SIAM J. Sci. Comput.
Vol. 39, No. 6, pp. A3040–A3066

A PARALLEL ITERATIVE ALGORITHM FOR DIFFERENTIAL LINEAR COMPLEMENTARITY PROBLEMS*

SHU-LIN WU† AND XIAOJUN CHEN‡

Abstract. We propose a parallel iterative algorithm for solving the differential linear complementarity problems consisting of two systems, a linear ODE system and a linear complementarity system (LCS). At each iteration we proceed in a *system decoupling* way: by using a rough approximation of the state variable obtained from the previous iteration, we solve the LCS; then we solve the ODE system and update the state variable for preparing for the next iteration, by using the obtained constraint variable as a known source term. The algorithm is highly parallelizable, because at each iteration the computations of both the LCS and the ODE system at all the time points of interest can start simultaneously. The parallelism for solving the LCS is natural and for the ODE system it is achieved by using the Laplace inversion technique. For the P-matrix LCS, we prove that the algorithm converges superlinearly with arbitrarily chosen initial iterate and for the Z-matrix LCS the algorithm still converges superlinearly if we use the initial value as the initial iterate. We show that this algorithm is superior to the widely used *time-stepping* method, with respect to robustness, flexibility, and computation time.

Key words. dynamic complementarity problems, parallel computation, iterations, convergence analysis

AMS subject classifications. 65M55, 65M12, 65M15, 65Y05

DOI. 10.1137/16M1103749

1. Introduction.

We are interested in solving the following differential linear complementarity problem (DLCP) with initial condition $x(0) = x_0$:

$$\dot{x}(t) = Ax(t) + By(t) + f(t),\ y(t) \in \mathrm{SOL}(Nx(t) + g(t), M),\ t \in (0, T), \tag{1.1}$$

where $x(t) \in \mathbb{R}^m$, $y(t) \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times m}$, $B \in \mathbb{R}^{m\times n}$, $N \in \mathbb{R}^{n\times m}$, $M \in \mathbb{R}^{n\times n}$, and $\mathrm{SOL}(Nx(t)+g(t), M)$ denotes the solution set of the linear complementarity problem (LCP):

$$0 \le y(t) \perp Nx(t) + g(t) + My(t) \ge 0. \tag{1.2}$$

Here and hereafter, we denote the linear complementarity problem (1.2) by $\mathrm{LCP}(q, M)$ with $q = Nx(t) + g(t)$. Applications of DLCPs and other related models, e.g., the differential variational inequalities arising from contact dynamics, span widely in the scientific and engineering fields; see the monographs [8, 16] and the excellent papers [2, 3, 5, 20, 23, 25, 27]. Throughout this paper, we assume that $f(t)$ and $g(t)$ in (1.1) are time-continuous functions.

*Submitted to the journal's Methods and Algorithms for Scientific Computing section November 16, 2016; accepted for publication (in revised form) September 18, 2017; published electronically December 19, 2017.

http://www.siam.org/journals/sisc/39-6/M110374.html

Funding: This work was supported by the NSFC of China (11771313), the Project of China Postdoctoral Science Foundation (2015M580777, 2016T90841), the Hong Kong Research Grant Council grant PolyU153000/15p, and the Hong Kong Polytechnic University Postdoctoral Fellowship Scheme.

†School of Science, Sichuan University of Science and Engineering, Zigong, Sichuan 643000, People's Republic of China (wushulin84@hotmail.com).

‡Department of Applied Mathematics, the Hong Kong Polytechnic University, Kowloon, Hong Kong (maxjchen@polyu.edu.hk).

The widely used approach for solving DLCPs is the time-stepping method [3, 4, 6, 7, 10, 11, 12, 13, 18, 19, 23, 24, 25], e.g., the famous implicit Euler method:

$$x_j = x_{j-1} + hAx_j + hBy_j + hf_j, \quad y_j \in \mathrm{SOL}(Nx_j + g_j, M), \tag{1.3}$$

where $x_j \approx x(t_j)$, $y_j \approx y(t_j)$, $j = 1, 2, \ldots, J$, and $h = T/J$ is the step-size. Here and hereafter, we assume that the time points $\{t_j\}_{j=0}^{J}$ are equally spaced, i.e., $\{t_j = jh\}_{j=0}^{J}$, but this is not a restrictive assumption since all the results obtained in this paper can be simply generalized to arbitrarily chosen time points. To forward (1.3) from one time point to the next, the common procedure is to solve x_j at first from the discrete ODE system, i.e.,

$$x_j = (I - hA)^{-1}x_{j-1} + h(I - hA)^{-1}By_j + h(I - hA)^{-1}f_j, \tag{1.4a}$$

and substitute x_j into the complementarity system to form a static LCS, as

$$y_j \in \mathrm{SOL}\left(q_j, M^h\right), \text{ with } M^h = hN(I - hA)^{-1}B + M, \tag{1.4b}$$

where $q_j = g_j + N(I - hA)^{-1}(hf_j + x_{j-1})$. Then, by solving $\mathrm{LCP}(q_j, M^h)$ and by substituting the solution y_j into (1.4a), we get x_j. From [8], the property of a $\mathrm{LCP}(q, M)$ is dominated by the matrix M. Precisely, if M is a P-matrix,[1] the LCS always has a unique solution for any input q; if M is positive semidefinite and the solution set $\mathrm{SOL}(q, M)$ is nonempty, the LCS has a unique least-norm solution; if M is a Z-matrix[2] and the feasible set $\mathrm{FEA}(q, M) := \{y | y \geq 0, q + My \geq 0\}$ is nonempty, the LCS has a unique least-element solution. The same conclusions go to $\mathrm{LCP}(q_j, M^h)$ in (1.4b) with the corresponding assumption on the matrix M^h. However, it is not easy to check whether M^h satisfies the assumption or not, because it depends on three matrices B, N, and M and it changes when we change the step-size h and/or the formula of the time-stepping method. This task becomes harder when the sizes of B, N, and M are large. Alternatively, researchers in this field usually assume that the matrix M satisfies some strong property and that the step-size h is sufficiently small. Apparently, under these two assumptions M^h inherits the property of M and thus the desired property of (1.4b) is guaranteed.

In this paper, we propose a method to solve (1.1), which avoids direct use of M^h. (Actually, as we will see, the matrix M^h never occurs in our algorithm.) Our algorithm is based on the following functional iteration:

$$y^{k+1}(t) \in \mathrm{SOL}\left(Nx^k(t) + g(t), M\right), \quad \begin{cases} \dot{x}^{k+1}(t) = Ax^{k+1}(t) + By^{k+1}(t) + f(t), \\ x^{k+1}(0) = x_0, \end{cases} \tag{1.5}$$

together with a novel discretization of the ODE system, where $k \geq 0$ denotes the iteration index and for $k = 0$ the function $x^0(t)$ is an initial guess. Functional iteration of this type is used widely in the study of DLCPs and DVIs; see, e.g., [10, 11, 12, 13, 18, 25]. In those papers, iteration of this type is often used as an intermediate step to study the convergence or to derive the error bounds of the time-stepping method; it is also used in [11] to prove the convergence of Newton's method. Functional iteration is, however, never used as a point of departure to design a practical numerical method for DLCPs. Indeed, the functional iteration itself cannot be used in practice, because it is impossible to get $(x^k(t), y^k(t))$ exactly. To make it useful we have to discretize

[1]A matrix $M \in \mathbb{R}^{n\times n}$ is a P-matrix if all the principal minors of M are positive.
[2]A matrix $M \in \mathbb{R}^{n\times n}$ is a Z-matrix if its off-diagonal elements are nonpositive.

the ODE system. The most natural choice is the time-stepping method, e.g., the implicit Euler method, which leads to

$$y_j^{k+1} \in \mathrm{SOL}(Nx_j^k + g_j, M),\ x_j^{k+1} = x_{j-1}^{k+1} + hAx_j^{k+1} + hBy_j^{k+1} + hf_j,$$

where $x_0^{k+1} = x_0$. Clearly, with $\{x_j^k\}_{j=1}^J$ being known from the previous iteration the computation of $\{y_j^{k+1}\}_{j=1}^J$ is naturally parallelizable.

However, the evolution of the discrete ODE system is a sequential process. To match the natural parallelization of the computation of the complementarity system, we abandon the widely used time-stepping methods and use another numerical treatment: the numerical Laplace inversion [21, 22, 28, 29, 30, 31]. This method cleverly utilizes the *linearity* of the ODE system and lies in representing the solution $x^{k+1}(t)$ by its inverse Laplace transform:

$$(1.6)\quad x^{k+1}(t) = \frac{1}{2\pi i}\int_\Gamma e^{zt}\widehat{\omega}(z)dz,\ \text{with } \widehat{\omega}(z) = (zI-A)^{-1}\left(x_0 + B\widehat{y}^{k+1}(z) + \widehat{f}(z)\right),$$

where Γ is a suitable contour in the complex plane and $\widehat{y}^k(z)$ and $\widehat{f}(z)$ are the Laplace forward transform of $y^k(t)$ and $f(t)$. Parameterizing the contour by $\Gamma : v \to z(v)$ with $v \in \mathbb{R}$, we can rewrite $x^{k+1}(t)$ as $x^{k+1}(t) = \frac{1}{2\pi i}\int_{-\infty}^{\infty} \dot{z}(v)e^{z(v)t}\widehat{\omega}(z(v))dv$. Then, by integrating the integral by the trapezoidal rule with step-size Δv and then by truncating the infinite summation to a finite one, we get a discrete analogue of (1.5) as

$$(1.7)\qquad y^{k+1}(t) \in \mathrm{SOL}\left(Nx_P^k(t) + g(t), M\right),\ x_P^{k+1}(t) = \frac{\Delta v}{2\pi i}\sum_{p=-P}^{P} \dot{z}_p e^{z_p t}\widehat{\omega}(z_p),$$

where $z_p = p\Delta v$, $\dot{z}_p = \dot{z}(v_p)$, $P > 1$ is an integer, and $x_P^k(t) \approx x^k(t)$. Clearly, for any time points $\{t_j\}_{j=1}^J$ of interest, the computation of $\{x_{P,j}^{k+1}\}_{j=1}^J$ is completely independent and is therefore naturally parallelizable as well. Moreover, solving the $(2P+1)$ linear algebraic equations, i.e., $\{(z_pI - A)^{-1}(x_0 + B\widehat{y}^k(z_p) + \widehat{f}(z_p))\}_{p=-P}^P$, is also naturally parallelizable. However, it is still impossible to use this scheme in practice, because it is difficult to get $\widehat{y}^k(z_p)$ (and $\widehat{f}(z_p)$ in many cases). Our idea toward avoiding direct use of the Laplace forward transform of $y^k(t)$ and $f(t)$, i.e., $\widehat{y}^k(z)$ and $\widehat{f}(z)$, is given in detail in section 2.1.

In summary, our motivation of this paper is twofold: design a highly parallelizable algorithm for solving the DLCPs by using the *linearity* of the DLCPs and remove the restriction on the step-size h in practice by solving the LCS and the ODE system separately. The remainder of this paper is organized as follows. In section 2, we present more details of algorithm (1.7) derived by applying the numerical Laplace inversion to the ODE system in (1.5). We then analyze the convergence of the algorithm for the P-matrix LCS. In section 3, we generalize our analysis to the Z-matrix LCS and other cases. Our numerical results are given in section 4 and we conclude this paper in section 5.

2. The algorithm and the convergence analysis for the P-matrix LCS.

In this section, we introduce the fully discrete analogue of (1.5), by applying the numerical Laplace inversion [21, 22, 28, 29, 30, 31] to the ODE system. We then analyze the convergence properties of the resulting iterative algorithm for the P-matrix LCS. A generalization to the Z-matrix LCS is given in the next section and the generalization to more broader cases is commented on in Remark 3.1. The following lemma shall be used in many places.

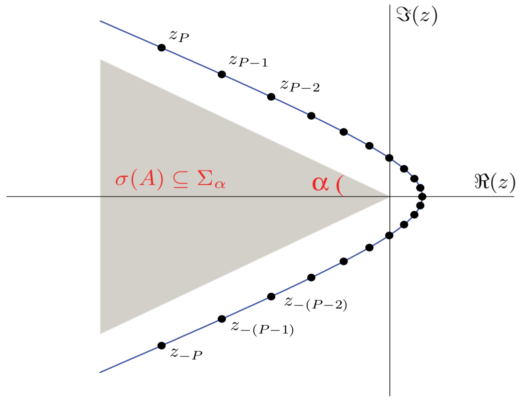

FIG. 1. *The hyperbolic contour* Γ *and the sector* Σ_α *that contains the spectrum* $\sigma(A)$. *The isolated dots* $\{z_p\}_{p=-P}^{P}$ *located on* Γ *denote the quadrature nodes used to discretize the contour integral in* (1.6).

LEMMA 2.1 (see [11, Corollary 2.1]). *Let M be a P-matrix and* $y(q_l) \in \mathrm{SOL}(q_l, M)$, $l = 1, 2$. *Then, it holds that* $\|y(q_1)-y(q_2)\| \leq L\|q_1-q_2\|$ *with* $L = \max_{S\subseteq\{1,\dots,n\}} \|M_{S,S}^{-1}\|$, *where $M_{S,S}$ denotes the principle submatrix of M indexed by the set S.*

This lemma implies that the solution $y(q)$ of a LCP(q, M) is a globally Lipschitz continuous function of the input vector q, when M is a P-matrix.

2.1. Derivation of the algorithm. To use the Laplace inversion technique, we need to know some information about the spectrum $\sigma(A)$ of the matrix A. For simplicity, we consider the following situation: $\sigma(A)$ is contained in a sector $\sum_\alpha = \{z : |\arg(-z)| \leq \alpha\}$ with some constant $\alpha \in (0, \pi/2)$, i.e.,

$$\sigma(A) \subseteq \Sigma_\alpha = \{z : |\arg(-z)| \leq \alpha\} \text{ with } \alpha \in (0, \pi/2). \tag{2.1a}$$

Under this assumption, it holds that (see [21, 22, 28])

$$\left\|(zI - A)^{-1}\right\| \leq \frac{\phi}{|z|} \text{ with } \phi = \frac{1}{\sin(\alpha)} \ \forall z \notin \Sigma_\alpha. \tag{2.1b}$$

For a function $u(t)$ with $t \in \mathbb{R}^+$, we denote by $\widehat{u}(z)$ the Laplace transform of u: $\widehat{u}(z) = \mathcal{L}(u)(z) := \int_0^{+\infty} e^{-zt}u(t)dt$. Applying the Laplace transform to the ODE system in (1.5) yields $\widehat{x}^{k+1}(z) = (zI - A)^{-1}[x_0 + B\widehat{y}^{k+1}(z) + \widehat{f}(z)]$. Then, inversing the Laplace transform along the contour Γ represents the solution $x^{k+1}(t)$ at time point t as given by (1.6). The contour Γ is a simple, closed, positively oriented curve that encloses both the spectrum of A and the singularities of $\widehat{y}^{k+1}(z)$ and $\widehat{f}(z)$. Popular choices of Γ are of Talbot type, parabolic type, and hyperbolic type; see the survey paper by Trefethen and Weideman [29, section 15] for more details. Here, we use the parameterized hyperbolic contour:

$$z(v) = \frac{\mu(1 + \sin(iv - \gamma))}{t} = \frac{\mu\,[1 - \sin(\gamma)\cosh(v) + i\cos(\gamma)\sinh(v)]}{t}, \tag{2.2}$$

where $v \in \mathbb{R}$, $\mu > 0$, and $\gamma \in (0, \pi/2 - \alpha)$.[3] The positive parameter μ/t controls the width of the contour and the other parameter γ determines its asymptotic angle; see Figure 1. Substituting $z = z(v)$ into (1.6) gives $x^{k+1}(t) = \frac{1}{2\pi i}\int_{-\infty}^{+\infty} e^{z(v)t}\widehat{\omega}(z(v))\dot{z}(v)dv$.

[3]The reason why γ must satisfy $\gamma \in (0, \pi/2 - \alpha)$ is explained in [30, section 4].

Then, applying the Trapezoidal rule to this contour integral gives approximation $x_P^{k+1}(t)$ of $x^{k+1}(t)$ for all $k \geq 0$, as given by (1.7). For $t > 0$, we can expect that the error $\|x_P^{k+1}(t) - x^{k+1}(t)\|$ decays exponentially as P increases, provided the contour parameters μ and γ and the quadrature step-size Δv are properly chosen.

Remark 2.1. The assumption (2.1a) excludes the case that the coefficient matrix A has zero eigenvalue, imaginary and/or unstable eigenvalues. Here, we propose an idea to extend the Laplace inversion technique to this case. Assume that $\sigma(A)$ consists of two parts: $\sigma(A) = \sigma_1(A) \cup \sigma_2(A)$, where $\sigma_1(A)$ is contained in the sector Σ_α as given by (2.1a) and $\sigma_2(A)$ consists of zero imaginary and/or unstable eigenvalues. For any $\lambda \in \sigma_2(A)$, let $\mathcal{C}$ be the circle centered at λ on the complex plane. Then, by using the Cauchy–Goursat theorem [1] we have

$$x^{k+1}(t) = \frac{1}{2\pi i}\int_\Gamma e^{tz}(zI-A)^{-1}\widehat{F}(z)dz + \frac{1}{2\pi i}\sum_{\lambda\in\sigma_2(A)}\oint_{\mathcal{C}} e^{tz}(zI-A)^{-1}\widehat{F}(z)dz, \tag{2.3}$$

where $\widehat{F}(z) = x_0 + B\widehat{y}^{k+1}(z) + \widehat{f}(z)$. Here, the integral along the circle $\mathcal{C}$ takes the counterclockwise orientation. Both integrals in (2.3) can be discretized by the trapezoidal rule after suitable parameterizations of Γ and $\mathcal{C}$. (The numerical analysis of discretization error corresponding to $\sigma_2(A)$ is incomplete and remains an ongoing work.) Note that, the time-stepping method can deal with the case $\sigma(A) = \sigma_1(A) \cup \sigma_2(A)$ uniformly, without special modification.

The contour quadrature given by (1.7) requires the values of the Laplace forward transforms $\widehat{f}(s)$ and $\widehat{y}^{k+1}(s)$ at the quadrature nodes $\{z_p\}_{p=-P}^{P}$, which, unfortunately, are unavailable in practice. So, the first step is to avoid direct use of the Laplace forward transform of $f(t)$ and $y^{k+1}(t)$. In our previous work [31], we proposed an idea for this goal, but the resulting quadrature for computing $x_P^{k+1}(t)$ depends on the derivative functions $\dot{y}^{k+1}(\tau)$ and $\dot{f}(\tau)$ for $\tau \in (0,t)$. This requirement is impractical here, because it is well known that we can only expect Lipschitz continuity instead of differentiability for the constraint variable of a DLCP.

Here, we propose a new idea toward avoiding direct use of $\widehat{f}(s)$ and $\widehat{y}^{k+1}(s)$. Let $K(t) = e^{tA}$ and $G(t) = \frac{1}{2\pi i}\int_\Gamma e^{zt}\widehat{K}(z)(B\widehat{y}^{k+1}(z) + \widehat{f}(z))dz$ with $\widehat{K}(z) = \mathcal{L}(K)(z) = (zI-A)^{-1}$. Then, we can represent (1.6) as

$$x^{k+1}(t) = \frac{1}{2\pi i}\int_\Gamma e^{zt}\widehat{K}(z)x_0 dz + G(t) = K(t)x_0 + G(t). \tag{2.4}$$

The idea here is twofold: treat the initial term x_0 exactly and treat the other term $G(t)$ by contour integral. Precisely, by choosing a contour Γ we can represent $K(t)$ as $K(t) = \mathcal{L}^{-1}(\widehat{K})(t) = \frac{1}{2\pi i}\int_\Gamma e^{zt}\widehat{K}(z)dz$. Then, by using the Duhamel formula we get

$$G(t) = \int_0^t K(t-\tau)f(\tau)d\tau = \int_0^t \left[\frac{1}{2\pi i}\int_\Gamma e^{z(t-\tau)}\widehat{K}(z)dz\right]\left(By^{k+1}(\tau) + f(\tau)\right)d\tau.$$

Thus, $G(t) = \frac{1}{2\pi i}\int_\Gamma \widehat{G}(t,z)dz$ with $\widehat{G}(t,z) = (zI-A)^{-1}\int_0^t e^{z(t-\tau)}(By^{k+1}(\tau)+f(\tau))d\tau$. (We include e^{zt} in the integrand $\widehat{G}(t,z)$ to avoid floating overflow.) Finally, substituting this into (2.4) gives a new representation of $x^{k+1}(t)$:

$$x^{k+1}(t) = e^{At}x_0 + \frac{1}{2\pi i}\int_{-\infty}^{+\infty}(z(v)I-A)^{-1}\left[\int_0^t e^{z(v)(t-\tau)}\left(By^{k+1}(\tau)+f(\tau)\right)d\tau\right]\dot{z}(v)dv.$$

Now, similar to (1.7) we get the following algorithm for solving DLCP (1.1):
(2.5)

$$\begin{cases} y^{k+1}(t) \in \mathrm{SOL}(Nx_P^k(t) + g(t), M), \\ x_P^{k+1}(t) = e^{At}x_0 + \frac{\Delta v}{2\pi i}\sum_{p=-P}^{P} \dot{z}_p(z_pI - A)^{-1}\int_0^t e^{z_p(t-\tau)}\left(By^{k+1}(\tau) + f(\tau)\right)d\tau. \end{cases}$$

Upon convergence, i.e., $k \to \infty$ in (2.5), we get the converged solution $(x_P^*(t), y^*(t))$ as
(2.6)

$$\begin{cases} 0 \le y^*(t) \perp Nx_P^*(t) + g(t) + My^*(t) \ge 0, \\ x_P^*(t) = e^{At}x_0 + \frac{\Delta v}{2\pi i}\sum_{p=-P}^{P} \dot{z}_p(z_pI - A)^{-1}\int_0^t e^{z_p(t-\tau)}\left(By^*(\tau) + f(\tau)\right)d\tau. \end{cases}$$

Remark 2.2 (parallelism of algorithm (2.5)). It is interesting to note that each iteration of (2.5) is highly parallelizable:

- The computations of both the constraint variables $y^{k+1}(t)$ and the state variables $x_P^{k+1}(t)$ at the time points $\{t_j\}_{j=1}^J$ of interest are naturally parallelizable, which means that the computations of the static LCS and then the ODE system at all the J time points can start simultaneously.
- At each time point t_j, let $b_p := \int_0^{t_j} e^{z_p(t_j-\tau)}(By^{k+1}(\tau) + f(\tau))d\tau$ and $X_p := (z_pI - A)^{-1}b_p$. Then, the computation of the $(2P + 1)$ linear problems in (2.6), i.e., $(z_pI - A)X_p = b_p$, is also naturally parallelizable.

Remark 2.3. If computing $e^{At}x_0$ is expensive in some situation, we can approximate it via Laplace inversion with another contour, too. For example, we can choose $\widetilde{\Gamma} = \{z : z(v) = \frac{\tilde{\mu}(1+\sin(iv-\tilde{\gamma}))}{t}, v \in \mathbb{R}\}$ and then approximate $e^{At}x_0$ via Laplace inversion along $\widetilde{\Gamma}$, as $e^{At}x_0 \approx \frac{\Delta v}{2\pi i}\sum_{p=-\tilde{P}}^{\tilde{P}} \dot{z}_p e^{z_pt}(z_pI - A)^{-1}x_0$, where $z_p = z(v_p)$ and $\dot{z}_p = \dot{z}(v_p)$. Carefully choosing $\tilde{\mu}$, $\tilde{\gamma}$, and Δv will lead to rapid reduction for the quadrature error as $\tilde{P}$ grows. Precisely, by choosing $\tilde{\mu} = 4.4921\tilde{P}$, $\tilde{\gamma} = 1.1721$, and $\Delta v = 1.0818/\tilde{P}$, according to [30, section 4] we have $\|e^{At}x_0 - \frac{\Delta v}{2\pi i}\sum_{p=-\tilde{P}}^{\tilde{P}} \dot{z}_p e^{z_pt}(z_pI - A)^{-1}x_0\| = O(e^{-2.32\tilde{P}})$. Indeed, replacing $e^{At}x_0$ in (2.5) by $\frac{\Delta v}{2\pi i}\sum_{p=-\tilde{P}}^{\tilde{P}} \dot{z}_p e^{z_pt}(z_pI - A)^{-1}x_0$ leads to more practical contour quadrature, but we shall not pursue this issue here, because our convergence analysis of algorithm (2.5) does not depend on the initial value x_0.

We cannot directly use algorithm (2.5) in practice, because it contains an integral $\int_0^t e^{z_p(t-\tau)}(By^{k+1}(\tau)+f(\tau))d\tau$, which cannot be exactly integrated. Suppose $\{t_j\}_{j=1}^J$ are the time points of interest for the solution $(x(t), y(t))$ of DLCP (1.1). Then, a natural idea for handling this integral is to construct piecewise polynomial $\widetilde{y}^{k+1}(t)$, which is obtained by interpolating $\{y_j^{k+1}\}_{j=0}^J$ at the time points $\{t_j\}_{j=0}^J$, and then replace $y^{k+1}(\tau)$ by $\widetilde{y}^{k+1}(\tau)$ in the integral. The integral $\int_0^t e^{z_p(t-\tau)}\widetilde{y}^{k+1}(\tau)d\tau$ can be integrated exactly. Similar treatment goes to the source term $f(t)$ if $\int_0^t e^{z_p(t-\tau)}f(\tau)d\tau$ cannot be exactly integrated as well. Convergence of the resulting algorithm is given in section 2.3, but we analyze algorithm (2.5) at first, because the proof provides a basis for the analysis of other cases.

2.2. Convergence of Algorithm (2.5).

The convergence of algorithm (2.5) and the accuracy of the converged solution $(x^*(t), y^*(t))$ is given as follows.

THEOREM 2.2. *Assume that the source terms $f(t)$ and $g(t)$ and the solution $x(t)$ of DLCP* (1.1) *are integrable functions on the interval $t \in [0, T]$. Assume that the spectrum $\sigma(A)$ of the matrix A satisfies* (2.1a) *with $\alpha + \gamma < \frac{\pi}{2}$ and that the matrix M is a P-matrix. Then, for algorithm* (2.5) *we have the following results.*

1. *For any $\mu > 0$, $\gamma \in (0, \frac{\pi}{2} - \alpha)$ and $\Delta v > 0$, it holds that*

$$\sup_{t\in[0,T]} \left\|x_P^k(t) - x_P^*(t)\right\| \leq \frac{\left(\frac{\Delta v \phi L \|B\| \|N\| \sqrt[k]{\prod_{l=1}^k \Theta_l}}{2\pi} T\right)^k}{k!} \sup_{t\in[0,T]} \left\|x_P^0(t) - x_P^*(t)\right\|, \tag{2.7a}$$

where ϕ is the constant given by (2.1b) *and Θ_l is defined by*

$$\Theta_l = l \int_0^1 (1-\tau)^{l-1} \left(\sum\nolimits_{p=-P}^{P} \left|\frac{\dot{z}_p}{z_p}\right| e^{\tau\mu[1-\sin(\gamma)\cosh(v_p)]}\right) d\tau \tag{2.7b}$$

with $\dot{z}_p = \dot{z}(v_p)$, $z_p = z(v_p)$, and $v_p = p\Delta v$.

2. *If we choose $\Delta v = \frac{2.0603}{\sqrt{P}}$ for the quadrature step-size and $\gamma = 0.794$, $\mu = \frac{\sqrt{P}}{4}$ as the parameters for the contour $\Gamma = \{z = z(v) : v \in \mathbb{R}\}$ defined by* (2.2)*, it holds that*

$$\sup_{t\in[0,T]} \|x(t) - x_P^*(t)\|_2 = O\left(\frac{e^{-2.06\sqrt{P}}}{\sqrt{P}}\right). \tag{2.8}$$

Note. The first result is concerned with the estimate of the convergence rate of algorithm (2.5) and the second result presents the accuracy of the converged solution if the inner integral $\int_0^t e^{z_p(t-\tau)}(By^{k+1}(\tau) + f(\tau))d\tau$ is exactly integrated. It will be proved in (2.15) that $\Theta_l \leq \Theta_{\max} := (2P+1)e^{\mu[1-\sin(\gamma)]}\sqrt{\frac{1+\sin(\gamma)}{1-\sin(\gamma)}} < \infty$ for all $l \geq 1$, which implies

$$\sqrt[k]{\prod_{l=1}^k \Theta_l} \leq \Theta_{\max}.$$

Hence, the bound given by (2.7a) shows that algorithm (2.5) converges superlinearly.

Proof. The proof is divided into two parts.

Part A: *the proof of* (2.7a). Let $\epsilon^k(t) = \|x_P^k(t) - x_P^*(t)\|$. Then, from (2.5) and (2.6),

$$\begin{aligned}\epsilon^{k+1}(t) &\leq \frac{\Delta v L \|B\| \|N\|}{2\pi} \int_0^t \left\|\sum_{p=-P}^{P} \dot{z}_p (z_p I - A)^{-1} e^{z_p(t-\tau)}\right\| \epsilon^k(\tau) d\tau \\ &\leq \frac{\Delta v \phi L \|B\| \|N\|}{2\pi} \int_0^t \sum_{p=-P}^{P} \left|\frac{\dot{z}_p}{z_p}\right| e^{\Re(z_p)(t-\tau)} \epsilon^k(\tau) d\tau,\end{aligned} \tag{2.9a}$$

where for the first (resp., second) inequality we have used Lemma 2.1 (resp., (2.1b)). Let

$$\theta(t,\tau) = \sum_{p=-P}^{P} \left|\frac{\dot{z}_p}{z_p}\right| e^{\Re(z_p)(t-\tau)} = \sum_{p=-P}^{P} \left|\frac{\cos(iv_p - \gamma)}{1 + \sin(iv_p - \gamma)}\right| e^{\frac{t-\tau}{t}\mu[1-\sin(\gamma)\cosh(v_p)]}, \tag{2.9b}$$

and $\epsilon_{\max}^0 = \sup_{t\in[0,T]} \epsilon^0(t)$. Then, using (2.9a) recursively gives

$$\begin{aligned}\epsilon^k(t) &\leq \left(\frac{\Delta v \phi L \|B\| \|N\|}{2\pi}\right)^k \int_0^t \theta(t,\tau_1) \cdots \int_0^{\tau_{k-1}} \theta(\tau_{k-1},\tau_k) \epsilon^0(\tau_k) d\tau_k \cdots d\tau_1 \\ &\leq \left(\frac{\Delta v \phi L \|B\| \|N\|}{2\pi}\right)^k \epsilon_{\max}^0 \int_0^t \theta(t,\tau_1) \cdots \int_0^{\tau_{k-1}} \theta(\tau_{k-1},\tau_k) d\tau_k \cdots d\tau_1.\end{aligned} \tag{2.10}$$

The kth integral, i.e., the rightmost one in the right-hand side of (2.10), satisfies

$$\int_0^{\tau_{k-1}} \theta(\tau_{k-1},\tau_k)d\tau_k = \int_0^{\tau_{k-1}} \sum_{p=-P}^{P} \left|\frac{\dot{z}_p}{z_p}\right| e^{\frac{\tau_{k-1}-\tau_k}{\tau_{k-1}}\mu[1-\sin(\gamma)\cosh(v_p)]}d\tau_k$$

$$= \tau_{k-1}\int_0^1 \sum_{p=-P}^{P} \left|\frac{\dot{z}_p}{z_p}\right| e^{\tau\mu[1-\sin(\gamma)\cosh(v_p)]}d\tau = \tau_{k-1}\Theta_1.$$

Here (and hereafter), we use an important fact that $\frac{\dot{z}_p}{z_p}$ is independent of t. Substituting this into the $(k-1)$th integral gives

$$\int_0^{\tau_{k-2}} \theta(\tau_{k-2},\tau_{k-1})\left(\int_0^{\tau_{k-1}} \theta(\tau_{k-1},\tau_k)d\tau_k\right)d\tau_{k-1}$$

$$= \Theta_1\int_0^{\tau_{k-2}} \tau_{k-1}\sum_{p=-P}^{P} \left|\frac{\dot{z}_p}{z_p}\right| e^{\frac{\tau_{k-2}-\tau_{k-1}}{\tau_{k-2}}\mu[1-\sin(\gamma)\cosh(v_p)]}d\tau_{k-1}$$

$$\overset{\tau=(\tau_{k-2}-\tau_{k-1})/\tau_{k-2}}{=\!=\!=\!=\!=} \Theta_1\tau_{k-2}^2\int_0^1 (1-\tau)\sum_{p=-P}^{P} \left|\frac{\dot{z}_p}{z_p}\right| e^{\tau\mu[1-\sin(\gamma)\cosh(v_p)]}d\tau = \frac{\tau_{k-2}^2}{2}\Theta_1\Theta_2.$$

Repeating this procedure, we arrive at

$$\int_0^t \theta(t,\tau_1)\cdots\int_0^{\tau_{k-1}} \theta(\tau_{k-1},\tau_k)d\tau_k\cdots d\tau_1$$

$$= \frac{\prod_{l=1}^{k-1}\Theta_l}{(k-1)!}\int_0^t \sum_{p=-P}^{P} \left|\frac{\dot{z}_p}{z_p}\right| e^{\frac{t-\tau_1}{t}\mu[1-\sin(\gamma)\cosh(v_p)]}\tau_1^{k-1}d\tau_1$$

$$\overset{\tau=(t-\tau_1)/t}{=\!=\!=\!=\!=} \frac{\prod_{l=1}^{k-1}\Theta_l}{(k-1)!}\int_0^1 \sum_{p=-P}^{P} \left|\frac{\dot{z}_p}{z_p}\right| e^{\tau\mu[1-\sin(\gamma)\cosh(v_p)]}[t(1-\tau)]^{k-1}t d\tau$$

$$= \frac{t^k\left(\prod_{l=1}^{k-1}\Theta_l\right)}{k!}\left[k\int_0^1 (1-\tau)^{k-1}\sum_{p=-P}^{P} \left|\frac{\dot{z}_p}{z_p}\right| e^{\tau\mu[1-\sin(\gamma)\cosh(v_p)]}d\tau\right] = \frac{t^k\left(\prod_{l=1}^{k}\Theta_l\right)}{k!}.$$

Substituting this into the second inequality in (2.10) gives the desired result (2.7a).

Part B: *the proof of* (2.8). To prove (2.8) for the accuracy of the converged solution, from the derivation of algorithm (2.5) we express the state variable $x(t)$ of DLCP (1.1) via the inverse Laplace transform along the parameterized contour Γ:

$$x(t) = e^{At}x_0 + \frac{1}{2\pi i}\int_{-\infty}^{+\infty} \dot{z}(v)(z(v)I-A)^{-1}$$
$$\left(\int_0^t e^{z(v)(t-\tau)}\left(By(Nx(\tau)+g(\tau))+f(\tau)\right)d\tau\right)dv,$$

where for any q we denote by $y(q)$ the solution of LCP(q,M). Let $e^k(t) = x_P^k(t)-x(t)$. Then, it holds that

(2.11)
$$e^{k+1}(t) = \frac{\Delta v}{2\pi i}\sum_{p=-P}^{P}\dot{z}_p(z_pI-A)^{-1}\int_0^t e^{z_p(t-\tau)}B\left[y(Nx_P^k(t)+g(t))-y(Nx(t)+g(t))\right]d\tau$$
$$+\,\mathtt{Err}(t),$$

where $\mathtt{Err}(t)$ denotes the error for the contour quadrature, i.e.,

$$\begin{aligned}\mathtt{Err}(t) = &\frac{\Delta v}{2\pi i}\sum_{p=-P}^{P}\dot{z}_p(z_pI-A)^{-1}\int_0^t e^{z_p(t-\tau)}\left(By(Nx(\tau)+g(\tau))+f(\tau)\right)d\tau\\ &-\frac{1}{2\pi i}\int_{-\infty}^{+\infty}\dot{z}(v)(z(v)I-A)^{-1}\\ &\qquad\left(\int_0^t e^{z(v)(t-\tau)}\left(By(Nx(\tau)+g(\tau))+f(\tau)\right)d\tau\right)dv.\end{aligned}$$

Under the assumption that $f(t)$, $g(t)$, and the solution $x(t)$ of DLCP (1.1) are integrable, it is easy to know that $y(Nx(\tau)+g(\tau))$ is also integrable, since the solution function $y(\cdot)$ is a Lipschitz continuous function. Then, from the analysis in [31, section 3.3] we have

$$\mathtt{Err}_{\max} := \sup_{t\in[0,T]}\|\mathtt{Err}(t)\|_2 = O\left(e^{-2.06\sqrt{P}}/\sqrt{P}\right)$$

if γ, μ, and Δv are chosen as stated. Substituting this into (2.11) gives

$$\left\|e^{k+1}(t)\right\| \le \frac{\Delta v\phi L\|B\|\|N\|}{2\pi}\int_0^t \theta(t,\tau))\left\|e^k(\tau)\right\|d\tau + \mathtt{Err}_{\max}, \tag{2.12}$$

where we have used Lemma 2.1. Here, $\theta(t,\tau)$ is given by (2.9b).

Let

$$\tilde{L} = \frac{\Delta v\phi L\|B\|\|N\|}{2\pi}. \tag{2.13}$$

Then, recursively using (2.12) gives

$$\left\|e^k(t)\right\| \le \left[\mathtt{Err}_{\max}\sum_{r=0}^{k-1}\tilde{L}^r\mathcal{J}_r(t)+\tilde{L}^k\mathcal{J}_k(t)\right]\sup_{\tau\in[0,t]}\left\|e^0(\tau)\right\|, \tag{2.14a}$$

where $\mathcal{J}_r(t)$ is the r-fold nested integral defined recursively as

$$\mathcal{J}_r(t) = \begin{cases}1, & r=0,\\ \int_0^t\theta(t,\tau)\mathcal{J}_{r-1}(\tau)d\tau, & r\ge 1.\end{cases} \tag{2.14b}$$

It is easy to see that $\mathcal{J}_r(t)$ (with $r=k$) is exactly the second nested integral given in (2.10). Hence, by using the analysis in Part A we have $\mathcal{J}_r(t) = \frac{t^r\left(\prod_{l=1}^r\Theta_l\right)}{r!}$, where Θ_l is the quantity defined by (2.7b). From (2.2) we have

$$\begin{aligned}\Theta_l &= l\int_0^1(1-\tau)^{l-1}\left(\sum_{p=-P}^{P}\left|\frac{\dot{z}_p}{z_p}\right|e^{\tau\mu[1-\sin(\gamma)\cosh(v_p)]}\right)d\tau\\ &\le\sqrt{\frac{1+\sin(\gamma)}{1-\sin(\gamma)}}\left[l\int_0^1(1-\tau)^{l-1}\left(\sum_{p=-P}^{P}e^{\tau\mu[1-\sin(\gamma)\cosh(v_p)]}\right)d\tau\right],\end{aligned}$$

where we have used $\left|\frac{\dot{z}(v_p)}{z(v_p)}\right| = \sqrt{\frac{\cosh(v_p)+\sin(\gamma)}{\cosh(v_p)-\sin(\gamma)}} \le \sqrt{\frac{1+\sin(\gamma)}{1-\sin(\gamma)}}$, which holds for all $v_p\in\mathbb{R}$ and can be verified by a routine calculation. Clearly, it holds that

$$\max_{l\ge 1}\Theta_l \le \Theta_{\max} \text{ with } \Theta_{\max} := (2P+1)e^{\mu[1-\sin(\gamma)]}\sqrt{\frac{1+\sin(\gamma)}{1-\sin(\gamma)}} < \infty. \tag{2.15}$$

Hence, it holds that

$$\mathcal{J}_r(t) \le \frac{(t\Theta_{\max})^r}{r!} \Rightarrow \begin{cases} \lim_{k\to\infty} \sum_{r=0}^{k-1} \tilde{L}^r \mathcal{J}_r(t) < \infty, \\ \lim_{k\to\infty} \tilde{L}^k \mathcal{J}_k(t) = 0. \end{cases} \tag{2.16}$$

This together with (2.14a) gives $\lim_{k\to\infty} \|e^k(t)\| = O(\frac{e^{-2.06\sqrt{P}}}{\sqrt{P}})$. ∎

Remark 2.4 (about the contour parameters). For $\gamma = 0.794$, the condition $\alpha+\gamma < \frac{\pi}{2}$ implies $\alpha < \frac{\pi}{2} - 0.794 = 0.7768$. If α, the maximal argument of the eigenvalue of the matrix A, exceeds this value, we can still expect that the error of the contour quadrature decays exponentially as $O(e^{-\delta\sqrt{P}}/\sqrt{P})$ as P increases, but with $\delta < 2.06$. In this case, we can properly choose the contour parameters μ and γ and the quadrature step-size Δv to maximize δ following the analysis in [31, section 3].

2.3. Implementation in practice. As we mentioned at the end of section 2.1, to use algorithm (2.5) in practice we need to construct piecewise polynomials by interpolating the values $\{y_j^{k+1}\}_{j=0}^J$ and $\{f_j\}_{j=0}^J$ at the time points $\{t_j\}_{j=0}^J$ and then compute this integral by replacing $y^{k+1}(\tau)$ and $f(\tau)$ by the corresponding interpolants. The natural idea is of course to construct the piecewise linear functions

$$\widetilde{y}^{k+1}(t) = y_j^{k+1}\left(1 - \frac{t-t_j}{t_{j+1}-t_j}\right) + y_{j+1}^{k+1}\frac{t-t_j}{t_{j+1}-t_j},\ t\in[t_j, t_{j+1}],$$
$$\widetilde{f}(t) = f(t_j)\left(1 - \frac{t-t_j}{t_{j+1}-t_j}\right) + f(t_{j+1})\frac{t-t_j}{t_{j+1}-t_j},\ t\in[t_j, t_{j+1}],$$

and replace $\int_0^{t_j} e^{z_p(t_j-\tau)}(By^{k+1}(\tau)+f(\tau))d\tau$ by $\int_0^{t_j} e^{z_p(t_j-\tau)}(B\widetilde{y}^{k+1}(\tau)+\widetilde{f}(\tau))d\tau$. We have

$$\int_0^{t_j} e^{z_p(t_j-\tau)}\left(B\widetilde{y}^{k+1}(\tau) + \widetilde{f}(\tau)\right)d\tau = \sum_{l=0}^{j-1} e^{z_p t_j}\left([a_l - b_l t_l]\omega_l^0 + b_l\omega_l^1\right),$$

where $a_l = By_l^{k+1} + f_l$, $b_l = \frac{a_{l+1}-a_l}{t_{l+1}-t_l}$ and

$$\omega_l^s = \int_{t_l}^{t_{l+1}} e^{-z_p\tau}\tau^s d\tau = \begin{cases} \frac{e^{-z_p t_l} - e^{-z_p t_{l+1}}}{z_p}, & s = 0, \\ \frac{e^{-z_p t_l}(1+t_l z_p) - e^{-z_p t_{l+1}}(1+t_{l+1}z_p)}{z_p^2}, & s = 1. \end{cases}$$

This treatment gives the following algorithm, which is a fully discrete analogue of (2.5):

$$\begin{cases} 0 \le y_j^{k+1} \perp Nx_{P,j}^k + g_j + My_j^{k+1} \ge 0, \\ x_{P,j}^{k+1} = e^{At_j}x_0 + \frac{\Delta v}{2\pi i}\sum_{p=-P}^{P} \dot{z}_p(z_pI - A)^{-1}\int_0^{t_j} e^{z_p(t_j-\tau)}\left(B\widetilde{y}^{k+1}(\tau) + \widetilde{f}(\tau)\right)d\tau, \end{cases} \tag{2.17}$$

where $j = 1, \ldots, N$. Upon convergence, the converged solution satisfies

$$\begin{cases} 0 \le y_j^* \perp Nx_{P,j}^* + g_j + My_j^* \ge 0, \\ x_{P,j}^* = e^{At_j}x_0 + \frac{\Delta v}{2\pi i}\sum_{p=-P}^{P} \dot{z}_p(z_pI - A)^{-1}\int_0^{t_j} e^{z_p(t_j-\tau)}\left(B\widetilde{y}^*(\tau) + \widetilde{f}(\tau)\right)d\tau, \end{cases}$$

where $\widetilde{y}^*(t)$ is the piecewise linear functions obtained by interpolating $\{y_j^*\}_{j=0}^J$.

THEOREM 2.3. *Under the assumption of Theorem 2.2, we have the following results.*

1. *For any $\mu > 0$, $\gamma \in (0, \frac{\pi}{2} - \alpha)$, and $\Delta v > 0$, the iterative scheme (2.17) is convergent.*

2. *If we use $\Delta v = \frac{2.0603}{\sqrt{P}}$ for the quadrature step-size and $\gamma = 0.794$, $\mu = \frac{\sqrt{P}}{4}$ as the parameters for the contour $\Gamma = \{z = z(v) : v \in \mathbb{R}\}$ defined by (2.2), it holds that*

$$\max_{0\le j\le J} \|x(t_j) - x^*_{P,j}\|_2 = O(h) + O\left(e^{-2.06\sqrt{P}}/\sqrt{P}\right). \tag{2.18}$$

Proof. For $t \in [t_l, t_{l+1}]$ we have

$$\widetilde{y}^{k+1}(t) - \widetilde{y}^*(t) = \left(y_l^{k+1} - y_l^*\right)\left(1 - \frac{t-t_l}{t_{l+1}-t_l}\right) + \left(y_l^{k+1} - y_l^*\right)\frac{t-t_l}{t_{l+1}-t_l}.$$

Hence, by using Lemma 2.1 it holds that

$$\left\|\widetilde{y}^{k+1}(\tau) - \widetilde{y}^*(\tau)\right\| \le L\|N\|\left(\left\|x^k_{P,l} - x^*_{P,l}\right\|\left(1 - \frac{t-t_l}{t_{l+1}-t_l}\right) + \left\|x^k_{P,l} - x^*_{P,l}\right\|\frac{t-t_l}{t_{l+1}-t_l}\right).$$

Let $\epsilon_j^k = \|x^k_{P,j} - x^*_{P,j}\|$ and $\widetilde{\epsilon}^k(t)$ be the piecewise linear function obtained by interpolating $\{\|x^k_{P,j} - x^*_{P,j}\|\}_{j=0}^J$ at the time points $\{t_j\}_{j=0}^J$. Let $\xi^k(t) = \sup_{\tau\in[0,t]} \widetilde{\epsilon}^k(\tau)$. Then, similar to (2.9a) it holds for $t = t_j$ that

$$\xi^{k+1}(t) \le \tilde{L}\int_0^t \theta(t,\tau)\xi^k(\tau)d\tau, \tag{2.19}$$

where $\tilde{L}$ is given by (2.13). Clearly, $\xi^k(T) = \max_{1\le j\le J}\epsilon_j^k$. Therefore, to prove the convergence of algorithm (2.17) it suffices to prove $\lim_{k\to\infty}\xi^k(T) = 0$. The difficulty of proving the latter lies in the fact that inequality (2.19) only holds on the time points $\{t_j\}_{j=1}^J$ and there is no guarantee that it holds for all $t \in (0,T)$ and therefore we cannot use it recursively as we did in the proof of Theorem 2.2. By noticing that $\theta(t,\tau) = \sum_{p=-P}^P |\frac{\dot{z}_p}{z_p}| e^{\frac{t-\tau}{t}[1-\sin(\gamma)\cosh(v_p)]}$ and that $\dot{z}_p/z_p$ is independent of t, we can rewrite (2.19) as

$$\xi^{k+1}(t) \le t\tilde{L}\int_0^1 \left(\sum_{p=-P}^P \left|\frac{\dot{z}_p}{z_p}\right| e^{\tau[1-\sin(\gamma)\cosh(v_p)]}\right)\xi^k(t(1-\tau))d\tau, \tag{2.20}$$

where $t \in \{t_j\}_{j=1}^J$. Without loss of generality, we assume $\xi^k(t_1) > 0$ for all $k \ge 0$, since otherwise if $\xi^{k^*}(t_1) = 0$ for some $k^* \ge 1$, we shall have $\xi^k(t) \equiv 0$ for all $t \in [0, t_1]$ and $k > k^*$ and in this case we only need to consider $t \in [t_2, T]$. Since $\xi^k(t)$ is an increasing function of t, the right-hand side of (2.20) is an increasing function of t as well. Hence, for $t \in [t_{j-1}, t_j]$ with $j \in \{2, 3, \dots, J\}$ we can always choose the constant

$$C_j = \frac{t_{j+1}}{t_j}\max_{k\ge0}\frac{\int_0^1\left(\sum_{p=-P}^P\left|\frac{\dot{z}_p}{z_p}\right|e^{\tau[1-\sin(\gamma)\cosh(v_p)]}\right)\xi^k(t_{j+1}(1-\tau))d\tau}{\int_0^1\left(\sum_{p=-P}^P\left|\frac{\dot{z}_p}{z_p}\right|e^{\tau[1-\sin(\gamma)\cosh(v_p)]}\right)\xi^k(t_j(1-\tau))d\tau} \ge 1,$$

such that $\xi^{k+1}(t) \le tC_j\tilde{L}\int_0^1(\sum_{p=-P}^P|\frac{\dot{z}_p}{z_p}|e^{\tau[1-\sin(\gamma)\cosh(v_p)]})\xi^k(t(1-\tau))d\tau$ for all $t \in$

$[t_j, t_{j+1}]$. Let $C^* = \max_{2\le j\le J-1} C_j$. Then, we have

$$\begin{aligned}\xi^{k+1}(t) &\le tC^*\tilde{L}\int_0^1 \left(\sum_{p=-P}^{P}\left|\frac{\dot{z}_p}{z_p}\right| e^{\tau[1-\sin(\gamma)\cosh(v_p)]}\right)\xi^k(t(1-\tau))d\tau \\ &= \tilde{L}^*\int_0^t \theta(t,\tau)\xi^k(\tau)d\tau \ \forall t\in[t_1,T],\end{aligned} \tag{2.21}$$

where $\tilde{L}^* = C^*\tilde{L}$. Now, by performing an analysis similar to Part A in the proof of Theorem 2.2 we have $\xi^k(T) \le \frac{(\tilde{L}^*\sqrt[k]{\prod_{l=1}^k \Theta_l})^k}{k!}\xi^0(T)$. This, together with (2.16) (with $\tilde{L}$ being replaced by $\tilde{L}^*$ there), implies $\lim_{k\to\infty}\xi^k(T) = 0$.

We now prove (2.18). Let $e_j^k = x_{P,j}^k - x(t_j)$. Then, similar to (2.11) we have

$$e_j^{k+1} = \frac{\Delta v}{2\pi i}\sum_{p=-P}^{P}\dot{z}_p(z_pI-A)^{-1}\int_0^t e^{z_p(t-\tau)}B\left[\widetilde{y}^{k+1}(\tau)-\widetilde{y}(\tau)\right]d\tau + \mathtt{Err}_1(t_j) + \mathtt{Err}_2(t_j),$$

where $\mathtt{Err}_1(t_j)$ and $\mathtt{Err}_2(t_j)$ are given by

$$\begin{aligned}\mathtt{Err}_1(t_j) &= \frac{\Delta v}{2\pi i}\sum_{p=-P}^{P}\dot{z}_p(z_pI-A)^{-1}\int_0^{t_j} e^{z_p(t_j-\tau)}\left(By(Nx(\tau)+g(\tau))+f(\tau)\right)d\tau \\ &\quad -\frac{1}{2\pi i}\int_{-\infty}^{+\infty}\dot{z}(v)(z(v)I-A)^{-1} \\ &\qquad \left(\int_0^t e^{z(v)(t-\tau)}\left(By(Nx(\tau)+g(\tau))+f(\tau)\right)d\tau\right)dv, \\ \mathtt{Err}_2(t_j) &= \frac{\Delta v}{2\pi i}\sum_{p=-P}^{P}\dot{z}_p(z_pI-A)^{-1}\int_0^{t_j} e^{z_p(t_j-\tau)}\left(B\widetilde{y}(\tau)+\widetilde{f}(\tau)\right)d\tau \\ &\quad -\frac{\Delta v}{2\pi i}\sum_{p=-P}^{P}\dot{z}_p(z_pI-A)^{-1}\int_0^{t_j} e^{z_p(t_j-\tau)}\left(By(Nx(\tau)+g(\tau))+f(\tau)\right)d\tau.\end{aligned}$$

The quantities $\mathtt{Err}_1(t_j)$ and $\mathtt{Err}_2(t_j)$ denote, respectively, the errors for the contour quadrature and the piecewise linear interpolation. For the error of the contour quadrature, as we mentioned in the proof of Theorem 2.2 it holds that $\|\mathtt{Err}_1(t_j)\|_2 = O(\frac{e^{-2.06\sqrt{P}}}{\sqrt{P}})$ if γ, μ, and Δv are chosen as stated. For the error of the piecewise linear interpolation, since $y(\cdot)$ is a Lipschitz continuous function, it is easy to know $\|\mathtt{Err}_2(t_j)\| = O(h)$; see Remark 2.5 below for an explanation. Hence, by letting $\mathtt{Err} = O(h) + O(\frac{e^{-2.06\sqrt{P}}}{\sqrt{P}})$ we have

$$e_j^{k+1} = \frac{\Delta v}{2\pi i}\sum_{p=-P}^{P}\dot{z}_p(z_pI-A)^{-1}\int_0^t e^{z_p(t-\tau)}B\left[\widetilde{y}^{k+1}(\tau)-\widetilde{y}(\tau)\right]d\tau + \mathtt{Err}.$$

By performing a similar deduction as that for deriving (2.14a) we get

$$\max_{0\le j\le J}\|e_j^k\| \le \left[\mathtt{Err}\sum_{r=0}^{k-1}\tilde{L}^r\mathcal{J}_r(t) + \tilde{L}^k\mathcal{J}_k(t)\right]\max_{0\le j\le J}\|e_j^0\| = O(h) + O\left(\frac{e^{-2.06\sqrt{P}}}{\sqrt{P}}\right),$$

where $\tilde{L}$ is given by (2.13) and for the equality "=" we have used (2.16). Now, by letting $k\to\infty$ we arrive at (2.18). ∎

Remark 2.5 (high order interpolation is unnecessary). From the proof of Theorem 2.3, we see that the term $O(h)$ appearing in the error estimate (2.18) arises from the error between a Lipschitz continuous function and its piecewise linear interpolant. For any function $u(t) \in C^1(0,T)$, the error of the piecewise linear interpolation is of order $O(h^2)$ with $h = \max_{0\le j\le J-1}(t_{j+1}-t_j)$. However, if $u(t)$ is only assumed to be Lipschitz continuous, we can only expect an error of order $O(h)$ in general.

The DLCP has very complicated dynamic properties and one of them is "the lack of differentiability" of the constraint variable $y(t)$. This has an implication for numerical computation in practice: for a given DLCP, unless we have some priori information about the time points where the constraint variable is nondifferential, we can only expect $\sup_{1\le j\le J}\|y(t_j)-y_j\| = O(h)$, no matter how accurately the ODE system is solved. This may be one of the reasons that most researchers in this field focus on studying the simple implicit Euler method (or some similar analogues), which is a typical lower order numerical method of order one.[4] We should explicitly point out that at the moment we have no intention to improve the accuracy by using the Laplace inversion as the numerical method for a DLCP. Our unique intention is to remove the restriction on the step-size h in practical computation and to maximize the parallelism for each of the functional iterations.

3. Generalization to the Z-matrix LCS.

The goal of this section is to generalize our work in the above section to the Z-matrix LCS, i.e., M is a Z-matrix in DLCP (1.1). In this case, from [11] we know that $\mathrm{LCP}(q,M)$ has a unique least-element solution[5] if the feasible set $\mathrm{FEA}(q,M) := \{y | y\ge 0, q+My\ge 0\}$ is nonempty. From [8], such a least-element solution can be obtained by solving the following linear programming problem:

$$\min \|y\|_1, \text{ s.t. } y\ge 0 \text{ and } My+q\ge 0, \tag{3.1}$$

where $\|y\|_1 = \sum_{l=1}^n y_l$ is the standard 1-norm for $y\ge 0$. Choosing the least-element solution for the LCS in (1.1) leads to the following least-element differential complementarity system:

$$\begin{cases} \dot{x}(t) = Ax(t) + By(t) + f(t) \text{ with } x(0) = x_0, \\ y(t) = \operatorname{argmin}\{\|v\|_1 : 0\le v \perp Nx(t) + g(t) + Mv \ge 0\}. \end{cases} \tag{3.2}$$

The corresponding algorithm based on the numerical Laplace inversion is

$$\begin{cases} y^{k+1}(t) = \operatorname{argmin}\{\|v\|_1 : 0\le v\perp Nx^k(t) + g(t) + Mv\ge 0\}, \\ x^{k+1}(t) = e^{At}x_0 + \frac{\Delta v}{2\pi i}\sum_{p=-P}^{P} \dot{z}_p(z_pI-A)^{-1}\int_0^t e^{z_p(t-\tau)}\left(By^{k+1}(\tau)+f(\tau)\right)d\tau. \end{cases} \tag{3.3}$$

To analyze the convergence of algorithm (3.3), we need the following lemma.

Lemma 3.1 (see [11, Theorem 2.3]). *Let $M\in\mathbb{R}^{n\times n}$ be a Z-matrix and $q_1,q_2\in\mathbb{R}^n$ such that $\mathrm{FEA}(q_1,M)\ne\emptyset$ and $\mathrm{FEA}(q_2,M)\ne\emptyset$. Then, we have*

$$\|y_{\min}(q_1) - y_{\min}(q_2)\| \le L\|q_1-q_2\|,$$

where $y_{\min}(q)\in \mathrm{SOL}(q,M)$ denotes the unique least-element solution and $L = \max\{\|M_{S,S}^{-1}\| : S\subseteq\{1,\dots,n\}$ and $M_{S,S}$ is nonsingular$\}$.

[4]The other reasons may be the simplicity for implementation and the strong stability for handing the stiffness of the ODE system.

[5]The least-element solution $y_{\min}$ is a solution of $0\le y\perp q+My\ge 0$ satisfying $y_{\min}\le y$ for all $y\in\mathrm{FEA}(q,M)$.

For the P-matrix LCS, the fact that lies in the heart of our convergence analysis for algorithm (2.5) is the Lipschitz continuity of the solution function $y(q)$ of $\mathrm{LCP}(q, M)$; see Lemma 2.1. Comparing Lemma 3.1 to Lemma 2.1, we see that to guarantee the Lipschitz continuity of the constraint variable the unique difference between the P-matrix LCS and the Z-matrix LCS is that for the latter we need to require the feasible set of the LCS to be nonempty. Therefore, it is easy to understand that for the least-element algorithm (3.3) if $\mathrm{FEA}(Nx_j^k + g_j, M) \neq \emptyset$ (with $j = 1, 2, \ldots, J$) at each iteration, the results given by Theorems 2.2 and 2.3 still hold.

Suppose $\mathrm{FEA}(Nx_0 + g(0), M) \neq \emptyset$ and there are constants $\beta_1 > 0$ and $\beta_2 > 0$ such that

$$\|\tilde{x} - x_0\| \leq \beta_1 \text{ and } \|\tilde{g} - g(0)\| \leq \beta_2 \Rightarrow \mathrm{FEA}(N\tilde{x} + \tilde{g}, M) \neq \emptyset. \tag{3.4}$$

Then, according to [11, Theorem 4.1] there exist suitable $T > 0$ and $\beta > 0$ such that

(3.5)
$$\forall t \in [0, T] : x(t) \in \mathcal{B}(x_0, \beta) := \{x : \|x - x_0\| \leq \beta\} \Rightarrow \mathrm{FEA}(Nx(t) + g(t), M) \neq \emptyset.$$

THEOREM 3.2. *For the DLCP* (3.2)*, let M be a Z-matrix and the spectrum $\sigma(A)$ of the matrix A satisfy* (2.1a)*. Assume that* $\mathrm{FEA}(Nx_0 + g(0), M) \neq \emptyset$ *and that* (3.4) *holds. Then, there exists some $T^* > 0$ such that the least-element Algorithm* (3.3) *is well-defined for $t \in [0, T^*]$, i.e.,* $\mathrm{FEA}(Nx^k(t) + g(t), M) \neq \emptyset$ *for $t \in [0, T^*]$, provided $\{x_{P,j}^0 = x_0\}_{j=0}^J$ and T^* satisfies*

$$\max_{k \geq 0} \left[C_0 \sum_{r=0}^{k-1} \left(T^* \tilde{L} \Theta_{\max}\right)^r \frac{1}{r!} + \left(T^* \tilde{L} \Theta_{\max}\right)^k \frac{1}{k!} \right] \leq \beta, \tag{3.6}$$

where $\Theta_{\max}$ is given by (2.15)*, β is given by* (3.5)*, $\tilde{L}$ is given by* (2.13)*, and C_0 is given by*

$$C_0 = \sup_{t \in [0, T^*]} \left(\|e^{At} - I\| \|x_0\| + \left\| \frac{\Delta v}{2\pi i} \int_0^t \left(\sum_{p=-P}^{P} \mathbf{A}_p e^{z_p(t-\tau)} \right) \left(f(\tau) + By^1(\tau)\right) d\tau \right\| \right)$$

with $\mathbf{A}_p = \dot{z}_p(z_p I - A)^{-1}$ and $y^1(t) = y_{\min}(Nx_0 + g(t))$.

Proof. We shall prove that if the initial iterate is chosen as $\{x_{P,j}^0 = x_0\}_{j=0}^J$ and T^* satisfies (3.6), all the iterates $\{x_P^k(t)\}_{k \geq 1}$ generated by the algorithm (3.3) lie in the ball $\mathcal{B}(x_0, \beta)$ defined by (3.5) for $t \in [0, T^*]$. Then, by using (3.5) repeatedly the well posedness of the least-element algorithm (3.3) follows.

From (3.3) we have

$$\begin{aligned}
&x_P^{k+1}(t) - x_0 \\
&\quad = \left(e^{At} - I\right) x_0 + \frac{\Delta v}{2\pi i} \int_0^t \left(\sum_{p=-P}^{P} \dot{z}_p (z_p I - A)^{-1} e^{z_p(t-\tau)} B \right) \left(y^{k+1}(\tau) - y^1(\tau)\right) d\tau \\
&\qquad + \frac{\Delta v}{2\pi i} \int_0^t \left(\sum_{p=-P}^{P} \dot{z}_p (z_p I - A)^{-1} e^{z_p(t-\tau)} \right) \left(f(\tau) + By^1(\tau)\right) d\tau \\
&\quad \Rightarrow \|x_P^{k+1}(t) - x_0\| \leq C_0 + \frac{\Delta v \|B\|}{2\pi} \int_0^t \theta(t, \tau) \left\|y^{k+1}(\tau) - y^1(\tau)\right\| d\tau,
\end{aligned}$$

where $\theta(t, \tau)$ is defined by (2.9b). Clearly, for $k = 0$ we have $\|x_P^1(t) - x_0\| \leq C_0$ and thus $x_P^1(t) \in \mathcal{B}(x_0, \beta)$ because of (3.6).

Suppose $x_P^k(t) \in \mathcal{B}(x_0, \beta)$ for $0 \le t \le T^*$. Then, by using Lemma 3.1 we know that $y^{k+1}(t)$ is uniquely existent and $\|x_P^{k+1}(t) - x_0\| \le C_0 + \tilde{L}\int_0^t \theta(t,\tau)\|x_P^k(\tau) - x_0\| d\tau$. Using this relation iteratively we get

$$\left\|x_P^k(t) - x_0\right\| \le \left[C_0 \sum_{r=0}^{k-1} \tilde{L}^r \mathcal{J}_r(t) + \tilde{L}^k \mathcal{J}_k(t)\right] \sup_{\tau\in[0,\tau]} \left\|x_P^k(\tau) - x_0\right\|, \tag{3.7}$$

where $\mathcal{J}_r(t)$ is the r-fold nested integral defined by (2.14b). In the proof of Theorem 2.3, we have already proved that $\mathcal{J}_r(t) = \frac{t^r(\prod_{l=1}^r \Theta_l)}{r!}$, where Θ_l is the quantity defined by (2.7b). From (2.15) we have $\mathcal{J}_r(t) \le \frac{(t\Theta_{\max})^r}{r!}$. Substituting this into (3.7) and then by using (3.6) we have $x_P^{k+1}(t) \in \mathcal{B}(x_0, \beta)$ for $t \in [0, T^*]$. ◻

In both Theorems 2.2 and 3.2, it is clear that, to get a better performance of the proposed iterative algorithm, the quantity $\Theta_{\max}$ defined by (2.15) should be as small as possible. This quantity heavily depends on the choice of P and the contour Γ. From (2.15), we see that $\Theta_{\max}$ increases linearly with respect to P. Fortunately, for a given time step-size h it is unnecessary to choose a very large P in practice. We explain this as follows. From Theorem 2.3 we know that the accuracy of the obtained numerical solution is of order $O(h) + O(e^{-2.06\sqrt{P}}/\sqrt{P})$, provided the spectrum $\sigma(A)$ satisfies (2.1a) and the contour Γ and the quadrature nodes z_p's are fixed according to Theorem 2.2. The second term decays very fast as P increases, and to balance the first term we just need a moderate value of P. For example, as h varies from 10^{-1} to 10^{-10} we need a P varying from 4 to 100.

Remark 3.1 (generalization to broader cases). To finish this section, we point out that the work in this paper can be generalized to broader cases, besides the P-matrix LCS and the Z-matrix LCS. The overall requirement is twofold: the Lipschitz continuity of the solution function $y(q)$ of LCP(q, M) and the (unique) existence of the solution function in some sense, e.g., the least-element solution, the least-norm solution or the spare solution, etc. From [8, 11], we know that the Lipschitz continuity of the solution function $y(q)$ holds for the case that M is a nondegenerate matrix or semidefinite matrix and many other cases, together with some additional conditions. To apply the algorithms proposed in this paper to these cases, we only need to explore suitable conditions such that the additional conditions are fulfilled in each iteration.

4. Numerical experiments.

In this section, we provide numerical results to validate the efficiency of the algorithm proposed in this paper. We consider the Z-matrix LCS only, because, compared to the P-matrix case, it is easier to verify whether a matrix is a Z-matrix or not. This kind of DLCPs arise frequently from the finite element or finite difference discretization of free boundary problems, reaction-diffusion problems, journal bearing problems, and equilibrium models in economics including input-output equilibrium models and Walrasian price equilibrium models [8, 9, 11, 17, 32].

For all the experiments, we divide the time interval $[0, T]$ equally with step-size h and algorithm (3.3) stops if

$$\max_{0\le j\le J} \left\|x_j^k - x_j^{\text{ref}}\right\|_\infty \le 10^{-12}, \tag{4.1}$$

where $\{x_j^{\text{ref}}\}_{j=0}^J$ denotes the converged solution. In all numerical experiments, the concerned LCP is solved by the semismooth Newton's method proposed in [11].

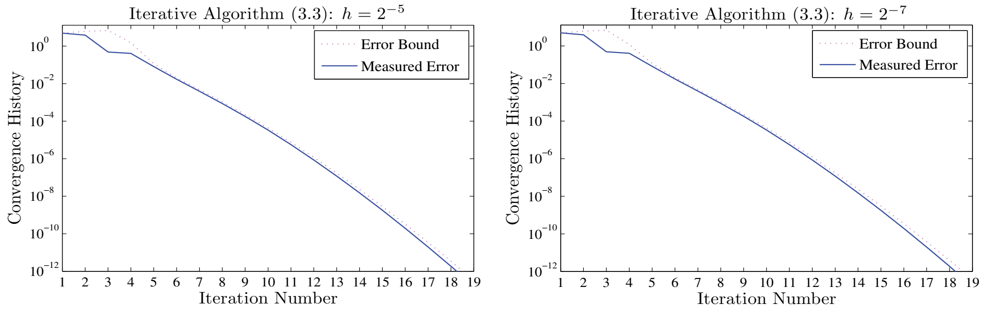

FIG. 2. *For two different values of the step-size h, the measured convergence rates of Algorithm* (3.3)*, together with the corresponding error bounds given by Theorem* 2.2.

4.1. A small-scale DLCP. In this first set of numerical results, we consider the following DLCP for $t \in [0, 1]$ with arbitrarily chosen coefficient matrices and source terms:
(4.2)

$$\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{pmatrix} = \overbrace{\begin{bmatrix}-2 & 1 & 0\\ 1 & -2 & \frac{3}{2}\\ 0 & -1 & -2\end{bmatrix}}^{=A}\begin{pmatrix}x_1\\ x_2\\ x_3\end{pmatrix} + \overbrace{\begin{bmatrix}-1 & -4 & -2 & 0\\ -3 & -2 & -2 & 0\\ -3 & -3 & 0 & 3\end{bmatrix}}^{=B}\begin{pmatrix}y_1\\ y_2\\ y_3\\ y_4\end{pmatrix} - \overbrace{\begin{pmatrix}\sqrt{t}[1-\cos(t^2)]\\ 3\cos(3\pi t)\\ \sqrt{t}[1-\cos(3t^2)]|\end{pmatrix}}^{=f(t)},$$

$$0 \le \begin{pmatrix}y_1\\ y_2\\ y_3\\ y_4\end{pmatrix} \perp \underbrace{\begin{bmatrix}0 & 0 & -1\\ -2 & -2 & \frac{3}{2}\\ -\frac{1}{2} & -1 & -\frac{1}{2}\\ 1 & \frac{5}{2} & \frac{3}{2}\end{bmatrix}}_{=N}\begin{pmatrix}x_1\\ x_2\\ x_3\end{pmatrix}$$

$$+ \underbrace{\begin{bmatrix}\frac{2}{3} & -1 & -\frac{1}{3} & -\frac{5}{3}\\ -\frac{2}{3} & \frac{1}{3} & -\frac{2}{3} & -\frac{4}{3}\\ -\frac{2}{3} & -\frac{1}{30} & 0 & -\frac{5}{3}\\ -1 & 0 & -\frac{5}{3} & \frac{1}{3}\end{bmatrix}}_{=M}\begin{pmatrix}y_1\\ y_2\\ y_3\\ y_4\end{pmatrix} + \underbrace{\begin{pmatrix}te^{-t}+\frac{3}{20}\\ -1.2\\ te^{-\frac{t}{3}}+\frac{3}{20}\\ te^{-\frac{t}{4}}+\frac{3}{20}\end{pmatrix}}_{=g(t)}.$$

For this DLCP, with two different values of the step-size h, $h = 2^{-5}$, and $h = 2^{-7}$, we show in Figure 2 the measured convergence rates of Algorithm (3.3), together with the error bound given by Theorem 2.2. We see that the convergence rate is strongly robust with respect to the change of the step-size and that the error bound predicts the measured convergence rate very well, except for the first few iterations.

With $h = 2^{-7}$, we next show in Figure 3 the solutions generated by running algorithm (3.3) for 1, 2, and 4 iterations at each time point, together with the converged solution, i.e., the "Ref.Solution" indicated by the solid line in each panel of Figure 3. In the bottom subfigure, we only plot the components $y_2(t)$ and $y_4(t)$, because $y_1(t) = y_3(t) \equiv 0$. We see that the algorithm proposed in this paper generates a satisfactory solution after only a few iterations.

It would be interesting to study the accuracy of the converged solution of algorithm (3.3). Because of the lack of differentiability of the constraint variable, the accuracy of the converged solution shall be of order $O(h) + O(e^{-2.06\sqrt{P}}/\sqrt{P})$; see

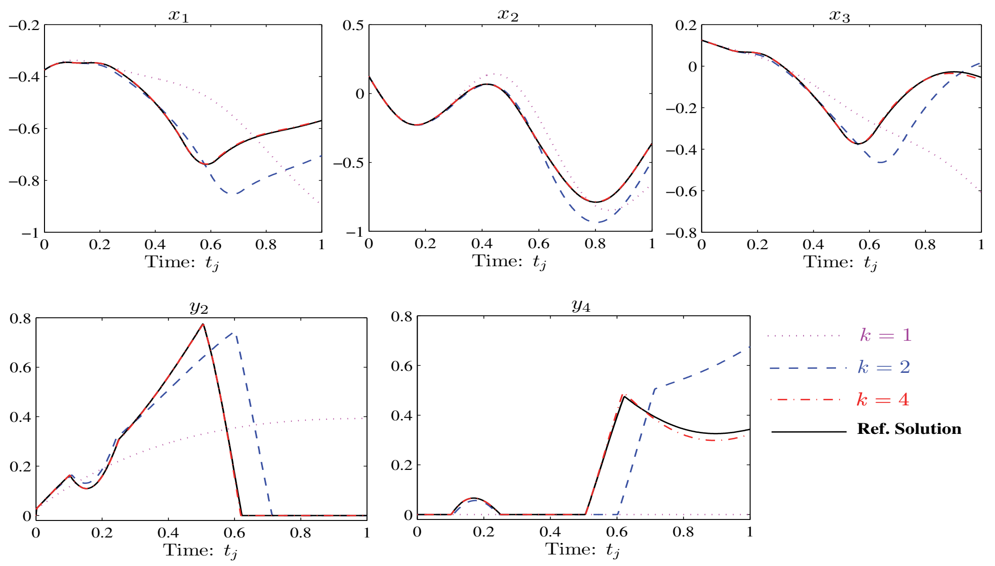

FIG. 3. *The solutions generated by running algorithm* (3.3) *for* 1, 2, *and* 4 *iterations, together with the converged solution, i.e., the "Ref.Solution" indicated by the solid line.*

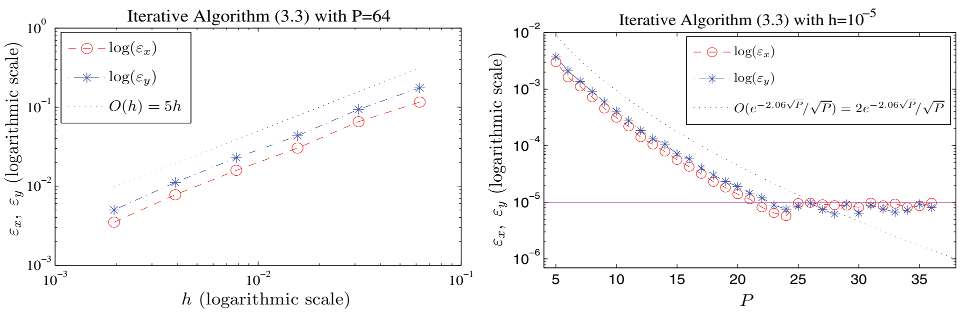

FIG. 4. *Left: for fixed* $P = 64$ *the quantities* ε_x *and* ε_y*, together with a bound* $O(h) = 5h$*, for* h *varies from* 2^{-3} *to* 2^{-9}*. Right: similar information when* $h = 10^{-5}$ *is fixed and* P *varies from* 4 *to* 38.

our discussion in section 2.3. We first check the dependence of the accuracy on the step-size h. To eliminate the effect of the error arising from the contour quadrature, we choose $P = 64$ for the number of quadrature nodes, which leads to an error of order $O(10^{-9})$ for the contour quadrature. Since the exact solution of DLCP (4.2) is unknown, we can only study the accuracy of the converged solution indirectly. We proceed as follows. Let $\{x_j^h, y_j^h\}$ and $\{x_j^{h/2}, y_j^{h/2}\}$ be two numerical solutions of DLCP (4.2) at the time points $\{t_j\}_{j=1}^N$ with $N = T/h$, obtained with step-sizes h and $\frac{h}{2}$, respectively. Then, we compute the difference between these two numerical solutions: $\varepsilon_x = \max_j \|x_j^h - x_j^{h/2}\|_\infty$ and $\varepsilon_y = \max_j \|y_j^h - y_j^{h/2}\|_\infty$.

Suppose $x(t_j) - x_j^h = O(h^p)$ with some constant $p \geq 1$. Then, we have $x_j^h - x_j^{h/2} = (x^h - x(t_j)) + (x(t_j) - x_j^{h/2}) = O(h^p) + O((\frac{h}{2})^p) = O(h^p)$. Hence, ε_x shall be a reliable prediction of the absolute error $x(t_j) - x_j^h$. The same statement goes to ε_y. In Figure 4 we plot the measured quantities ε_x and ε_y, together with a bound $O(h) = 5h$, as h varies from 2^{-3} to 2^{-9}. We see that both ε_x and ε_y decay with a rate $O(h)$ as h decreases. We next fix $h = 10^{-5}$ and vary the quantity P, the number of the quadrature nodes used in algorithm (3.3), from 4 to 38. The measured ε_x and ε_y for each P are plotted in Figure 4 on the right. We see that for $P \leq 23$, both ε_x and ε_y decay with a rate $O(e^{-2.06\sqrt{P}}/\sqrt{P})$ as P increases and when P exceeds 23 these two quantities stop decaying and stagnate around 10^{-5}. This can be explained like this: for $P = 23$ we have $e^{-2.0680\sqrt{P}}/\sqrt{P} = 1.05 \times 10^{-5} \approx h$; hence for $P > 23$ the accuracy of the converged solution of algorithm (3.3) is dominated by the term $O(h)$ arising from the piecewise linear interpolation. The results shown in Figure 4 confirm our error analysis given by Theorem 2.3 very well.

If we change the piecewise linear interpolation to high order interpolation, e.g., the cubic spline interpolation, the plots look very similar to Figure 4.

We next show that algorithm (3.3) is more flexible than the strategy of directly forwarding the implicit Euler method as described by (1.4a)–(1.4b). To this end, we show in Figure 5 the components $y_2(t)$ and $y_4(t)$ of constraint variable for three step-sizes. In the top row, we show the results for algorithm (3.3), where we see clearly that the difference between the profiles of $y_2(t)$ (and $y_4(t)$) for the three step-sizes is almost invisible, which implies that algorithm (3.3) generates a *reliable* solution of DLCP (4.2) for all these three step-sizes. The bottom row corresponds to the case of implicit Euler time-stepping method. Of particular interest is the case $h = 2^{-6}$ indicated by the thickened dashed line, for which the computation by the implicit Euler time-stepping method gives correct solution for $t \in [0, 0.36]$ and somehow incorrect solution for $t \in [0.36, 0.75]$. When t exceeds the critical time point $t = 0.75$, no numerical solution is found and the computation is broken.

This result for the implicit Euler time-stepping method can be explained by examining the coefficient matrix M^h for the LCS:

$$M_1^h = \begin{pmatrix} 0.6725 & -0.9942 & -0.3333 & -1.6725 \\ -0.6598 & 0.3480 & -0.6511 & -1.3246 \\ -0.6569 & -0.0226 & 0.0058 & -1.6696 \\ -1.0253 & -0.0263 & -1.6803 & 0.3421 \end{pmatrix},$$

$$M_2^h = \begin{pmatrix} 0.6782 & -0.9884 & -0.3334 & -1.6783 \\ -0.6528 & 0.3628 & -0.6355 & -1.3160 \\ -0.6472 & -0.0119 & 0.0117 & -1.6725 \\ -1.0506 & -0.0526 & -1.6939 & 0.3509 \end{pmatrix},$$

$$M_3^h = \begin{pmatrix} 0.6896 & -0.9770 & -0.3335 & -1.6897 \\ -0.6385 & 0.3925 & -0.6045 & -1.2993 \\ -0.6279 & \mathbf{0.0095} & 0.0232 & -1.6785 \\ -1.1007 & -0.1050 & -1.7207 & 0.3686 \end{pmatrix},$$

where $M_{1,2,3}^h$ correspond to $h = 2^{-8}$, $h = 2^{-7}$, and h^{-6}, respectively. We see that the first two matrices are Z-matrix, but for $h = 2^{-6}$ the matrix M_3^h is not a Z-matrix since it has a positive off-diagonal element 0.0095. So, for a given input vector q there is no guarantee that the solution set $\mathrm{SOL}(q, M_3^h)$ is nonempty or the least-element solution is correctly founded.

4.2. A large-scale DLCP.

We now consider a large-scale DLCP arising from the space discretization of the *parabolic Signorini problem* in a real parallel computation situation.

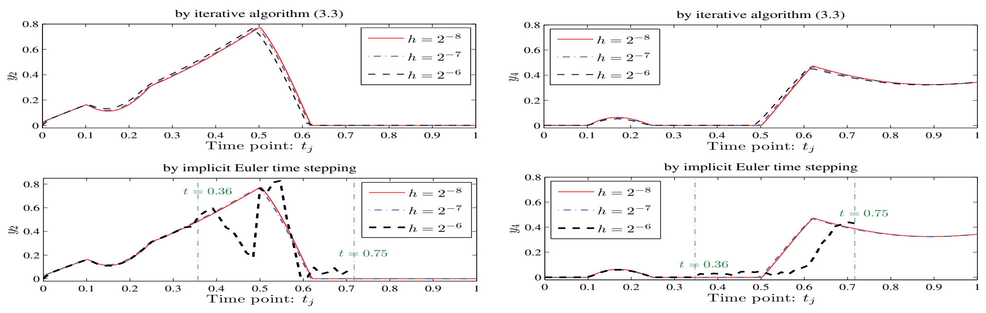

FIG. 5. *Top row: the converged solutions $y_2(t)$ and $y_4(t)$ by Algorithm* (3.3) *with three stepsizes h. Bottom row: similar information for the case that the discrete DLCP is solved directly by the procedure described by* (1.4a)–(1.4b).

4.2.1. The parabolic Signorini problem. The parabolic Signorini problem studied here consists of a regular diffusion equation and a Signorini boundary condition:

$$\begin{cases} c\Delta V - \partial_t V = 0 & \text{in } \Omega_T := \Omega \times (0,T), \\ 0 \le \partial_\nu V \perp (V-\psi) \ge 0 & \text{on } \mathcal{M}_T := \mathcal{M} \times (0,T), \\ V = f & \text{on } \mathcal{S}_T := \mathcal{S} \times (0,T), \\ V(\cdot,0) = V_0 & \text{on } \Omega_0 := \Omega \times \{0\}, \end{cases} \tag{4.3}$$

where $\mathcal{S} = \partial\Omega \setminus \mathcal{M}$, $c > 0$ is the diffusion coefficient, ∂_ν denotes the outer normal derivative on $\partial\Omega$ (the boundary of the spatial domain $\Omega \in \mathbb{R}^d$ with $d \ge 2$), $\mathcal{M}$ denotes an open subset of $\partial\Omega$ (in its relative topology), and V denotes the pressure of the chemical solution satisfying a diffusion equation $c\Delta V - \partial_t V = 0$ over the interior of the domain Ω. Here, we consider the following setting for the domain:

$$\Omega = (0,1) \times (0,1), \ \mathcal{M} = (0,1) \times \{0\}. \tag{4.4}$$

Classical examples where Signorini-type boundary conditions appear are the problems with unilateral constraints in elastostatics, problems with semipermeable membranes in fluid mechanics (including the phenomenon of osmosis and osmotic pressure in biochemistry), and the problems on the temperature control on the boundary in thermics. We refer to the books of Duvaut and Lions [15] and Petrosyan, Shahgholian, and Uraltseva [26], where many such applications are discussed and the mathematical models are derived. We also refer the interested reader to the survey paper [14] for the most recent progress of this kind of problems.

The region $\mathcal{M}$ denotes the *semipermeable* part of the boundary, which can be considered as a semipermeable membrane that is permeable only for a certain type of molecules (solvents) and blocks other molecules (solutes); see Figure 6 on the left. Because of the chemical imbalance, the solvent flows through the membrane from the region of smaller concentration of solute to the region of higher concentration, due to the *osmotic pressure* ψ. The flow occurs in one direction and continues until a sufficient pressure builds up on the other side of the membrane to compensate for osmotic pressure, which then shuts the flow. The boundary condition on the semipermeable part $\mathcal{M}$, which is terminologically called *Signorini boundary condition*, is determined by the mechanism of semipermeable osmosis described above; see Figure 6 on the right.

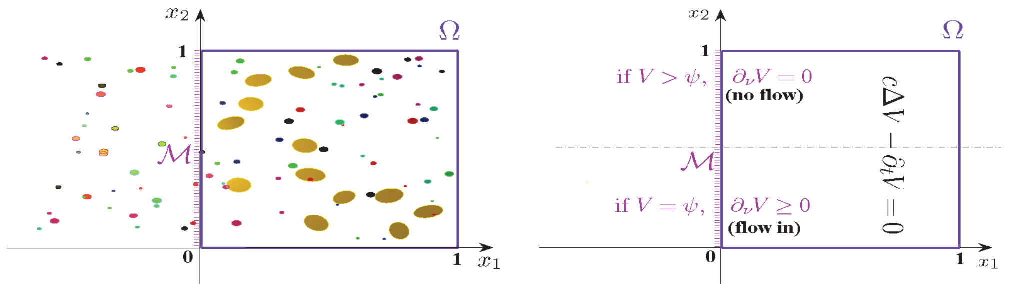

FIG. 6. *The semipermeable membranes and osmosis in the two-dimensional case. Left: the semipermeable membrane is a membrane that is permeable only for a certain type of molecules (solvents) and blocks other molecules (solutes). Right: the mathematical formulation of the unilateral problem illustrated on the left.*

For discretization, we employ the spatial grids $\{(x_{1,p}, x_{2,q}) = (p\Delta x, q\Delta x)\}_{p,q=0}^{Q+1}$ with $Q = \frac{1}{\Delta x} - 1$. Then, applying the centered finite difference formula to the spatial derivatives in (4.3) gives the following DLCP (details are presented in Appendix A):

$$\dot{\mathbf{X}}(t) = \mathbf{A}\mathbf{X}(t) + \mathbf{B}\mathbf{Y}(t) + \mathbf{F}(t), \quad 0 \leq \mathbf{Y}(t) \perp \mathbf{N}\mathbf{X}(t) + \mathbf{M}\mathbf{Y}(t) + \mathbf{G}(t) \geq 0, \tag{4.5}$$

where $\mathbf{X}(t) \in \mathbb{R}^{Q^2}$, $\mathbf{Y}(t) \in \mathbb{R}^Q$, $\mathbf{F}(t) = \mathbf{f}(t) + \frac{c}{\Delta x^2}(I \otimes \mathbf{E}_1)\boldsymbol{\psi}(t) \in \mathbb{R}^{Q^2}$, and $\mathbf{G}(t) = -\mathbf{g}(t) + (2I - \Delta x^2 A)\boldsymbol{\psi}(t) \in \mathbb{R}^Q$ with $I \in \mathbb{R}^{Q\times Q}$ being the identity matrix, $\mathbf{E}_1 = (1, 0, \ldots, 0)^\top \in \mathbb{R}^Q$, and $\otimes$ denotes the Kronecker product. The four coefficient matrices are defined by

(4.6)
$$\mathbf{A} = c\,(I \otimes A + A \otimes I)\,, \ \mathbf{B} = \frac{c}{\Delta x^2}(I \otimes \mathbf{E}_1), \ \mathbf{M} = 2I - \Delta x^2 A, \ \mathbf{N} = -2(I \otimes \mathbf{E}_1^\top)$$
$$\text{with } A = -\frac{1}{\Delta x^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}_{Q\times Q}.$$

Clearly, the matrix $\mathbf{M}$ is a Z-matrix.

4.2.2. Parallel computation. For numerical experiments, we use the following data:

(4.7)
$$t \in (0, 4), \ c = 2 \times 10^{-3}, \ f \equiv 0, \ P = 25 \text{ (number of contour quadrature nodes)},$$
$$V_0(x_1, x_2) = 2x_1x_2(1 - x_1)(1 - x_2), \ \psi(x_2, t) = \begin{cases} \frac{4}{1+t} & \text{if } |x_2 - \frac{1}{2}| \geq \frac{1}{4}, \\ \sin(2\pi t) & \text{otherwise.} \end{cases}$$

For algorithm (3.3), we set the following tolerance for the iterations:

$$\max_{0\leq j\leq J} \|\mathbf{X}_j^k - \mathbf{X}_j^{\mathrm{ref}}\|_\infty \leq \frac{\min\{h, \Delta x^2\}}{50}, \tag{4.8}$$

where $J = \frac{T}{h}$ and $\{\mathbf{X}_j^{\mathrm{ref}}\}_{j=0}^J$ denotes the converged solution. This tolerance shall be sufficient to match the error arising from the spatial discretization and the piecewise linear interpolation used for treating the inner integral involved in the Laplace inversion.

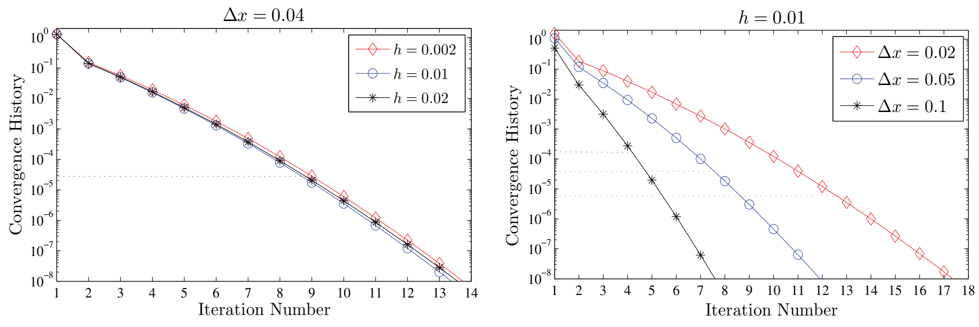

FIG. 7. *Left: convergence rates of Algorithm* (3.3) *for three values of the temporal step-size h, while the spatial mesh size is fixed to $\Delta x = 0.04$. Right: similar information as given in the left subfigure, when $h = 0.01$ is fixed and Δx varies.*

All experiments are conducted by using the following hardware and software:

- *CPU*: Intel Core i7-3770K 3.5 GHz and 32 GB RAM using gcc 4.8.1. A single CPU was used for the sequential computation of DLCP (4.5) by using the implicit Euler method. The codes were tested with gcc's fast math option (`ffast_math`).
- *GPU*: NVIDIA GeForce GTX 660 installed in a system with the above described CPU. The GPU operates at 1.10 GHz clock speed and consists of 5 multiprocessors (each contains 192 CUDA cores). We compiled the code using CUDA version 5.5 in combination with the gcc 4.8.1 compiler with fast math option (`use_fast_math`).

Particularly, the GPU is used to carry out the parallel computation of DLCP (4.5) at the discrete time points. As mentioned in Remark 2.2, for each discrete time point the $(2P + 1)$ linear systems involved in the Laplace inversion are independent, and therefore we solve them in parallel by GPU as well. For both the computations carried out by CPU and CPU, the linear algebras concerning the computation of the ODE system is solved by Fourier spectral method, by noticing the special structure of the matrix $\mathbf{A}$ given by (4.6).

We first check the convergence rates of algorithm (3.3) for different discretization parameters. In Figure 7 on the left, we show the measured error between the current iterate and the converged solution for three values of the temporal step-size h, while the spatial mesh-size is fixed to $\Delta x = 0.04$. We see that the convergence rate is strongly robust with respect to the change of h, the same as we observed in Figure 2 in the first example. If we fix h, the convergence rate somehow slightly increases as Δx decreases; see Figure 7 on the right. This issue is worth further study and shall be addressed in our forthcoming work. In both subfigures, the (dotted) horizontal line denotes the tolerance given by (4.8) which indicates where the algorithm should stop in practice.

With $h = 0.01$ and $\Delta x = 0.025$, we show in Figure 8 the approximation of the shifted solution $V^k(0, x_2, t) + \psi$ on the semipermeable boundary $\mathcal{M}$, after $k = 1$, 3, and 5 iterations, to the converged solution $V(0, x_2, t) + \psi$. We see that each iteration needs approximately 6 seconds and after 30 seconds the solution generated by algorithm (3.3) is sufficiently close to the converged solution. These four subfigures have subtle difference for the evolution of the four large bulges. A local description of this difference is shown in Figure 9, where we show the 1st, 2nd, and 5th iterates and the converged solution as a function of x_2 when $t = 4$ (and as a function of t when $x_2 = 0.5$).

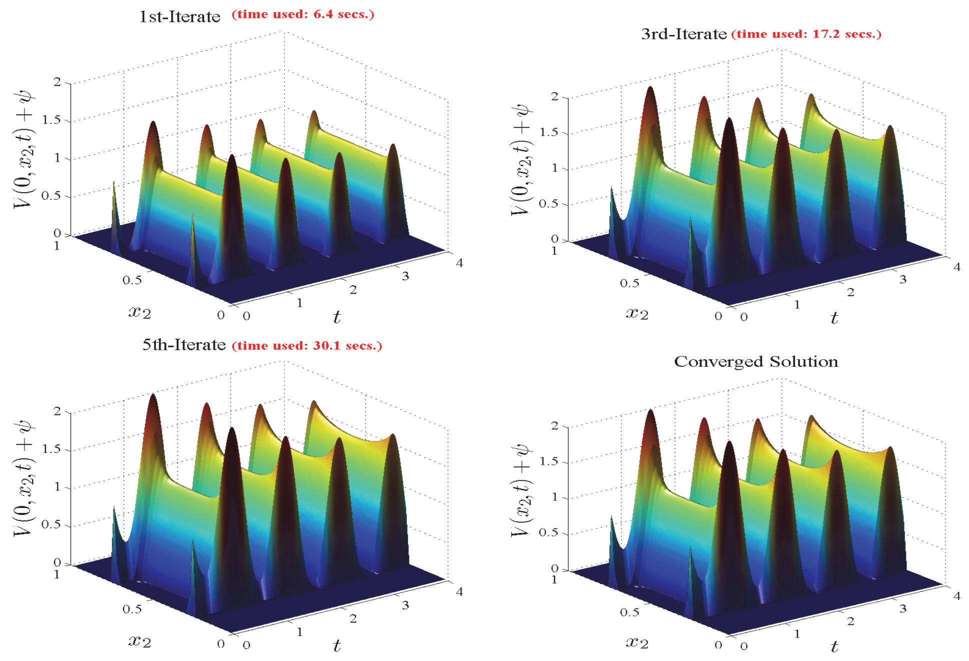

FIG. 8. *Approximation of the shifted solution $V^k(0,x_2,t)+\psi$ at the semipermeable boundary $\mathcal{M}$, after $k=1,3$ and 5 iterations, to the converged solution $V(0,x_2,t)+\psi$.*

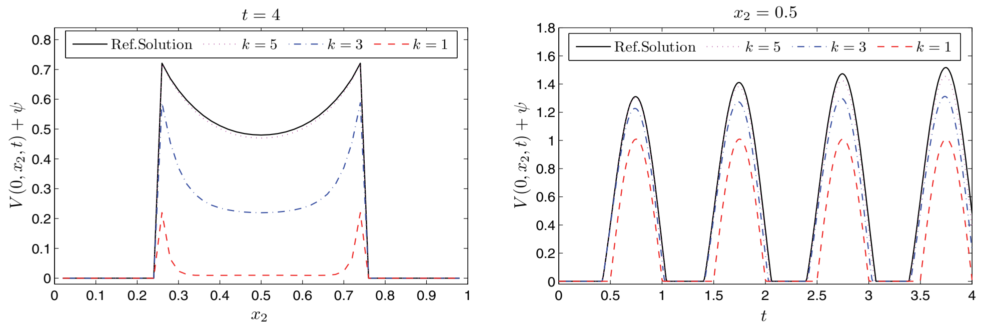

FIG. 9. *Left: the* 1st, 3rd, *and* 5th *iterates and the converged solution as a function of x_2 when $t=4$. Right: similar information when $x_2=0.5$ and t varies from 0 to 4.*

We now illustrate the advantages of using the parallel computation over the sequential computation. With $h=0.01$, $\Delta x=0.025$, and $T=4$, as we already saw in Figure 8, algorithm (3.3) needs 30 seconds to generate accurate approximation of the converged solution. However, it takes about 27.5 minutes to finish the computation for $t\in(0,4)$ by the implicit Euler method step by step (i.e., in the sequential computation mode). An illustration of the evolution of the numerical solution generated by the implicit Euler method in the sequential mode, after 30, 180, 900, and 1200 seconds, is shown in Figure 10. By comparing Figure 8 to Figure 10, it is clear that algorithm (3.3) with full parallelization is dramatically faster than the implicit Euler method used in the sequential mode.

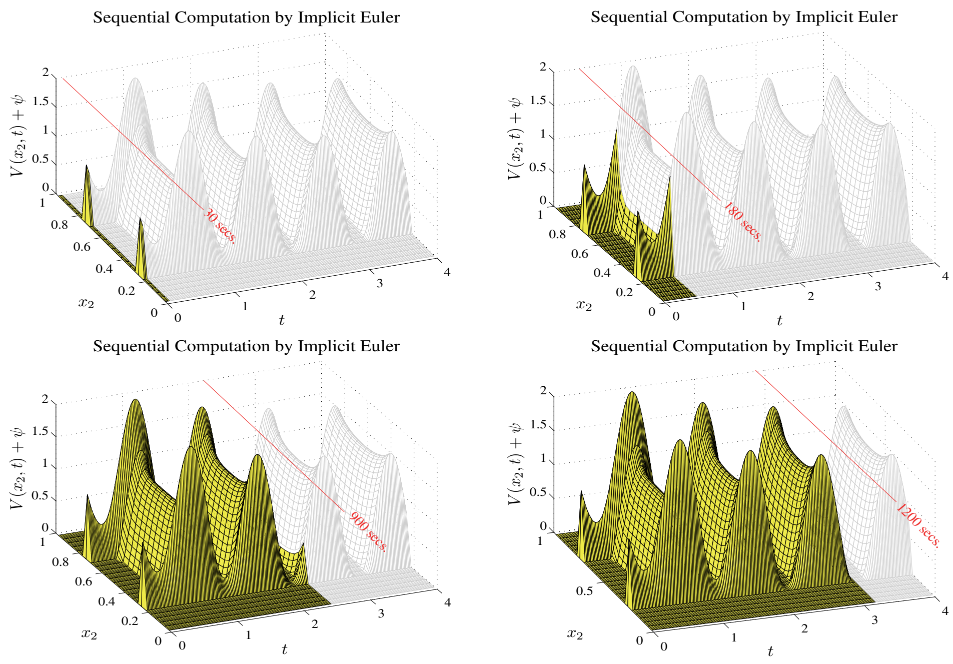

FIG. 10. *The profile of* $V(0,x_2,t)+\psi$ *generated by the implicit Euler method in the sequential mode, after* 30, 180, 900, *and* 1200 *seconds. Here,* $h=0.01$ *and* $\Delta x=0.025$.

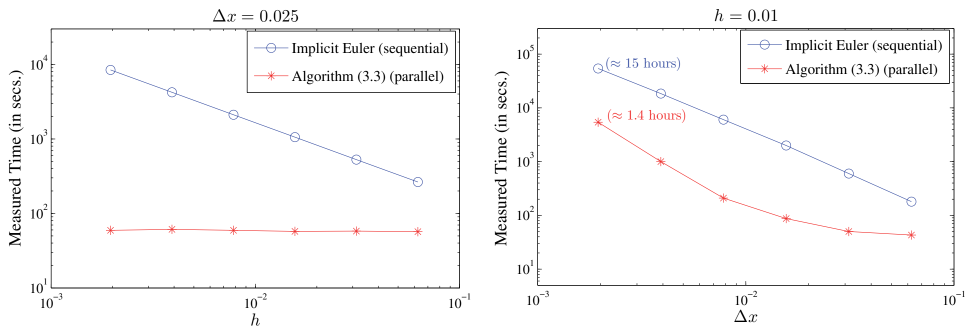

FIG. 11. *Comparison with respect to time for Algorithm* (3.3) *(with full parallelization) and the implicit Euler method implemented in the sequential mode.*

More comparisons with respect to the computation time between algorithm (3.3) and the implicit Euler method are shown in Figure 11. Precisely, in the left subfigure we show the measured time when Δx is fixed and h varies. We see that the time costed for algorithm (3.3) maintains a constant value around 60 seconds when h changes, while the time for implementing the implicit Euler method increases linearly (in the logarithmic scale) as h reduces. The latter is natural and the former is mainly because algorithm (3.3) behaves robustly with respect to the change of h when Δx is fixed; see Figure 7 on the left for evidence. In Figure 11 on the right, we fix $h=0.01$ and vary Δx from $\frac{1}{16}$ to $\frac{1}{512}$. In this case, both algorithm (3.3) and the implicit Euler

TABLE 1
GPU computation: percentage of the wall-clock time for four components of the Algorithm (3.3).

Case I: $\Delta x = 0.025$ is fixed and h varies (corresponds to Figure 11 on the left)						
	$h = \frac{1}{16}$	$h = \frac{1}{32}$	$h = \frac{1}{64}$	$h = \frac{1}{128}$	$h = \frac{1}{256}$	$h = \frac{1}{512}$
LCP solver	47.5%	48.8%	46.1%	42.8%	35.6%	32.0%
Linear algebra	35.7%	34.7%	39.5%	42.4%	50.7%	53.7%
Interpolation	4.5%	4%	4.3%	4.1%	4%	4.1%
Communication	12.3%	12.5%	10.1%	10.7%	9.7%	10.2%
Case II: $h = 0.01$ is fixed and Δx varies (corresponds to Figure 11 on the right)						
	$\Delta x = \frac{1}{16}$	$\Delta x = \frac{1}{32}$	$\Delta x = \frac{1}{64}$	$\Delta x = \frac{1}{128}$	$\Delta x = \frac{1}{256}$	$\Delta x = \frac{1}{512}$
LCP Solver	46.9%	46.1%	49.3%	40.1%	30.2%	22.8%
Linear Algebra	37.9%	37.3%	34.1%	40.2%	51.7%	56.8%
Interpolation	4.5%	4.7%	5.1%	6.3%	6.6%	7%
Communication	10.7%	11.9%	14.2%	13.4%	11.5%	13.4%

method need more time as Δx reduces.[6] However, the computation time of algorithm (3.3) is still significantly less than that of the implicit Euler method. For example, with $h = 0.01$ and $\Delta x = \frac{1}{512}$, the implicit Euler method needs 15 hours to finish the computation for $t \in (0, 4)$, while by using algorithm (3.3) with full parallelization it only takes 1.4 hours!

It would be interesting to show how the total wall-clock time of the GPU computation is split among the components of the algorithm, e.g., the LCP solver, the linear algebras, the linear interpolations (see section 2.3), and the communication cost between the CUDA cores. To this end, we show in Table 1 the percentage of the total wall-clock time for these four components of the iterative algorithm proposed in this paper.

5. Conclusions.

We have proposed an iterative algorithm for solving the DLCPs, which is based on the idea of functional iteration together with a novel treatment of the ODE system, namely, the numerical Laplace inversion. Different from the widely used time-stepping method, which requires that the step-size h should be sufficiently small such that the coefficient matrix M^h in the concerned LCP(q, M^h) satisfies some desired property, the proposed algorithm is concerned with a LCP(q, M) in which the matrix M^h never occurs. Moreover, the proposed algorithm is highly parallelizable in time, while the time-stepping method studied so far can be only implemented in the *sequential-in-time* mode. Convergence analysis for the new algorithm is performed for the P-matrix (and Z-matrix) LCS and generalization to broader cases is also discussed; see Remark 3.1. The estimate of the convergence rate given by Theorem 2.2 implies that the algorithm converges superlinearly and our numerical results indicate that the estimate is sharp and confirms numerical results well; see Figure 2. The numerical experiments conducted on the GPU-based parallel computation platform for the parabolic Signorini problem show that, with the same

[6]There are two things that increase the computation time of algorithm (3.3). First, as Δx reduces we need more iterations to reach the tolerance (4.8); see Figure 7 on the right for evidence. Second, as Δx reduces the sizes of the coefficient matrices, $\mathbf{A}$ and $\mathbf{M}$ in DLCP (4.5) become larger and naturally this increases the computation time for solving the involved linear algebraic equations and LCPs.

problematic/discretization configurations, the computation time of the proposed algorithm is less in several magnitudes than that of the widely used time-stepping method.

Our ongoing work is twofold. First, we try to generalize the current work to the case that the matrix A in the ODE system contains zero, imaginary, and/or unstable eigenvalues, by using the Cauchy–Goursat theorem as we mentioned in Remark 2.1. Second, we try to apply the Laplace inversion technique to dynamic complementarity problems in the nonlinear case:

$$\begin{aligned} &\dot{x}(t) = f(t, x(t), y(t)), \\ &\quad 0 \le y(t) \perp g(t, x(t), y(t)) \ge 0. \end{aligned} \tag{5.1}$$

To this end, we construct the following functional iterations:

$$\begin{aligned} 0 \le y^{k+1}(t) \perp g\left(t, x^k(t), y^{k+1}(t)\right) \ge 0, \\ \dot{x}^{k+1}(t) = Wx^{k+1}(t) + f\left(t, x^k(t), y^{k+1}(t)\right) - Wx^k(t), \end{aligned} \tag{5.2}$$

where $W \in \mathbb{R}^{m\times m}$ is a suitable matrix. Upon convergence, i.e., $k \to \infty$, it is clear that $\{x^\infty(t), y^\infty(t)\}$ is the solution of (5.1). Apparently, the ODE system in (5.2) is *linear* and therefore the Laplace inversion technique is applicable, by treating $f(t, x^k(t), y^{k+1}(t)) - Wx^k(t)$ as the source term.

Appendix A. Discretization of the parabolic Signorini equation (4.3)–(4.4). To discretize (4.3) by the centered finite difference formula, we first discretize the outer normal derivative $\partial_\nu V$ at $x_1 = 0$ by

$$\partial_\nu V(0, x_2, t) = \frac{V(x_{1,-1}, x_2, t) - V(x_{1,1}, x_2, t)}{2\Delta x} + O\left(\Delta x^2\right), \tag{A.1}$$

where $x_{1,-1} = -\Delta x$ denotes the extended grid point along the negatively horizontal direction. The quantity $V(x_{1,-1}, x_2)$ is therefore a "ghost" value because $x_{1,-1}$ lies outside the domain Ω. In the community of numerical partial differential equations, a widely used idea to fix such a ghost value is to discretize the governing equation on the boundary together with some suitable truncation. First, we assume that the diffusion equation also holds on the boundary $\mathcal{M}$. This assumption permits us to make the following discretization:

$$\begin{aligned} \frac{V(x_{1,-1}, x_2, t) - 2V(0, x_2, t) + V(x_{1,1}, x_2, t)}{\Delta x^2} &+ \partial^2_{x_2} V(0, x_2, t) \\ &= c^{-1}\partial_t V(0, x_2, t) + O\left(\Delta x^2\right). \end{aligned}$$

Then, for sufficiently small Δx we can expect $V(x_{1,-1}, x_2, t) = 2V(0, x_2, t) - \Delta x^2 \partial^2_{x_2} V(0, x_2, t) - V(x_{1,1}, x_2, t) + O(\Delta x^2)$. Substituting this into (A.1) gives the following discretization for the outer normal derivative on the semipermeable boundary $\mathcal{M}$:

$$\partial_\nu V(0, x_2, t) \approx \frac{2V(0, x_2, t) - 2V(x_{1,1}, x_2, t) - \Delta x^2 \partial^2_{x_2} V(0, x_2, t)}{2\Delta x}. \tag{A.2}$$

Let $\mathbf{V}(x_2, t) = (V(x_{1,1}, x_2, t), V(x_{1,2}, x_2, t), \ldots, V(x_{1,Q}, x_2, t))^\top$. Then, we have

$$\begin{cases} \partial_t \mathbf{V}(x_2, t) = cA\mathbf{V}(x_2, t) + c\partial^2_{x_2}\mathbf{V}(x_2, t) + \frac{c}{\Delta x^2}\mathbf{E}_1 V(0, x_2, t) + \frac{c}{\Delta x^2}\mathbf{E}_Q f(1, x_2, t), \\ 0 \le 2V(0, x_2, t) - \Delta x^2 \partial^2_{x_2} V(0, x_2, t) - 2\mathbf{E}_1^\top \mathbf{V}(x_2, t) \perp V(0, x_2, t) - \psi \ge 0, \end{cases} \tag{A.3}$$

where $\mathbf{E}_Q = (0, \ldots, 0, 1)^\top \in \mathbb{R}^Q$ and A is the tridiagonal matrix defined by (4.6).

Similarly, discretizing $\partial^2_{x_2}\mathbf{V}(x_2,t)$ in (A.3) by the centered finite difference formula gives

$$\begin{cases}\mathbf{X}'(t) = c(I\otimes A + A\otimes I)\mathbf{X}(t) + \frac{c}{\Delta x^2}(I\otimes \mathbf{E}_1)\mathbf{y}(t) + \mathbf{f}(t),\\ 0 \le (2I - \Delta x^2 A)\mathbf{y}(t) - 2(I\otimes \mathbf{E}_1^\top)\mathbf{X}(t) - \mathbf{g}(t) \perp \mathbf{y}(t) - \boldsymbol{\psi}(t) \ge 0,\end{cases} \tag{A.4}$$

where $\mathbf{X}(t) = (\mathbf{V}^\top(x_{2,1},t), \mathbf{V}^\top(x_{2,2},t), \ldots, \mathbf{V}^\top(x_{2,Q},t))^\top \in \mathbb{R}^{Q^2}$ and

$$\mathbf{y}(t) = (V(0,x_{2,1},t), V(0,x_{2,2},t), \ldots, V(0,x_{2,Q},t))^\top \in \mathbb{R}^Q,$$
$$\mathbf{f}(t) = \frac{c}{\Delta x^2}\left[(I\otimes \mathbf{E}_Q)\mathbf{f}_{x_1=1}(t) + (\mathbf{E}_1\otimes I)\mathbf{f}_{x_2=0}(t) + (\mathbf{E}_1\otimes I)\mathbf{f}_{x_2=1}(t)\right],$$
$$\mathbf{g}(t) = \mathbf{E}_1 f(0,0,t) + \mathbf{E}_Q f(0,1,t),\ \boldsymbol{\psi} = (\psi(x_{2,1},t), \psi(x_{2,2},t), \ldots, \psi(x_{2,Q},t))^\top.$$

The vector $\mathbf{f}_{x_1=1}(t)$ is defined by $\mathbf{f}_{x_1=1}(t) = (f(1,x_{2,1},t),\ldots,f(1,x_{2,Q},t))^\top$; similar definitions go to $\mathbf{f}_{x_2=0}(t)$ and $\mathbf{f}_{x_2=1}(t)$. To get the standard form of a DLCP, it remains to make a shift to $\mathbf{y}(t)$: by letting $\mathbf{Y}(t) = \mathbf{y}(t) - \boldsymbol{\psi}(t)$ we can rewrite (A.4) as (4.5).

Acknowledgments. The authors are very grateful to the anonymous referees for the careful reading of a preliminary version of the manuscript and their valuable suggestions and comments, which greatly improved the quality of this paper.

REFERENCES

[1] L. V. Ahlfors, *Complex Analysis,* 3rd ed., McGraw-Hill, New York, 1978.

[2] M. Anitescu and F. A. Potra, *Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems*, Nonlinear Dynam., 14 (1997), pp. 231–247.

[3] M. Anitescu and A. Tasora, *An iterative approach for cone complementarity problems for nonsmooth dynamics*, Comput. Optim. Appl., 47 (2010), pp. 207–235.

[4] X. J. Ban, J.-S. Pang, H. X. Liu, and R. Ma, *Modeling and solving continuous-time instantaneous dynamic user equilibria: A differential complementarity systems approach*, Transport. Res. B Meth., 46 (2012), pp. 389–408.

[5] B. Brogliato, *Some perspectives on the analysis and control of complementarity systems*, IEEE Trans. Automat. Control, 48 (2003), pp. 918–935.

[6] M. K. Camlibel, *Complementarity Methods in the Analysis of Piecewise Linear Dynamical Systems*, Ph.D. thesis, Center for Economic Research, Tilburg University, Tilburg, the Netherlands, 2001.

[7] M. K. Camlibel, W. P. M. H. Heemels, and J. M. Schumacher, *Consistency of a time stepping method for a class of piecewise-linear networks*, IEEE Trans. Circuits Syst. I. Fund. Theory Appl., 49 (2002), pp. 349–357.

[8] R. W. Cottle, J.-S. Pang, and R. E. Stone, *The Linear Complementarity Problem*, Academic Press, Boston, MA, 1992.

[9] X. Chen and S. Xiang, *Sparse solutions of linear complementarity problems*, Math. Program. Ser. A, 159 (2016), pp. 539–556.

[10] X. Chen and Z. Wang, *Computational error bounds for a differential linear variational inequality*, IMA. J. Numer. Anal., 32 (2012), pp. 957–982.

[11] X. Chen and S. Xiang, *Newton iterations in implicit time-stepping scheme for differential linear complementarity systems*, Math. Program. Ser. A, 138 (2013), pp. 579–606.

[12] X. Chen and Z. Wang, *Convergence of regularized time-stepping methods for differential variational inequalities*, SIAM J. Optim., 23 (2013), pp. 1647–1671.

[13] X. Chen and Z. Wang, *Differential variational inequality approach to dynamic games with shared constraints*, Math. Program. Ser. A, 146 (2014), pp. 379–408.

[14] D. Danielli, N. Garofalo, A. Petrosyan, and T. To, *Optimal regularity and the free boundary in the parabolic Signorini problem*, Mem. Amer. Math. Soc., 249 (2017), https://doi.org/10.1090/memo/1181.

[15] G. Duvaut and J.-L. Lions, *Inequalities in Mechanics and Physics*, Springer, New York, 1976.

[16] F. Facchinei and J.-S. Pang, *Finite-Dimensional Variational Inequalities and Complementarity Problems*, Springer, New York, 2003.

[17] M. C. Ferris and J.-S. Pang, *Engineering and economic applications of complementarity problems*, SIAM Rev., 39 (1997), pp. 669–713.

[18] S. Greenhalgh, V. Acary, and B. Brogliato, *On preserving dissipativity properties of linear complementarity dynamical systems with the θ-method*, Numer. Math., 125 (2013), pp. 601–637.

[19] L. Han, A. Tiwari, M. K. Camlibel, and J.-S. Pang, *Convergence of time-stepping schemes for passive and extended linear complementarity systems*, SIAM J. Numer. Anal., 47 (2009), pp. 3768–3796.

[20] T. Heyn, M. Anitescu, A. Tasora, and D. Negrut, *Using Krylov subspace and spectral methods for solving complementarity problems in many-body contact dynamics simulation*, Internat. J. Numer. Methods Engrg., 95 (2013), pp. 541–561.

[21] W. Mclean, I. H. Sloan, and V. Thomée, *Time discretization via Laplace transformation of an integro-differential equation of parabolic type*, Numer. Math., 102 (2006), pp. 497–522.

[22] W. Mclean and V. Thomée, *Time discretization of an evolution equation via Laplace transformation*, IMA J. Numer. Anal., 24 (2004), pp. 439–463.

[23] F. A. Potra, M. Anitescu, B. Gavrea, and J. Trinkle, *A linearly implicit trapezoidal method for integrating stiff multibody dynamics with contact, joints, and friction*, Internat. J. Numer. Methods Engrg., 66 (2006), pp. 1079–1124.

[24] J.-S. Pang, L. Han, G. Ramadurai, and S. Ukkusuri, *A continuous-time linear complementarity system for dynamic user equilibria in single bottleneck traffic flows*, Math. Program. Ser. A, 133 (2012), pp. 437–460.

[25] J.-S. Pang and D. Stewart, *Differential variational inequalities*, Math. Program. Ser. A, 113 (2008), pp. 345–424.

[26] A. Petrosyan, H. Shahgholian, and N. Uraltseva, *Regularity of Free Boundaries in Obstacle-Type Problems*, Vol. 136, AMS, Providence, RI, 2012.

[27] J. M. Schumacher, *Complementarity systems in optimization*, Math. Program. Ser. B, 101 (2004), pp. 263–296.

[28] D. Sheen, I. H. Sloan, and V. Thomée, *A parallel method for time discretization of parabolic problems based on contour integral representation and quadrature*, Math. Comp., 69 (1999), pp. 177–195.

[29] L. N. Trefethen and J. A. C. Weideman, *The exponentially convergent Trapezoidal rule*, SIAM Rev., 56 (2014), pp. 385–458.

[30] J. A. C. Weideman and L. N. Trefethen, *Parabolic and hyperbolic contours for computing the Bromwich integral*, Math. Comp., 76 (2007), pp. 1341–1356.

[31] S. L. Wu, *Laplace inversion for the solution of an abstract heat equation without the forward transform of the source term*, J. Numer. Math., to appear; also available online from https://doi.org/10.1515/jnma-2016-1014.

[32] Z. Wang and Y. X. Yuan, *Componentwise error bounds for linear complementarity problems*, IMA J. Numer. Anal., 31 (2011), pp. 348–357.