \tag{5}
\]

The variants can be derived from [12], [13] to obtain the solution \( x_{n+1} \). Nonetheless, the iteration schemes, such as Newton-Raphson or fixed point methods, cannot be avoided. In another word, the LU is still required in every iteration. Another important aspect is that Krylov subspace method is used for MEVP approximation. To simplify the notation, we use \( A \) instead of \( A(x_n) \). The subspace for MEVP is constructed as

\[
K_m(A,v) := \text{span}\{v, Av, \ldots, A^{m-1}v\}. \tag{6}
\]

The Krylov subspace is constructed by Arnoldi process, which has the following relation,

\[
A V_m = V_m H_m + h_{m+1,m} V_{m+1} e_{m+1}^T, \tag{7}
\]

where \( h_{m+1,m} \) is the \((m+1,m)\) entry of Hessenberg matrix \( H_m \). \( H_m \) is usually much smaller compared to \( A \) [19]. \( e_m \) is the vector with 1 in \( m \)-th element while others are zero. Then, MEVP is computed via

\[
e^{h A} v \approx \|v\| V_m e^{h H_m} e_1. \tag{8}
\]

We call this method as standard Krylov subspace for MEVP computation. However, some properties of above method limit it in nonlinear PDN simulation. First, the assumption is that \( C(x(t)) \) should be not singular. Otherwise, regularization process [12] is required. When the circuit is large, such process could be time-consuming. Besides, when the values of elements in \( C(x(t)) \) vary in magnitude [21]–[24], the system becomes very stiff. Standard Krylov subspace [12], [19] requires large number of bases dimension to approximate MEVP [15], which degrades performance of matrix exponential-based circuit simulation, so that it limited previous work for simulating mildly stiff circuit system [12], [13].

III. PDN Simulation Model Using Exponential Rosenbrock-Euler Formulation

In this section, we apply one-stage exponential Rosenbrock formulation [17], [18] for simulating nonlinear PDN system. This method still inherits the advantages from matrix exponential method, such as the nature of explicit formulation, and also reduce the number of LU decomposition operations.

A. Nonlinear Dynamical System Using Exponential Rosenbrock-Euler Formulation

Considering a non-autonomous differential equation system

\[
\frac{dx(t)}{dt} = F(x, u, t) \tag{9}
\]

The numerical solution \( x_{n+1} \) is derived as [17],

\[
x_{n+1} = x_n + h_n \phi_1(h_n J_n) F_n(x_n) + h_n^2 \phi_2(h_n J_n) g_n \tag{10}
\]

where \( F_n(x_n) = F(x_n, u_n, t_n) \),

\[
J_n = \frac{\partial F_n(x_n)}{\partial x}, \quad g_n = F_n(x_n) - J_n x_n \tag{11}
\]

\( J_n \) is the Jacobian of \( f \) and \( g_n \) is the nonlinear remainder, which is evaluated at \( x_n \).

\[
\phi_1(h_n J_n) = e^{h_n J_n} - I, \quad \phi_2(h_n J_n) = \phi_1(h_n J_n) - I \tag{12}
\]

One advantage of Eq. (10) is that the stable in the entire complex plane [25], which is superior to all of backward differentiation formula (BDF) methods. Therefore, the \( \phi \) function permit a large value for the step size \( h_n \).

B. Exponential Rosenbrock-Euler Formulation for PDN

To formulate PDN model using the exponential Rosenbrock-Euler, \( C(x) \) is first assumed to be invertible (in Sec. IV, we can avoid the direct inversion of \( C(x) \) by Krylov subspace approximation).

\[
\frac{dx(t)}{dt} + C(x(t))^{-1} G(x(t)) x(t) = C(x(t))^{-1} B(u(t)) + C(x(t))^{-1} I_{cur}(x(t)) \tag{13}
\]
The numerical solution at \( t_{n+1} \) is computed by

\[
x_{n+1} = x_n + h_n G(x_n)^{-1} C(x_n) G(x_n)^{-1} B \frac{u(t_{n+1}) - u(t_n)}{h_n} + (e^{h_n J_n} - I)(x_n - G(x_n)^{-1}(I_{cur}(x_n) + Bu(t_n))) + ((e^{h_n J_n} - I)G(x_n)^{-1}C(x_n)G(x_n)^{-1}B) \left( \frac{u(t_{n+1}) - u(t_n)}{h_n} \right)
\] (14)

where \( J_n = -C(x_n)^{-1}G(x_n) \), nonlinear effects are modeled as \( g_n = C(x_n)^{-1}I_{cur}(x_n) \), and

\[
F_n(h_n J_n) = J_n x_n + g_n + b(t_n)
\] (15)

\[
\phi_1(h_n J_n) = -\frac{1}{h_n}(e^{h_n J_n} - I)G^{-1}(x_n)C(x_n)
\] (16)

\[
\phi_2(h_n J_n) = -\frac{1}{h_n}(\phi_1(h_n J_n) - I)G^{-1}(x_n)C(x_n)
\] (17)

There are two MEVPs to evaluate \( e^{h_n J_n} v_1 \) and \( e^{h_n J_n} v_2 \),

\[
v_1 = x_n - G(x_n)^{-1}(I_{cur}(x_n) + Bu(t_n))
\] (18)

\[
v_2 = G(x_n)^{-1}C(x_n)G(x_n)^{-1}B \left( \frac{u(t_{n+1}) - u(t_n)}{h_n} \right)
\] (19)

This leads to

\[
x_{n+1}(h_n) = x_n - v_1 - v_2 + h_n v_2 + e^{h_n J_n} v_1 + e^{h_n J_n} v_2
\] (20)

Another advantage of this formulation using exponential Rosenbrock-Euler method is that \( J_n \) is maintained, only by changing step size \( h_n \) to minimize the nonlinear error. It is quite different from BDF methods, which updates the linear system with different time step size. Then, time step \( t_{n+1} \) is \( t_n + \alpha h_n \),

\[
x_{n+1}(\alpha h_n) = x_n - v_1 - v_2 + \alpha h_n v_2 + e^{\alpha h_n J_n} v_1 + e^{\alpha h_n J_n} v_2
\] (21)

IV. KRYLOV SUBSPACES-BASED MEVP IN EXPONENTIAL ROSENBRUCK-EULER METHOD

MEVP is computed using Krylov subspace method with invert bases [15]. The standard Krylov subspace in [12] performs well in mild systems. However, the standard subspace used in matrix exponential methods in [12], [13] for MEVP computation converges in a very slow rate, which also consumes huge memory. In the practical application of PDN simulation, the values of elements in matrix, especially in \( C(x) \), quite differ from each other, e.g., the range of device capacitance and the range of power network’s wire capacitance sometimes vary in magnitudes.

A. Krylov Subspace Method Using Inverted Basis for MEVP Computation

We construct the Krylov subspace as follows,

\[
K_m(A^{-1}, v) := \text{span}\{v, A^{-1}v, \ldots, A^{-(m-1)}v\}
\] (22)

which is still obtained by Arnoldi method,

\[
A^{-1}V_m = V_m H_m + h_{m+1,m}V_{m+1}e_m^T
\] (23)

and \( V_m^T A^{-1} V_m = H_m \). In proposed method, subspace is constructed as

\[
K_m(G^{-1}C, v) := \text{span}\{v, -G^{-1}Cv, \ldots, (-G^{-1}C)^{(m-1)}v\}
\] (24)

To simplify the notations, \( G = G(x_n), C = C(x_n) \). MEVP is computed as

\[
e^{h_n A} v = e^{-h_n C^{-1} G} v \approx \|v\|V_e e^{H_m^{-1}} e_1
\] (25)

This method is called as \textit{inert Krylov subspace} for MEVP approximation [15].

B. Error Estimation of Inverted Krylov Approximation

Error estimation is a control kernel for Krylov subspace convergence and adaptive stepping controlling. The difference among the exact solution’s derivative and approximated one is derived based on the work of [26],

\[
x' - x'_{exact} = \|v\| \|V_m H_m^{-1} - AV_m\| e^{h_n H_m} e_1 \approx \|v\| \|A h_{m+1,m} v_{m+1} e^{h_n H_m} e_1\|
\] (26)

Above error estimator contains \( A \), which requires the inversion of \( C \). However, we do not want this matrix due to its possible singularity and extra inversion operations. In PDN simulation, the difference of derivative \( x' \) is not our primary concern, but \( (C x' - C x'_{exact}) \) is, which has physical meaning for circuit, the residue contributed by node voltages and branch currents. Therefore, by multiplying \( C(x_n) \), we have

\[
r_m(h) = \|v\| \|h_{m+1,m} G v_{m+1} e^{h_n H_m^{-1}} e_1\|
\] (27)

The whole Arnoldi process for MEVP computation using inert Krylov subspace method is shown in \textbf{Algorithm 1}.

\textbf{Algorithm 1: MEVP\_Approx: MEVP approximation using inert Krylov subspace}

\begin{algorithmic}[1]
\State \textbf{Input:} \( C, G, v, \epsilon, B, h, t \)
\State \textbf{Output:} \( M_{MEVP}, V_m, H_m, v, m \)
1 \State \( v_1 = \frac{x}{\|x\|} \)
2 \For {j = 1 \colon m}
3 \State Solve \( G w = -C v_j \)
4 \For {i = 1 \colon j}
5 \State \( h_{i,j} = w^T v_i \)
6 \State \( w = w - h_{i,j} v_i \)
7 \EndFor
8 \State \( h_{j+1,j} = \|w\| \)
9 \State \( v_{j+1} = \frac{w}{h_{j+1,j}} \)
10 \If {\( \|h_{j+1,j}\| < \epsilon \) by Eq. (27) then}
11 \State \( m = j; \) break;
12 \EndIf
13 \EndFor
14 \State \( M_{MEVP} = \|v\|V_e e^{H_m^{-1}} e_1 \)
\end{algorithmic}

C. Nonlinear Error Control Scheme

There is another error source from nonlinear devices. The error estimation is derived based on Eq. (3.2) of [17],

\[
err \approx \|h_n \phi_1(h_n J_n)(g_{n+1} - g_n)\|
\] (28)

\[
= \|h_n e^{h_n J_n} - I\| \|G(x_n)^{-1} C(x_n) (g(x_{n+1}) - g(x_n))\|
\]

\[
\approx \|v\| \|V_m (e^{h_n H_m} - I) e_1\|
\]

where \( v = G(x_n)^{-1}(I_{cur}(x_{n+1}) - I_{cur}(x_n)) \). When the \( err \) is larger than error budget, change time step \( h_n = \alpha h_n \), where \( \alpha < 1 \). The advantage of our proposed method is there is no need to update the Krylov subspace, only by scaling the time step with the ratio \( \alpha \) (\( \alpha \leq 1 \)). The overall algorithm is shown in \textbf{Algorithm 2}.
Algorithm 2: Proposed nonlinear PDN solver

Input: PDN systems (netlist)
Output: Voltage/current solution vectors x during time period [0, T]
1. $t = 0, n = 0$
2. $x(0) = x_0 = \text{DC analysis}$
3. while $t < T$ do
   4. Obtain $C(x_n), G(x_n), \text{In}(x_n)$ after evaluation of devices at $x_n$
   5. LU decompose $G(x_n)$
   6. Compute $v_1$ via Eq. (18), and $e^{h_n + J_n} v_1$ via MEVP_Approx (Alg. 1)
   7. Compute $v_2$ via Eq. (19), and $e^{h_n + J_n} v_2$ via MEVP_Approx (Alg. 1)
   8. $i = 0$
   9. $x_{i+1} = x_n$
   10. while true do
        11. Compute $x_{i+1}^{n+1}$ using Eq. (20)
        12. Obtain $\text{In}(x_{i+1}^{n+1})$ after evaluation of devices at $x_{i+1}^{n+1}$
        13. Compute $\text{err}$ estimation by Eq. (28)
        14. if $\text{err}$ is smaller than error budget then
            15. $x(t + h) = x_{i+1}^{n+1}$
            16. break;
        end
        17. else
            18. $h = \alpha h$, such as $\alpha = 1/2$
            19. $i = i + 1$
        end
    end
    22. $t = t + h, n = n + 1$
    23. if iteration number $i$ small enough then
        24. $h = 2h$
        25. // fast convergence rate, then increase time step size
    end
end

V. EXPERIMENTAL RESULTS

We implement all algorithms in MATLAB R2013a and C/C++, where MOSFET devices are evaluated using opensource SPICE BSIM3 model implemented in the C programming language. The interactions between C/C++ and MATLAB is through MATLAB’s MEX interface. We use MATLAB’s UMFPACK package for LU factorization. The experiment is carried on a Linux workstation with Intel Core™ i7 4770 3.40GHz processor and 32GB memory.

Table I reports performance results of testcases PDN1-PDN5. BENR represents the backward Euler formulation with Newton-Raphson iterations, which serves the baseline of our comparisons. The LU RD: the reduction of total LU operations (Proposed method over BENR).

### Table I

<table>
<thead>
<tr>
<th>Designs &amp; Specifications</th>
<th>BENR</th>
<th>Proposed Method</th>
<th>LU RD</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDN1 55.0K 0.2K 19K 168K</td>
<td>#Steps</td>
<td>#Avg. NR</td>
<td>Runtime(s)</td>
</tr>
<tr>
<td>PDN2 170K 0.2K 91K 619K</td>
<td>3.05</td>
<td>59.3</td>
<td>398.8</td>
</tr>
<tr>
<td>PDN3 2.50M 0.4K 1.0M 6.30M</td>
<td>2.86</td>
<td>5.94</td>
<td>1460.2</td>
</tr>
<tr>
<td>PDN4 3.21M 2.0K 1.6M 9.70M</td>
<td>2.85</td>
<td>5.94</td>
<td>762</td>
</tr>
<tr>
<td>PDN5 55.0K 0.2K 28K 168K</td>
<td>3.05</td>
<td>5.94</td>
<td>15349.2</td>
</tr>
</tbody>
</table>

#Node represents the number of PDN size. #Device is the number of nonlinear MOSFET components within the PDN design. “#nnz” is a metric to represent sparsity of one matrix. PDN1-PDN4 has relatively sparser $C_i$ than PDN5.

Among PDN1-PDN4, our proposed method has average step number 813 smaller than 866 of BENR. The average number of LU operations is 2513 of BENR, while only 813 in our proposed method. We observe that speedups come from reduced LU operation numbers (around 3.1X reductions on average). Our method needs to build Krylov subspace, which is around 6.0 for each step on average in those PDN designs; Plus, we spend extra computational resources on MEVP computation by Eq. (25) and error estimations by Eq. (28). Those factors drag down the runtime improvement of transient simulation to some extent. The accuracy of above testcases is validated through finer time step version of our algorithms, our solutions converge well. Noted that the capacitance matrix of PDN5 has more non-zeros fill-ins and relatively complicated distribution, which results in more non-zero elements in factorized LU matrices than the ones of PDN1. By profiling this case, it shows another distinguished feature of our algorithm. We only LU decompose $G(x)$. When such less sparse matrix $C_i$ degrades the BENR, our method shows runtime performance boost, 6.9X speedup.

VI. CONCLUSIONS AND FUTURE WORK

In this work, we devise a novel and efficient time-domain nonlinear PDN algorithm using matrix exponential time integration kernel. The nonlinearity is modeled by exponential Rosenbrock-Euler method. The matrix exponential and vector product is computed via invert Krylov subspace to accelerate the convergence rate. Compared to de facto circuit solver algorithm using Newton-Raphson backward Euler method, our proposed method takes less LU decompositions per time step, while backward Euler needs at least two LU decompositions. Moreover, our method does not need to repeat decompositions when the length of time steps are adjusted for error controls. In our experimental results, with comparable time step numbers, the total LU operation numbers are 3.1X less on average. The runtime speedup is up to 6.9X. It is worth to point out that when the problem size become larger, such constant order speedup is more valuable when problem size scaling.

The work can be extended to more general VLSI designs, which contain much larger numbers of nonlinear devices with more complicated coupling effects. We find the promising property of our method handling those systems, since matrices within the matrix factorizations are more simple than the ones of traditional methods. Another aspect is how to reuse computed intermediate terms for error
estimations of nonlinear parts, and further improve the accuracy of our method.

VII. ACKNOWLEDGMENTS

We acknowledge the supports from NSF CCF-1017864, UCSD Powell Fellowship, Qualcomm FMA Fellowship and Jacobs Fellowship. We thanks Prof. Mike Botchev, Dr. Quan Chen, Ryan Coutts, Prof. Marlis Hochbruck, Dr. John Loffeld, Dr. Jingwei Lu, Prof. Alexander Ostermann, Dr. Qingwei Su, Prof. Mayya Tokman, Dr. Lining Zhang and Xiang Zhang for the helpful discussions.

REFERENCES
