INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2008; **76**:1353–1378
Published online 19 June 2008 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nme.2361

Asynchronous multi-domain variational integrators for non-linear problems

Mark Gates, Karel Matouš*,† and Michael T. Heath

Computational Science and Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801, U.S.A.

SUMMARY

We develop an asynchronous time integration and coupling method with domain decomposition for linear and non-linear problems in mechanics. To ensure stability in the time integration and in coupling between domains, we use variational integrators with local Lagrange multipliers to enforce continuity at the domain interfaces. The asynchronous integrator lets each domain step with its own time step, using a smaller time step where required by stability and accuracy constraints and a larger time step where allowed. We show that in practice the time step is limited by accuracy requirements rather than by stability requirements.

Received 12 July 2007; Revised 22 February 2008; Accepted 5 March 2008

KEY WORDS: variational integrator; subcycling; multi-time; domain decomposition

1. INTRODUCTION

In time-dependent engineering applications, time integration is an important issue, especially when different multi-physics components are coupled together. Typically, the partial differential equations (PDEs) for these problems are solved by discretizing spatially using finite elements or finite volumes, and then integrating over time using a numerical ordinary differential equation (ODE) solver. Owing to their size, these simulations often require parallel computing, which can be effectively done using domain decomposition. Both the coupling between domains and the time integration must be handled with care to avoid numerical instability, since the stability and accuracy of the coupling is dictated not only by individual local limits, but also by the data transfer method between domains. We present a technique for integrating multi-domain non-linear problems, which is stable and easily parallelized.

*Correspondence to: Karel Matouš, Computational Science and Engineering, University of Illinois at Urbana-Champaign, 1304 W. Springfield Ave., Urbana, IL 61801, U.S.A.
†E-mail: matous@uiuc.edu

Contract/grant sponsor: U.S. Department of Energy; contract/grant number: B523819

For stability, the time step of explicit integrators is limited by the material properties and smallest element size, as well as by the coupling scheme. The stability and accuracy requirements for different domains may therefore necessitate different time steps. For example, a stiff domain may require small time steps for numerical stability, or a fluid domain may require small time steps to resolve turbulence accurately, whereas other domains may permit larger time steps. Typically, the integrator is constrained to use the smallest of these time steps. To relax this constraint, mixed methods use an explicit integrator in one domain and an implicit integrator in another, whereas subcycling or multi-time methods use different time steps for each domain. Belytschko and Mullen [1, 2] were the first to use a mixed explicit–implicit method with a nodal partition, whereas Hughes and Liu [3, 4] introduced a mixed method using an element partition. Belytschko *et al.* [5] introduced subcycling for first-order problems, and later extended it to non-integer time step ratios [6]. Neal and Belytschko [7] developed subcycling with non-integer ratios for second-order structural problems. Subcycling techniques for second-order problems are popular, but proofs of their stability have been elusive. Smolinski and Sleith [8] contrived an explicit subcycling algorithm that was proved stable [9], but is less accurate than other algorithms, whereas Daniel [10] showed that Neal and Belytschko's algorithm is 'statistically stable,' but unstable for certain time steps smaller than the expected stability limit. More recently, Combescure and Gravouil [11, 12] developed a FETI-like transient method that enforces continuity of velocities and is proved stable but dissipative for subcycling. Prakash and Hjelmstad [13] improved this method to be stable and non-dissipative for subcycling, but their method is based on binary trees for parallelism and superposition, rendering it less suitable for high-performance computing and non-linear problems.

For accurate long-term integration, it is highly desirable that the integrator preserve energy and momentum. ODE integrators that discretize the differential equation itself, such as Runge–Kutta methods, often artificially dissipate energy to achieve numerical stability. Variational integrators, in contrast, are derived by discretizing the Lagrangian and applying variational calculus, resulting in symplectic methods that exactly preserve momentum and do not dissipate energy, making them superior for long-term time integration. Variational integrators have been developed in Veselov [14, 15], Wendlandt and Marsden [16], and Marsden and West [17], among others. Kane *et al.* [18] show that the popular Newmark method [19] can also be derived variationally. Lew *et al.* [20, 21] developed an explicit asynchronous variational integrator, where each element has its own time step, with a parallel implementation in Kale and Lew [22]. We use a variational approach to take advantage of its favorable properties. For those unfamiliar with variational integrators, we review their derivation in Section 2.

For efficient parallel computing, domain decomposition is often used to create multiple domains that can be solved concurrently. For elliptic problems, Farhat and Roux [23] proposed the popular FETI method, which was subsequently extended to the transient problems [24, 25]. FETI enforces continuity between domains by adding constraints with Lagrange multipliers. Park *et al.* [26] developed a variant using local Lagrange multipliers that constrain domains to an intermediate interface rather than directly to each other. Such an intermediate interface is used in Park *et al.* [27] and Brezzi and Marini [28] to handle domains with non-matching meshes, as illustrated in Figure 1. We will consider the case of PDEs with non-matching meshes in future work, and thus introduce an intermediate interface here in anticipation.

We develop a method in Section 3 for integrating non-linear problems with domain decomposition, which allows each domain to use its own time step, with arbitrary time step ratios between domains. We linearly interpolate the interface motion and enforce continuity of displacements by local Lagrange multipliers. A displacement constraint seems most natural and avoids the

DOI: 10.1002/nme

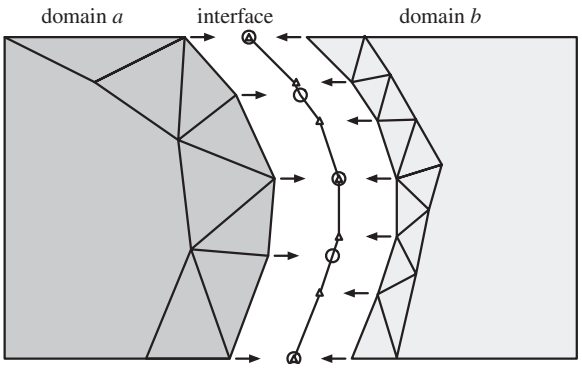

Figure 1. Domain decomposition with interface between domains. Arrows represent the local Lagrange multipliers constraining each domain to the interface.

discontinuities that develop between domains if continuity is imposed only for velocities. In the Hamiltonian framework, our integrator constrains both displacement and velocity. By treating the constraints variationally, we ensure stability in the domain coupling. The synchronous implicit midpoint version achieves unconditional stability, whereas with asynchronous steps it is conditionally stable, but we show that in practice the time step is limited by the accuracy requirement rather than by the stability requirement. For non-linear problems, the time step is also limited by the convergence region of Newton's method. We present examples in Section 4 involving a split single degree-of-freedom system and a mesh of springs that demonstrate the robustness and accuracy of our asynchronous method.

2. VARIATIONAL INTEGRATORS

2.1. Continuous Lagrangian and Hamiltonian frameworks

For completeness and clarity of presentation, we begin with a brief review of the governing equations for a mechanical system; see Hairer *et al.* [29] for more details. The derivation of variational integrators for computing the motion in the discrete setting will follow analogously.

Let Q be the configuration manifold of positions $\mathbf{q}$, with phase space TQ of positions and associated velocities $\dot{\mathbf{q}}$. The *Lagrangian* is a map on phase space, $L : TQ \to \mathbb{R}$, defined as kinetic minus potential energy. We will assume a mechanical system with Lagrangian

$$L(\mathbf{q}, \dot{\mathbf{q}}) = \tfrac{1}{2}\dot{\mathbf{q}}^{\mathrm{T}}\mathbf{M}\dot{\mathbf{q}} - V(\mathbf{q}) \tag{1}$$

where $\mathbf{M}$ is a symmetric positive-definite mass matrix and $V(\mathbf{q})$ is the potential energy. The *action* A is the integral of the Lagrangian from the initial time t_0 to the final time t_{F},

$$A = \int_{t_0}^{t_{\mathrm{F}}} L(\mathbf{q}, \dot{\mathbf{q}})\, \mathrm{d}t \tag{2}$$

Hamilton's principle states that the motion of a system makes the action A stationary with respect to all possible paths $\mathbf{q}(t)$ connecting the fixed initial position $\mathbf{q}_0$ and final position $\mathbf{q}_{\mathrm{F}}$. Consider a

DOI: 10.1002/nme

variation $\mathbf{q}+\varepsilon\delta\mathbf{q}$ with boundary conditions $\delta\mathbf{q}(t_0)=\delta\mathbf{q}(t_\mathrm{F})=\mathbf{0}$. Applying integration by parts and the boundary conditions yields

$$\delta A=\frac{\partial}{\partial\varepsilon}A(\mathbf{q}+\varepsilon\delta\mathbf{q})\bigg|_{\varepsilon=0}=\int_{t_0}^{t_\mathrm{F}}\left(\frac{\partial L}{\partial\mathbf{q}}-\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial\dot{\mathbf{q}}}\right)\cdot\delta\mathbf{q}\,\mathrm{d}t=0 \tag{3}$$

Since this holds for any variation $\delta\mathbf{q}$, we obtain the familiar *Euler–Lagrange* differential equation

$$\frac{\partial L}{\partial\mathbf{q}}-\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial\dot{\mathbf{q}}}=\mathbf{0} \tag{4}$$

Turning now to the Hamiltonian framework, we define the momentum $\mathbf{p}$ by the *Legendre transform* $\mathbb{F}L$, which maps the position–velocity space TQ to the position–momentum space T^*Q by

$$\mathbb{F}L:(\mathbf{q},\dot{\mathbf{q}})\mapsto(\mathbf{q},\mathbf{p})=\left(\mathbf{q},\frac{\partial L}{\partial\dot{\mathbf{q}}}\right) \tag{5}$$

The *Hamiltonian*, given by $H=\mathbf{p}^\mathrm{T}\mathbf{q}-L(\mathbf{q},\dot{\mathbf{q}})$, then defines the total energy of the system.

2.2. Discrete Lagrangian and Hamiltonian frameworks

Our derivation of discrete variational integrators follows that of Veselov [14], Wendlandt and Marsden [16], and Marsden and West [17]. We define the discrete state space to be $Q\times Q$, representing positions $(\mathbf{q}_n,\mathbf{q}_{n+1})$ at consecutive time steps t_n,t_{n+1}. The *discrete Lagrangian* is a map, $L_\mathrm{d}:Q\times Q\to\mathbb{R}$, that approximates the Lagrangian (1) over one time step

$$L_\mathrm{d}(\mathbf{q}_n,\mathbf{q}_{n+1})\approx\int_{t_n}^{t_{n+1}}L(\mathbf{q},\dot{\mathbf{q}})\,\mathrm{d}t \tag{6}$$

and the *discrete action sum* is a map, $A_\mathrm{d}:Q^{N+1}\to\mathbb{R}$, that approximates the action integral (2) over N time steps from the initial time t_0 to the final time t_N,

$$A_\mathrm{d}(\mathbf{q}_0,\ldots,\mathbf{q}_N)=\sum_{n=0}^{N-1}L_\mathrm{d}(\mathbf{q}_n,\mathbf{q}_{n+1})\approx\int_{t_0}^{t_N}L(\mathbf{q},\dot{\mathbf{q}})\,\mathrm{d}t \tag{7}$$

Similar to the continuous case, the *discrete Hamilton principle* states that the motion makes the discrete action sum stationary with respect to all paths $\mathbf{q}_0,\ldots,\mathbf{q}_N$ with fixed endpoints $\mathbf{q}_0$ and $\mathbf{q}_N$. Consider a variation $\mathbf{q}_n+\varepsilon\delta\mathbf{q}_n$ for $n=0,\ldots,N$ with boundary conditions $\delta\mathbf{q}_0=\mathbf{0}$ and $\delta\mathbf{q}_N=\mathbf{0}$, which yields

$$\begin{aligned}\delta A_\mathrm{d}=\frac{\partial}{\partial\varepsilon}A_\mathrm{d}(\{\mathbf{q}_n+\varepsilon\delta\mathbf{q}_n\})\bigg|_{\varepsilon=0}&=\sum_{n=0}^{N-1}\frac{\partial}{\partial\mathbf{q}_n}L_\mathrm{d}(\mathbf{q}_n,\mathbf{q}_{n+1})\cdot\delta\mathbf{q}_n+\frac{\partial}{\partial\mathbf{q}_{n+1}}L_\mathrm{d}(\mathbf{q}_n,\mathbf{q}_{n+1})\cdot\delta\mathbf{q}_{n+1}\\&=\sum_{n=1}^{N-1}\left(\frac{\partial}{\partial\mathbf{q}_n}L_\mathrm{d}(\mathbf{q}_{n-1},\mathbf{q}_n)+\frac{\partial}{\partial\mathbf{q}_n}L_\mathrm{d}(\mathbf{q}_n,\mathbf{q}_{n+1})\right)\cdot\delta\mathbf{q}_n=0\end{aligned}$$

DOI: 10.1002/nme

The last step was accomplished by rearranging the sum and canceling the first and last terms due to boundary conditions. As in the continuous case, since $\delta\mathbf{q}_n$ is arbitrary, we obtain the *discrete Euler–Lagrange equations*,

$$\frac{\partial}{\partial \mathbf{q}_n} L_\mathrm{d}(\mathbf{q}_{n-1}, \mathbf{q}_n) + \frac{\partial}{\partial \mathbf{q}_n} L_\mathrm{d}(\mathbf{q}_n, \mathbf{q}_{n+1}) = \mathbf{0} \tag{8}$$

which are solved for each $\mathbf{q}_{n+1}$ to obtain a recurrence relation. The Lagrangian integrator given by (8) is a two-step method that updates position.

To formulate a Hamiltonian integrator, we define the discrete momentum $\mathbf{p}_n$ by the *discrete Legendre transform*, which maps the discrete state space $Q \times Q$ to the position–momentum space T^*Q by

$$\mathbb{F}^- L_\mathrm{d} : (\mathbf{q}_n, \mathbf{q}_{n+1}) \mapsto (\mathbf{q}_n, \mathbf{p}_n) = \left(\mathbf{q}_n, -\frac{\partial}{\partial \mathbf{q}_n} L_\mathrm{d}(\mathbf{q}_n, \mathbf{q}_{n+1}) \right) \tag{9}$$

and gives an implicit equation for the next position $\mathbf{q}_{n+1}$,

$$\mathbf{p}_n = -\frac{\partial}{\partial \mathbf{q}_n} L_\mathrm{d}(\mathbf{q}_n, \mathbf{q}_{n+1}) \tag{10}$$

Note that $\mathbf{p}_n$ is simply the negative of the second term in the discrete Euler–Lagrange equations (8). Written at $n+1$, this implies that the first term in the discrete Euler–Lagrange equations gives an explicit equation for the next momentum,

$$\mathbf{p}_{n+1} = \frac{\partial}{\partial \mathbf{q}_{n+1}} L_\mathrm{d}(\mathbf{q}_n, \mathbf{q}_{n+1}) \tag{11}$$

and the alternate discrete Legendre transform

$$\mathbb{F}^+ L_\mathrm{d} : (\mathbf{q}_n, \mathbf{q}_{n+1}) \mapsto (\mathbf{q}_n, \mathbf{p}_{n+1}) = \left(\mathbf{q}_n, \frac{\partial}{\partial \mathbf{q}_{n+1}} L_\mathrm{d}(\mathbf{q}_n, \mathbf{q}_{n+1}) \right) \tag{12}$$

Equations (10) and (11) together result in a single-step method that updates both position and momentum.

The momentum $\mathbf{p}_n$ acts to store known quantities in the discrete Euler–Lagrange equations; hence, the implementation of Lagrangian and Hamiltonian integrators is essentially identical. The main advantage of the Hamiltonian integrator is a simpler initialization using $\mathbf{p}_0 = \mathbf{M}\mathbf{v}_0$, instead of requiring the first two positions $\mathbf{q}_0, \mathbf{q}_1$, as with the Lagrangian integrator. Adding constraints in Section 3 will make the distinction between Lagrangian and Hamiltonian frameworks more important.

2.3. Examples of variational integrators

Using different quadrature rules for the discrete Lagrangian (6) yields different variational integrators. Consider the symmetric generalized midpoint approximation

$$L_\mathrm{d}^{\mathrm{sym},\alpha}(\mathbf{q}_n, \mathbf{q}_{n+1}) = \frac{1}{2}\Delta t\, L\left(\mathbf{q}_{n+\alpha}, \frac{\mathbf{q}_{n+1} - \mathbf{q}_n}{\Delta t} \right) + \frac{1}{2}\Delta t\, L\left(\mathbf{q}_{n+1-\alpha}, \frac{\mathbf{q}_{n+1} - \mathbf{q}_n}{\Delta t} \right) \tag{13}$$

DOI: 10.1002/nme

which yields second-order accurate integrators in both the Lagrangian and the Hamiltonian frameworks. In the Lagrangian framework, the discrete Euler–Lagrange equations (8) become

$$\frac{1}{2}\Delta t\left(-\alpha\nabla V(\mathbf{q}_{n-1+\alpha})-(1-\alpha)\nabla V(\mathbf{q}_{n-\alpha})-(1-\alpha)\nabla V(\mathbf{q}_{n+\alpha})-\alpha\nabla V(\mathbf{q}_{n+1-\alpha})\right)$$
$$-\mathbf{M}\left(\frac{\mathbf{q}_{n+1}-2\mathbf{q}_n+\mathbf{q}_{n-1}}{\Delta t}\right)=\mathbf{0} \tag{14}$$

with the time step $\Delta t=t_{n+1}-t_n$ and abbreviation $\mathbf{q}_{n+\alpha}=(1-\alpha)\mathbf{q}_n+\alpha\mathbf{q}_{n+1}$. For $\alpha=0$ or 1, this reduces to the explicit Newmark method [19], which is equivalent to the Verlet method [30],

$$\mathbf{M}\left(\frac{\mathbf{q}_{n+1}-2\mathbf{q}_n+\mathbf{q}_{n-1}}{\Delta t^2}\right)=-\nabla V(\mathbf{q}_n) \tag{15}$$

With the same $L_{\mathrm{d}}^{\mathrm{sym},\alpha}$, the Hamiltonian integrator defined by (10) and (11) becomes

$$\mathbf{p}_n=\frac{1}{2}\Delta t(1-\alpha)\nabla V(\mathbf{q}_{n+\alpha})+\frac{1}{2}\Delta t\alpha\nabla V(\mathbf{q}_{n+1-\alpha})+\mathbf{M}\left(\frac{\mathbf{q}_{n+1}-\mathbf{q}_n}{\Delta t}\right) \tag{16}$$

$$\mathbf{p}_{n+1}=-\frac{1}{2}\Delta t\alpha\nabla V(\mathbf{q}_{n+\alpha})-\frac{1}{2}\Delta t(1-\alpha)\nabla V(\mathbf{q}_{n+1-\alpha})+\mathbf{M}\left(\frac{\mathbf{q}_{n+1}-\mathbf{q}_n}{\Delta t}\right) \tag{17}$$

For $\alpha=0$ or 1, we recover the velocity Verlet method.

Since they both solve the discrete Euler–Lagrange equations (8), the Lagrangian and Hamiltonian integrators are equivalent and will compute identical positions. In the Lagrangian framework, we can use the simpler generalized midpoint rule

$$L_{\mathrm{d}}^{\alpha}=\Delta t\,L\left(\mathbf{q}_{n+\alpha},\frac{\mathbf{q}_{n+1}-\mathbf{q}_n}{\Delta t}\right) \tag{18}$$

to derive identical integrators as from $L_{\mathrm{d}}^{\mathrm{sym},\alpha}$ for $\alpha=0$, $\frac{1}{2}$, and 1. However, in the Hamiltonian framework, L_{d}^{α} yields the symplectic Euler methods for $\alpha=0$ and 1, which are only *weakly equivalent* to the $L_{\mathrm{d}}^{\mathrm{sym},0}$ integrator in that all three compute the same positions $\mathbf{q}_n$ but different momenta $\mathbf{p}_n$. In particular, with an initial momentum $\mathbf{p}_0=\mathbf{M}\mathbf{v}_0$, the symplectic Euler methods are only first-order accurate. If the initial momentum is contrived for the L_{d}^{0} integrator so that it agrees with the $L_{\mathrm{d}}^{\mathrm{sym},0}$ integrator on the first two positions $\mathbf{q}_0$ and $\mathbf{q}_1$, then they will agree on the remaining positions and differ only on momenta.

3. DOMAIN DECOMPOSITION

We are interested in using variational integrators to solve problems that have been decomposed into multiple domains to be solved in parallel. The continuity of positions at the domain interfaces is enforced by the Lagrange multipliers. We employ the method of local Lagrange multipliers, introduced by Park *et al.* [26], where each domain is constrained to an intermediate interface with position $\boldsymbol{\psi}$, as shown in Figure 2. This method eliminates over-constraints where more than two subdomains meet, and is advantageous for non-matching meshes, as shown in Brezzi and Marini [28] and Park *et al.* [27].

DOI: 10.1002/nme

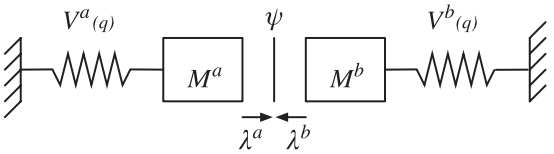

Figure 2. Split single degree-of-freedom system with masses constrained together.

For simplicity, we will consider two domains, labeled with superscripts a and b. For domain a, let the operator $\mathbf{B}^a$ extract the interface degrees of freedom from $\mathbf{q}^a$, and the operator $\mathbf{C}^a$ extract the corresponding degrees of freedom from $\boldsymbol{\psi}$ such that the constraint

$$\boldsymbol{\phi}^a = \mathbf{B}^a \mathbf{q}^a - \mathbf{C}^a \boldsymbol{\psi} = \mathbf{0} \tag{19}$$

enforces continuity between the domain and the interface. Domain b has similar operators $\mathbf{B}^b$, $\mathbf{C}^b$, and constraint $\boldsymbol{\phi}^b$. A standard Lagrangian for mechanical systems (1) is used for each domain. We start with synchronous integrators, where both domains take the same time step, and then develop asynchronous integrators, where each domain has its own time step.

3.1. Synchronous integrators

3.1.1. Lagrangian framework. To formulate a variational integrator with constraints, our treatment follows that of Lew *et al.* [21, p. 165] and Marsden and West [17, p. 439]. For the constraint $\boldsymbol{\phi}(\mathbf{q}) = \mathbf{0}$, consider the *augmented discrete Lagrangian*

$$\hat{L}_\mathrm{d}(\mathbf{q}_n, \mathbf{q}_{n+1}, \boldsymbol{\lambda}_{n+1}) = L_\mathrm{d}(\mathbf{q}_n, \mathbf{q}_{n+1}) + \boldsymbol{\phi}(\mathbf{q}_{n+1})^\mathrm{T} \boldsymbol{\lambda}_{n+1} \tag{20}$$

where $\boldsymbol{\lambda}$ is a vector of Lagrange multipliers. With the augmented action sum, $\hat{A}_\mathrm{d} = \sum_{n=0}^{N-1} \hat{L}_\mathrm{d}(\mathbf{q}_n, \mathbf{q}_{n+1})$, the discrete Euler–Lagrange equations (8) become

$$\frac{\partial \hat{A}_\mathrm{d}}{\partial \mathbf{q}_n} = \frac{\partial}{\partial \mathbf{q}_n} \hat{L}_\mathrm{d}(\mathbf{q}_{n-1}, \mathbf{q}_n) + \frac{\partial}{\partial \mathbf{q}_n} \hat{L}_\mathrm{d}(\mathbf{q}_n, \mathbf{q}_{n+1}) + \mathbf{J}_\phi(\mathbf{q}_n)^\mathrm{T} \boldsymbol{\lambda}_n = \mathbf{0} \tag{21a}$$

$$\frac{\partial \hat{A}_\mathrm{d}}{\partial \boldsymbol{\lambda}_{n+1}} = \boldsymbol{\phi}(\mathbf{q}_{n+1}) = \mathbf{0} \tag{21b}$$

where $\mathbf{J}_\phi$ is the Jacobian matrix of $\boldsymbol{\phi}$. This system is solved for the next position $\mathbf{q}_{n+1}$ and the Lagrange multipliers at the current time step $\boldsymbol{\lambda}_n$.

Consider our model problem with the two domains constrained to a common interface using local Lagrange multipliers $\boldsymbol{\lambda}^a$ and $\boldsymbol{\lambda}^b$, as shown in Figure 2. The augmented discrete Lagrangian (20) yields

$$\hat{L}_\mathrm{d}(\mathbf{q}_n, \mathbf{q}_{n+1}) = L_\mathrm{d}^a(\mathbf{q}_n^a, \mathbf{q}_{n+1}^a) + L_\mathrm{d}^b(\mathbf{q}_n^b, \mathbf{q}_{n+1}^b) + \boldsymbol{\phi}^a(\mathbf{q}_{n+1}^a)^\mathrm{T} \boldsymbol{\lambda}_{n+1}^a + \boldsymbol{\phi}^b(\mathbf{q}_{n+1}^b)^\mathrm{T} \boldsymbol{\lambda}_{n+1}^b \tag{22}$$

where L_d^a and L_d^b are the discrete Lagrangians for each subdomain. For more than two subdomains, we can simply add additional terms to $\hat{L}_\mathrm{d}$. Using the generalized midpoint rule L_d^α (18) for L_d^a

DOI: 10.1002/nme

and L_{d}^b, the constrained discrete Euler–Lagrange equations (21) for domain a become

$$\frac{\partial \hat{A}_{\mathrm{d}}}{\partial \mathbf{q}_n^a} = -\alpha \Delta t \nabla V^a(\mathbf{q}_{n-1+\alpha}^a) - (1-\alpha)\Delta t \nabla V^a(\mathbf{q}_{n+\alpha}^a)$$

$$-\mathbf{M}^a(\mathbf{q}_{n+1}^a - 2\mathbf{q}_n^a + \mathbf{q}_{n-1}^a)/\Delta t + \mathbf{B}^{a\mathrm{T}}\boldsymbol{\lambda}_n^a = \mathbf{0} \tag{23a}$$

$$\boldsymbol{\phi}^a = \mathbf{B}^a \mathbf{q}_{n+1}^a - \mathbf{C}^a \boldsymbol{\psi}_{n+1} = \mathbf{0} \tag{23b}$$

and similarly for domain b, plus

$$\frac{\partial \hat{A}_{\mathrm{d}}}{\partial \boldsymbol{\psi}_n} = -\mathbf{C}^{a\mathrm{T}}\boldsymbol{\lambda}_n^a - \mathbf{C}^{b\mathrm{T}}\boldsymbol{\lambda}_n^b = \mathbf{0} \tag{23c}$$

for the interface. For $\alpha = 0$ or 1 this is the explicit SHAKE method [31], which is the constrained version of Verlet. For other α it is implicit and must be solved using an iterative technique such as Newton's method,

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \mathbf{J}_g(\mathbf{x}^{(i)})^{-1}\mathbf{g}(\mathbf{x}^{(i)}) \tag{24}$$

where i is the iteration index and $\mathbf{x} = [\mathbf{q}^a\ \boldsymbol{\lambda}^a\ \mathbf{q}^b\ \boldsymbol{\lambda}^b\ \boldsymbol{\psi}]^{\mathrm{T}}$ is the solution vector. The residual function $\mathbf{g}$ is given by (23), with the Jacobian

$$\mathbf{J}_g = \left[\begin{array}{cc|cc|c} \mathbf{F}^a & \mathbf{B}^{a\mathrm{T}} & & & \mathbf{0} \\ \mathbf{B}^a & \mathbf{0} & & & -\mathbf{C}^a \\ \hline & & \mathbf{F}^b & \mathbf{B}^{b\mathrm{T}} & \mathbf{0} \\ & & \mathbf{B}^b & \mathbf{0} & -\mathbf{C}^b \\ \hline \mathbf{0} & -\mathbf{C}^{a\mathrm{T}} & \mathbf{0} & -\mathbf{C}^{b\mathrm{T}} & \mathbf{0} \end{array}\right] \tag{25}$$

where

$$\mathbf{F}^a = \alpha(1-\alpha)\Delta t \mathbf{J}_f^a(\mathbf{q}_{n+\alpha}^a) - \mathbf{M}^a/\Delta t$$

$$\mathbf{F}^b = \alpha(1-\alpha)\Delta t \mathbf{J}_f^b(\mathbf{q}_{n+\alpha}^b) - \mathbf{M}^b/\Delta t$$

and $\mathbf{J}_f^a$ is the Jacobian of the force $\mathbf{f}^a(\mathbf{q}) = -\nabla V^a(\mathbf{q})$. The matrix $\mathbf{J}_g$ has a block-bordered form amenable to parallel computation, with each block assigned to a different processor. For more than two domains, each domain will have a block on the diagonal, and the interface will have entries along the bottom and right-hand side, continuing the arrow form of (25). These types of matrices can be efficiently solved in parallel by using a direct or iterative method to solve the Schur complement for the interface unknowns, then solving each domain independently (see Chan and Mathew [32] for further details).

3.1.2. Hamiltonian framework. Now turning to the Hamiltonian framework, to add a constraint it is also necessary to keep the momentum on the constrained phase space T^*N, since for the augmented Hamiltonian scheme, the evolution of $\boldsymbol{\lambda}$ is not uniquely defined at t_0. Following Marsden

DOI: 10.1002/nme

and West [17, p. 435], let $\eta: T^*N \to T^*Q$ be the symplectic embedding given by

$$\eta = \left\{ (\mathbf{q}, \mathbf{p}) \in T^*Q : \boldsymbol{\phi}(\mathbf{q}) = \mathbf{0} \text{ and } \mathbf{J}_\phi(\mathbf{q}) \frac{\partial H}{\partial \mathbf{p}}(\mathbf{q}, \mathbf{p}) = \mathbf{0} \right\} \tag{26}$$

The extra constraint $\mathbf{J}_\phi \partial H / \partial \mathbf{p} = \mathbf{0}$ will enforce the continuity of velocity, in addition to continuity of positions. Adding a second Lagrange multiplier $\boldsymbol{\mu}_n$ to enforce this extra constraint yields the constrained Hamiltonian integrator

$$\mathbf{p}_n = -\frac{\partial}{\partial \mathbf{q}_n} L_\mathrm{d}(\mathbf{q}_n, \mathbf{q}_{n+1}) - \mathbf{J}_\phi(\mathbf{q}_n)^\mathrm{T} \boldsymbol{\lambda}_n \tag{27a}$$

$$\phi(\mathbf{q}_{n+1}) = \mathbf{0} \tag{27b}$$

which is solved implicitly for $\mathbf{q}_{n+1}$ and $\boldsymbol{\lambda}_n$, and

$$\mathbf{p}_{n+1} = \frac{\partial}{\partial \mathbf{q}_{n+1}} L_\mathrm{d}(\mathbf{q}_n, \mathbf{q}_{n+1}) + \mathbf{J}_\phi(\mathbf{q}_{n+1})^\mathrm{T} \boldsymbol{\mu}_n \tag{28a}$$

$$\mathbf{J}_\phi(\mathbf{q}_{n+1}) \frac{\partial}{\partial \mathbf{p}} H(\mathbf{q}_{n+1}, \mathbf{p}_{n+1}) = \mathbf{0} \tag{28b}$$

which is solved explicitly for $\mathbf{p}_{n+1}$ and $\boldsymbol{\mu}_n$.

To maintain second-order accuracy, we use the symmetric generalized midpoint rule $L_\mathrm{d}^{\mathrm{sym},\alpha}$ (13) for L_d^a and L_d^b. Substituting the augmented discrete Lagrangian (22) into the constrained momentum equations (27) yields the system of equations

$$\mathbf{p}_n^a = \tfrac{1}{2}(1-\alpha)\Delta t \nabla V^a(\mathbf{q}_{n+\alpha}^a) + \tfrac{1}{2}\alpha \Delta t \nabla V^a(\mathbf{q}_{n+1-\alpha}^a) + \mathbf{M}^a(\mathbf{q}_{n+1}^a - \mathbf{q}_n^a)/\Delta t - \mathbf{B}^{a\mathrm{T}} \boldsymbol{\lambda}_n^a \tag{29a}$$

$$\boldsymbol{\phi}^a = \mathbf{B}^a \mathbf{q}_{n+1}^a - \mathbf{C}^a \boldsymbol{\psi}_{n+1} = \mathbf{0} \tag{29b}$$

for domain a, and similarly for domain b, and

$$\frac{\partial \hat{L}_\mathrm{d}}{\partial \boldsymbol{\psi}_n} = -\mathbf{C}^{a\mathrm{T}} \boldsymbol{\lambda}_n^a - \mathbf{C}^{b\mathrm{T}} \boldsymbol{\lambda}_n^b = \mathbf{0} \tag{29c}$$

for the interface. Similar to the Lagrangian framework, for $\alpha = 0$ or 1 this is the explicit RATTLE method [33], which is the constrained version of the velocity Verlet, while for other α we can solve it using an iterative method. The Jacobian is again a block-bordered matrix of the same form as (25), with

$$\mathbf{F}^a = \tfrac{1}{2}\alpha(1-\alpha)\Delta t J_f^a(\mathbf{q}_{n+\alpha}^a) + \tfrac{1}{2}\alpha(1-\alpha)\Delta t J_f^a(\mathbf{q}_{n+1-\alpha}^a) - \mathbf{M}^a/\Delta t$$

$$\mathbf{F}^b = \tfrac{1}{2}\alpha(1-\alpha)\Delta t J_f^b(\mathbf{q}_{n+\alpha}^b) + \tfrac{1}{2}\alpha(1-\alpha)\Delta t J_f^b(\mathbf{q}_{n+1-\alpha}^b) - \mathbf{M}^b/\Delta t$$

Once the positions have been computed using (29), we must compute the new momenta. For our purposes, the extra constraint $\mathbf{J}_\phi \partial H / \partial \mathbf{p} = \mathbf{0}$ on the momentum enforces continuity of velocity. Because the common interface has no mass, we enforce the continuity of velocity directly between

DOI: 10.1002/nme

domains using

$$\mathbf{J}_{\phi^a}\frac{\partial H}{\partial \mathbf{p}^a}-\mathbf{J}_{\phi^b}\frac{\partial H}{\partial \mathbf{p}^b}=\mathbf{B}^a(\mathbf{M}^a)^{-1}\mathbf{p}^a_{n+1}-\mathbf{B}^b(\mathbf{M}^b)^{-1}\mathbf{p}^b_{n+1}=\mathbf{0} \tag{30}$$

Formally, this is the same condition for momentum that one would get without the interface $\boldsymbol{\psi}$, by substituting the constraint $\boldsymbol{\phi}=\mathbf{B}^a\mathbf{q}^a_n-\mathbf{B}^b\mathbf{q}^b_n=\mathbf{0}$ into (28b). With this condition, the constrained momentum equations (28) for $\mathbf{p}_{n+1}$ and $\boldsymbol{\mu}_n$ yield the system

$$\mathbf{p}^a_{n+1}=-\tfrac{1}{2}\alpha\Delta t\nabla V^a(\mathbf{q}^a_{n+\alpha})-\tfrac{1}{2}(1-\alpha)\Delta t\nabla V^a(\mathbf{q}^a_{n+1-\alpha})+\mathbf{M}^a(\mathbf{q}^a_{n+1}-\mathbf{q}^a_n)/\Delta t+{\mathbf{B}^a}^{\mathrm{T}}\boldsymbol{\mu}_n \tag{31a}$$

$$\mathbf{p}^b_{n+1}=-\tfrac{1}{2}\alpha\Delta t\nabla V^b(\mathbf{q}^b_{n+\alpha})-\tfrac{1}{2}(1-\alpha)\Delta t\nabla V^b(\mathbf{q}^b_{n+1-\alpha})+\mathbf{M}^b(\mathbf{q}^b_{n+1}-\mathbf{q}^b_n)/\Delta t-{\mathbf{B}^b}^{\mathrm{T}}\boldsymbol{\mu}_n \tag{31b}$$

$$\mathbf{B}^a(\mathbf{M}^a)^{-1}\mathbf{p}^a_{n+1}-\mathbf{B}^b(\mathbf{M}^b)^{-1}\mathbf{p}^b_{n+1}=\mathbf{0} \tag{31c}$$

Note that (31) can always be solved explicitly, even when the force $-\nabla V$ is non-linear, because $\mathbf{q}_{n+1}$ is already known.

3.2. Asynchronous integrators

Since the stability and accuracy requirements for different domains may necessitate different time step sizes, we develop an asynchronous integrator where each domain steps with its own step size. The integrator takes s^a substeps for domain a and s^b substeps for domain b to make one system time step Δt, as shown in Figure 3. For intermediate steps, the position of each domain's interface is constrained to match the linear interpolation of the position $\boldsymbol{\psi}$ of the common interface. We denote time steps of domain a by the subscript i, time steps of domain b by the subscript j, and system time steps by the subscript n. A different variational integrator may be used for each domain, for example, $\alpha^a=\frac{1}{2}$ for domain a and $\alpha^b=0$ for domain b, or *vice versa*. Unless otherwise stated, we use the same integrator for each domain, with $\alpha=\alpha^a=\alpha^b$.

3.2.1. The Lagrangian framework. The discrete Lagrangian for one system time step, augmented with constraints, is given by

$$\begin{aligned}\hat{L}_{\mathrm{d}}=&\sum_{k=0}^{s^a-1}L^a_{\mathrm{d}}(\mathbf{q}_{i+k},\mathbf{q}_{i+k+1})+(\mathbf{B}^a\mathbf{q}_{i+k+1}-\mathbf{C}^a\boldsymbol{\psi}_{i+k+1})^{\mathrm{T}}\boldsymbol{\lambda}_{i+k+1}\\&+\sum_{k=0}^{s^b-1}L^b_{\mathrm{d}}(\mathbf{q}_{j+k},\mathbf{q}_{j+k+1})+(\mathbf{B}^b\mathbf{q}_{j+k+1}-\mathbf{C}^b\boldsymbol{\psi}_{j+k+1})^{\mathrm{T}}\boldsymbol{\lambda}_{j+k+1}\end{aligned} \tag{32}$$

where L^a_{d} and L^b_{d} are the discrete Lagrangians for domains a and b, respectively, and the position of the common interface $\boldsymbol{\psi}$ is linearly interpolated by

$$\boldsymbol{\psi}_{i+k}=\left(1-\frac{k}{s^a}\right)\boldsymbol{\psi}_n+\left(\frac{k}{s^a}\right)\boldsymbol{\psi}_{n+1} \tag{33}$$

$$\boldsymbol{\psi}_{j+k}=\left(1-\frac{k}{s^b}\right)\boldsymbol{\psi}_n+\left(\frac{k}{s^b}\right)\boldsymbol{\psi}_{n+1} \tag{34}$$

DOI: 10.1002/nme

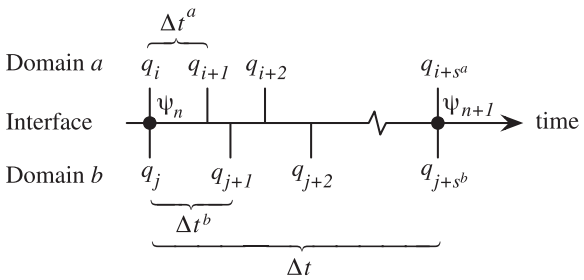

Figure 3. Substeps of system time step.

Each domain's substep size is proportional to the system time step, $\Delta t^a = \Delta t / s^a$, and the appropriate substep size should be used in the discrete Lagrangians L_{d}^a and L_{d}^b. Using the generalized midpoint rule L_{d}^{α} (18) for L_{d}^a and L_{d}^b, the constrained discrete Euler–Lagrange equations (21) yield s^a equations for the positions of domain a,

$$\frac{\partial \hat{A}_{\mathrm{d}}}{\partial \mathbf{q}_i} = -\alpha \Delta t^a \nabla V^a(\mathbf{q}_{i-1+\alpha}) - (1-\alpha)\Delta t^a \nabla V^a(\mathbf{q}_{i+\alpha}) \\ -\mathbf{M}^a(\mathbf{q}_{i+1} - 2\mathbf{q}_i + \mathbf{q}_{i-1})/\Delta t^a + \mathbf{B}^{a\mathrm{T}}\boldsymbol{\lambda}_i = \mathbf{0} \tag{35a}$$

$$\vdots$$

$$\frac{\partial \hat{A}_{\mathrm{d}}}{\partial \mathbf{q}_{i+s^a-1}} = -\alpha \Delta t^a \nabla V^a(\mathbf{q}_{i+s^a-2+\alpha}) - (1-\alpha)\Delta t^a \nabla V^a(\mathbf{q}_{i+s^a-1+\alpha}) \\ -\mathbf{M}^a(\mathbf{q}_{i+s^a} - 2\mathbf{q}_{i+s^a-1} + \mathbf{q}_{i+s^a-2})/\Delta t^a + \mathbf{B}^{a\mathrm{T}}\boldsymbol{\lambda}_{i+s^a-1} = \mathbf{0} \tag{35b}$$

s^a equations constraining domain a to the interface at each substep,

$$\boldsymbol{\phi}^a(\mathbf{q}_{i+1}) = \mathbf{B}^a \mathbf{q}_{i+1} - \mathbf{C}^a \boldsymbol{\psi}_{i+1} = \mathbf{0} \tag{35c}$$

$$\vdots$$

$$\boldsymbol{\phi}^a(\mathbf{q}_{i+s^a}) = \mathbf{B}^a \mathbf{q}_{i+s^a} - \mathbf{C}^a \boldsymbol{\psi}_{i+s^a} = \mathbf{0} \tag{35d}$$

and similar equations for domain b. For the interface, we have

$$\frac{\partial \hat{A}_{\mathrm{d}}}{\partial \boldsymbol{\psi}_n} = \sum_{k=0}^{s^a-1} \mathbf{C}^{a\mathrm{T}} \left[-\left(\frac{k}{s^a}\right)\boldsymbol{\lambda}_{i+k-s^a} - \left(1-\frac{k}{s^a}\right)\boldsymbol{\lambda}_{i+k} \right] \\ + \sum_{k=0}^{s^b-1} \mathbf{C}^{b\mathrm{T}} \left[-\left(\frac{k}{s^b}\right)\boldsymbol{\lambda}_{j+k-s^b} - \left(1-\frac{k}{s^b}\right)\boldsymbol{\lambda}_{j+k} \right] = \mathbf{0} \tag{35e}$$

DOI: 10.1002/nme

As usual, for a non-linear problem this system must be solved using Newton's method or another iterative technique. The Jacobian is a block-bordered matrix; for example, with $s^a = 3$ and $s^b = 1$ we obtain

$$\mathbf{J}_g = \left[\begin{array}{ccccccc|cc|c}
\mathbf{F}_{i+1} & & & \mathbf{B}^{a\mathrm{T}} & & & & & & \\
-\mathbf{G}_{i+2} & \mathbf{F}_{i+2} & & & \mathbf{B}^{a\mathrm{T}} & & & & & \\
\mathbf{F}_{i+2} & -\mathbf{G}_{i+3} & \mathbf{F}_{i+3} & & & & \mathbf{B}^{a\mathrm{T}} & & & \\
\mathbf{B}^a & & & & & & & & & -\tfrac{1}{3}\mathbf{C}^a \\
 & \mathbf{B}^a & & & & & & & & -\tfrac{2}{3}\mathbf{C}^a \\
 & & \mathbf{B}^a & & & & & & & -\mathbf{C}^a \\
\hline
 & & & & & & & \mathbf{F}_{j+1} & \mathbf{B}^{b\mathrm{T}} & \\
 & & & & & & & \mathbf{B}^b & & -\mathbf{C}^b \\
\hline
 & & & -\mathbf{C}^{a\mathrm{T}} & -\tfrac{2}{3}\mathbf{C}^{a\mathrm{T}} & -\tfrac{1}{3}\mathbf{C}^{a\mathrm{T}} & & & -\mathbf{C}^{b\mathrm{T}} &
\end{array}\right]$$

where

$$\mathbf{F}_{i+1} = \alpha(1-\alpha)\Delta t^a \mathbf{J}_f^a(\mathbf{q}_{i+\alpha}) - \frac{1}{\Delta t^a}\mathbf{M}^a$$

$$\mathbf{G}_{i+1} = -\alpha^2 \Delta t^a \mathbf{J}_f^a(\mathbf{q}_{i-1+\alpha}) - (1-\alpha)^2 \Delta t^a \mathbf{J}_f^a(\mathbf{q}_{i+\alpha}) - \frac{2}{\Delta t^a}\mathbf{M}^a$$

for domain a, and similar $\mathbf{F}_j$ and $\mathbf{G}_j$ for domain b. The solution vector is

$$\mathbf{x} = [\underbrace{\mathbf{q}_{i+1}\ \mathbf{q}_{i+2}\ \mathbf{q}_{i+3}\ \boldsymbol{\lambda}_i\ \boldsymbol{\lambda}_{i+1}\ \boldsymbol{\lambda}_{i+2}}_{\text{domain } a}\ \underbrace{\mathbf{q}_{j+1}\ \boldsymbol{\lambda}_j}_{\text{domain } b}\ \underbrace{\boldsymbol{\psi}_{n+1}}_{\text{interface}}\ ]^{\mathrm{T}} \tag{36}$$

For linear problems, the **F** and **G** submatrices will be constant; hence, they could potentially be stored just once in the memory. For non-linear problems, each submatrix of the Jacobian depends on a different position; hence, they will vary. This can make the Jacobian expensive to compute and store. Alternatively, we could use an approximate Jacobian where the submatrices are all equal. This would make the Jacobian much faster to compute, but could adversely affect the convergence rate of Newton's method.

To initialize the Lagrangian integrator, the positions and Lagrange multipliers for the entire first system step must be known. Given an initial position and velocity, these can be computed using a single step of the asynchronous Hamiltonian integrator. In our formulation, the Lagrangian integrator results in a smaller system to solve than the Hamiltonian integrator, and hence is more efficient for computing subsequent steps. Since the Lagrangian and Hamiltonian integrators are equivalent, they will compute the same positions and have the same rate of convergence, as shown in Section 4.

DOI: 10.1002/nme

3.2.2. The Hamiltonian framework. To derive the asynchronous Hamiltonian integrator, we apply s^a substeps of the single-step Hamiltonian integrator to domain a and s^b substeps to domain b, with the constraint at each substep that their interfaces match the linear interpolation of the common interface. Since there is a mismatch between times in which momentum is computed for each domain, we propose applying the momentum constraint on only the system time. Constraining a linear interpolation of the momenta, as for displacements, would lead to a dissipative integrator.

Applying the symmetric generalized midpoint rule $L_{\mathrm{d}}^{\mathrm{sym},\alpha}$ (13), and introducing the abbreviation $\mathbf{v}_{i+1/2}=(\mathbf{q}_{i+1}-\mathbf{q}_i)/\Delta t^a$, we obtain s^a equations for the positions of domain a,

$$\mathbf{p}_i = \tfrac{1}{2}(1-\alpha)\Delta t^a \nabla V(\mathbf{q}_{i+\alpha}) + \tfrac{1}{2}\alpha\Delta t^a \nabla V(\mathbf{q}_{i+1-\alpha}) + \mathbf{M}\mathbf{v}_{i+1/2} - \mathbf{B}^{a\mathrm{T}}\boldsymbol{\lambda}_i \tag{37a}$$

$$\vdots$$

$$\mathbf{p}_{i+s^a-1} = \tfrac{1}{2}(1-\alpha)\Delta t^a \nabla V(\mathbf{q}_{i+s^a-1+\alpha}) + \tfrac{1}{2}\alpha\Delta t^a \nabla V(\mathbf{q}_{i+s^a-\alpha}) + \mathbf{M}\mathbf{v}_{i+s^a-1/2} - \mathbf{B}^{a\mathrm{T}}\boldsymbol{\lambda}_{i+s^a-1} \tag{37b}$$

and s^a equations for momenta,

$$\mathbf{p}_{i+1} = -\tfrac{1}{2}\alpha\Delta t^a \nabla V(\mathbf{q}_{i+\alpha}) - \tfrac{1}{2}(1-\alpha)\Delta t^a \nabla V(\mathbf{q}_{i+1-\alpha}) + \mathbf{M}\mathbf{v}_{i+1/2} \tag{37c}$$

$$\vdots$$

$$\mathbf{p}_{i+s^a-1} = -\tfrac{1}{2}\alpha\Delta t^a \nabla V(\mathbf{q}_{i+s^a-2+\alpha}) - \tfrac{1}{2}(1-\alpha)\Delta t^a \nabla V(\mathbf{q}_{i+s^a-1-\alpha}) + \mathbf{M}\mathbf{v}_{i+s^a-3/2} \tag{37d}$$

$$\mathbf{p}_{i+s^a} = -\tfrac{1}{2}\alpha\Delta t^a \nabla V(\mathbf{q}_{i+s^a-1+\alpha}) - \tfrac{1}{2}(1-\alpha)\Delta t^a \nabla V(\mathbf{q}_{i+s^a-\alpha}) + \mathbf{M}\mathbf{v}_{i+s^a-1/2} + \mathbf{B}^{a\mathrm{T}}\boldsymbol{\mu}_{n+1} \tag{37e}$$

and similar equations for the positions and momenta of domain b, except in the last equation the sign of $\boldsymbol{\mu}$ is changed,

$$\mathbf{p}_{j+s^b} = -\tfrac{1}{2}\alpha\Delta t^b \nabla V(\mathbf{q}_{j+s^b-1+\alpha}) - \tfrac{1}{2}(1-\alpha)\Delta t^b \nabla V(\mathbf{q}_{j+s^b-\alpha}) + \mathbf{M}\mathbf{v}_{j+s^b-1/2} - \mathbf{B}^{b\mathrm{T}}\boldsymbol{\mu}_{n+1} \tag{37f}$$

There is also an equation constraining the momentum at the final substep,

$$\mathbf{B}^a(\mathbf{M}^a)^{-1}\mathbf{p}_{i+s^a} - \mathbf{B}^b(\mathbf{M}^b)^{-1}\mathbf{p}_{j+s^b} = \mathbf{0} \tag{37g}$$

and an equation for the interface,

$$\sum_{k=0}^{s^a-1} \mathbf{C}^{a\mathrm{T}}\left[-\left(\frac{k}{s^a}\right)\boldsymbol{\lambda}_{i+k-s^a} - \left(1-\frac{k}{s^a}\right)\boldsymbol{\lambda}_{i+k}\right] + \sum_{k=0}^{s^b-1} \mathbf{C}^{b\mathrm{T}}\left[-\left(\frac{k}{s^b}\right)\boldsymbol{\lambda}_{j+k-s^b} - \left(1-\frac{k}{s^b}\right)\boldsymbol{\lambda}_{j+k}\right] = \mathbf{0} \tag{37h}$$

Note that $\mathbf{q}_{i+2}$ depends on $\mathbf{p}_{i+1}$, $\mathbf{q}_{i+3}$ depends on $\mathbf{p}_{i+2}$, and so on. Hence, unlike in the previous sections, the intermediate momenta must be computed simultaneously with the positions. The final momenta $\mathbf{p}_{i+s^a}$ and $\mathbf{p}_{j+s^b}$ could still be post-processed along with the constraint on momentum,

DOI: 10.1002/nme

but we have not separated it here. As usual, this gives a block-bordered Jacobian matrix; for example, with $s^a = 2$ and $s^b = 1$ we obtain

$$\mathbf{J}_g = \left[\begin{array}{cccccc|cc|cc}
\mathbf{F}_{i+1} & & & & -\mathbf{B}^a & & & & & \\
-\mathbf{G}_{i+2} & \mathbf{F}_{i+2} & I & & & -\mathbf{B}^a & & & & \\
\mathbf{G}_{i+1} & & I & & & & & & & \\
-\mathbf{F}_{i+2} & \mathbf{G}_{i+2} & & I & & & & & \mathbf{B}^{a\mathrm{T}} & \\
\mathbf{B}^a & & & & & & & & & -\tfrac{1}{2}\mathbf{C}^a \\
 & \mathbf{B}^a & & & & & & & & -\mathbf{C}^a \\
\hline
 & & & & & & \mathbf{F}_{j+1} & -\mathbf{B}^{b\mathrm{T}} & & \\
 & & & & & & \mathbf{G}_{j+1} & I & -\mathbf{B}^{b\mathrm{T}} & \\
 & & & & & & \mathbf{B}^a & & & -\mathbf{B}^a \\
\hline
 & & \mathbf{D}^a & & & & & -\mathbf{D}^b & & \\
 & & & & -\mathbf{C}^{a\mathrm{T}} & -\tfrac{1}{2}\mathbf{C}^{a\mathrm{T}} & & -\mathbf{C}^{b\mathrm{T}} & &
\end{array}\right]$$

where

$$\mathbf{F}_i = \tfrac{1}{2}\alpha(1-\alpha)\Delta t^a \mathbf{J}_f^a(\mathbf{q}_{i+\alpha}) + \tfrac{1}{2}\alpha(1-\alpha)\Delta t^a \mathbf{J}_f^a(\mathbf{q}_{i+1-\alpha}) + \mathbf{M}^a/\Delta t^a$$

$$\mathbf{G}_i = \tfrac{1}{2}\alpha^2 \Delta t^a \mathbf{J}_f^a(\mathbf{q}_{i+\alpha}) + \tfrac{1}{2}(1-\alpha)^2 \Delta t^a \mathbf{J}_f^a(\mathbf{q}_{i+1-\alpha}) - \mathbf{M}^a/\Delta t^a$$

$$\mathbf{D}^a = \mathbf{B}^a(\mathbf{M}^a)^{-1}$$

for domain a, and similar $\mathbf{F}_j$, $\mathbf{G}_j$, and $\mathbf{D}^b$ for domain b. The solution vector is

$$\mathbf{x} = [\underbrace{\mathbf{q}_{i+1}\ \mathbf{q}_{i+2}\ \mathbf{p}_{i+1}\ \mathbf{p}_{i+2}\ \boldsymbol{\lambda}_i\ \boldsymbol{\lambda}_{i+1}}_{\text{domain } a}\ \underbrace{\mathbf{q}_{j+1}\ \mathbf{p}_{j+1}\ \boldsymbol{\lambda}_j}_{\text{domain } b}\ \underbrace{\boldsymbol{\mu}_{n+1}\ \boldsymbol{\psi}_{n+1}}_{\text{interface}}]^{\mathrm{T}} \tag{38}$$

As before, this block-bordered matrix naturally maps to parallel computing, with each processor getting a diagonal block of the matrix.

3.3. Stability

For a linear problem without domain decomposition, the stability condition for the Verlet ($\alpha = 0$) is $\Delta t < 2/\omega$, where ω^2 is the largest eigenvalue of the stiffness matrix and the midpoint method ($\alpha = \frac{1}{2}$) is unconditionally stable. This can be shown using the modal analysis [34, p. 421] or by the energy method [35]. For non-linear problems, there is ongoing research in the stability analysis. A generally accepted condition is that the stability limits given above are met by eigenvalues of the linearized problem at every Newton's iteration. However, Skeel and Srinivas [36] have shown that non-linearities may limit the time step more than this linearized analysis implies. The initial guess must also be sufficiently close for Newton's method to converge, which may further limit

DOI: 10.1002/nme

the time step from a computational perspective. Thus, in real non-linear simulations, one must pay attention to all three time step limitations—stability, accuracy, and region of convergence.

4. NUMERICAL EXPERIMENTS

To investigate properties of the proposed integrators, we performed numerical experiments simulating non-linear springs. The first set of numerical experiments uses a split single degree-of-freedom system. For the second set of numerical experiments, we chose the asynchronous Hamiltonian integrator and investigate a mesh of springs in three dimensions.

4.1. Split single degree of freedom

The first problem, shown in Figure 2, has a stiff spring with a light mass in domain a, and a non-stiff spring with a heavy mass in domain b, designed to accentuate potential problems with the coupling. Each spring has force

$$f(q) = -c \tanh\left(\frac{q}{2d}\right) \tag{39}$$

shown in Figure 4, and the parameters

$$M^a = 0.1, \quad c^a = 45, \quad d^a = 0.5$$
$$M^b = 3.9, \quad c^b = 3, \quad d^b = 0.5$$

Consistent units are used throughout the examples.

Owing to their variational nature, the integrators do not artificially dissipate energy. Figure 5 shows that all of the error is phase error, rather than amplitude error. Even over a very long time interval, where we have integrated to time $t = 10^6$ in Figure 6, the amplitude is maintained and the energy continues to oscillate around the true energy. Another consequence of the variational formulation of our integrators is that they are symplectic and so preserve area in the phase space.

For our non-linear model problem with domain decomposition, we found that for $\alpha = \frac{1}{2}$ the synchronous integrator retained the unconditional stability. When we changed to asynchronous steps, the unconditional stability was lost, but in our tests the stability limit for the time step still exceeded the time step required to attain reasonable accuracy. Whether this holds is of course problem dependent. Although we lost the unconditional stability, we gained the ability for each

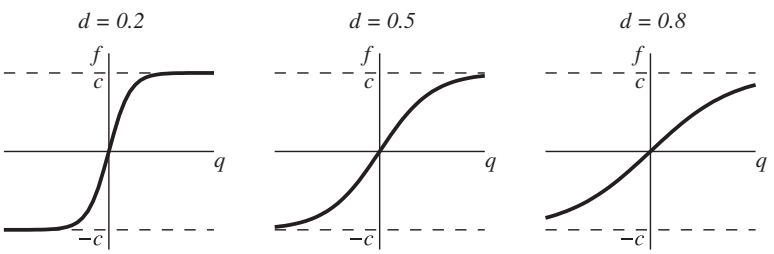

Figure 4. Non-linear spring force for various values of d.

DOI: 10.1002/nme

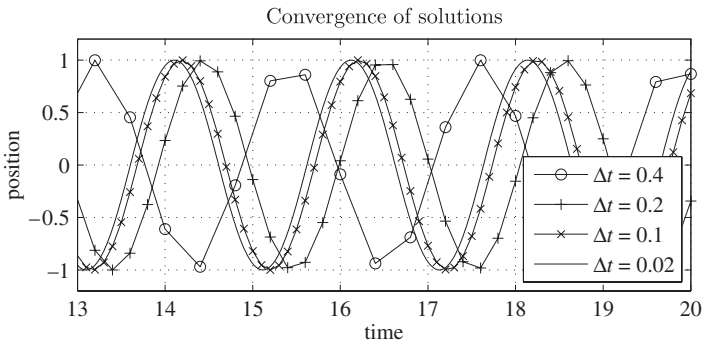

Figure 5. Convergence of the synchronous Hamiltonian integrator (29), (31) with $\alpha = \frac{1}{2}$ as time step $\Delta t \to 0$. Note that amplitude is same in all the cases; the only change is in the phase error.

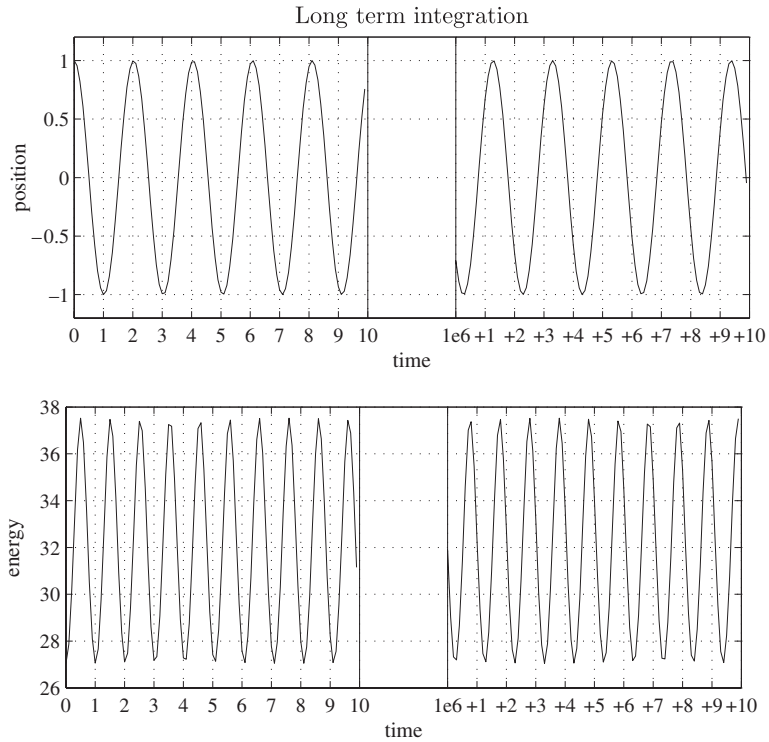

Figure 6. Long-term behavior of the synchronous Hamiltonian integrator (29), (31) with $\alpha = 0$, showing results near $t = 0$ and 10^6. Note that the amplitude does not change, and energy oscillates about a fixed value.

domain to use a different time step for the required accuracy in that domain. For problems where the accuracy requirement in one domain would otherwise limit the system time step, asynchronous integrators allow other domains to use larger time steps.

DOI: 10.1002/nme

In the error plots that follow, we use percent error in the average length of the period, shown in Figure 7 as this measurement is independent of the final time of integration. Figures 8–10 compare the stability limit with various levels of accuracy, showing that the time step for reasonable accuracy requirements is much smaller than the stability limit. The solution with 15% error shown in Figure 11 is clearly very coarse; hence, one would want to use a smaller time step than that. We verified the stability by monitoring the eigenvalues of the amplification matrix at every Newton step to ensure that they were never greater than 1 in magnitude. To estimate the error, we computed a reference solution using the Verlet integrator, without domain decomposition, and $\Delta t = 10^{-4}$.

We note some interesting features of the stability graphs. For $\alpha = 0$, the stability limit increases as the number of substeps increases, and approaches the limit for the $\alpha = \frac{1}{2}$ case. This is not entirely surprising: each substep is a linear interpolation between the system time steps, making the

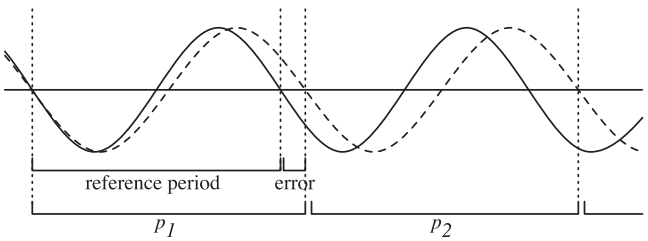

Figure 7. Phase error is computed by comparing the average solution period, $p = \sum_{i=1}^{n} p_i / n$, with the reference solution period.

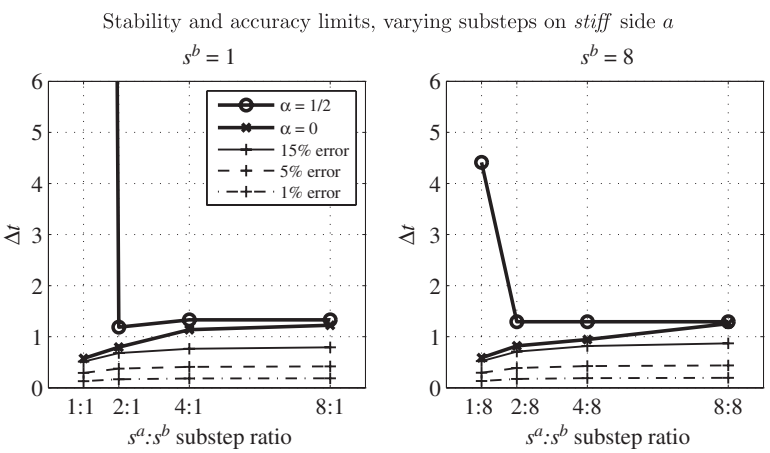

Figure 8. Bold lines are stability limits. Thin lines are constant error curves showing Δt that achieve specified error. For both values of α, time step to achieve reasonable error is significantly below stability limits. For $\alpha = \frac{1}{2}$ integrator, the stability limit initially drops from unconditional to conditional in changing from synchronous to asynchronous, but then becomes independent of the time step ratio. For $\alpha = 0$ integrator, the stability limit increases by a factor of 2.13 from 1:1 to 8:1 ratio.

DOI: 10.1002/nme

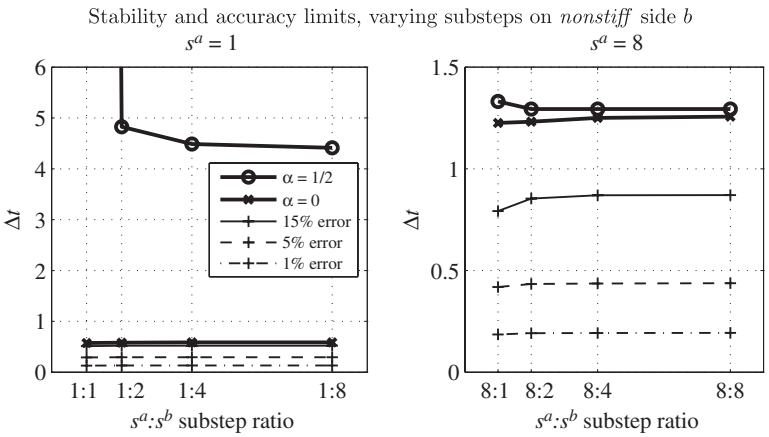

Figure 9. The stability limit is largely unaffected by changing the number of substeps on non-stiff side b.

integrator semi-implicit. It also shows that the coupling is as important as the individual integrators used in each subdomain. The stability limit then plateaus and becomes independent of the time step ratio as the number of substeps increases. Such behavior is important if one wants to utilize an arbitrary time step ratio without decreasing the stability limit. Changing the number of substeps on the non-stiff side has little effect on the stability limit, indicating that the limit is largely dictated by the stiff domain, as expected. When using different integrators for each domain, with $\alpha^a = \frac{1}{2}$ and $\alpha^b = 0$, or *vice versa*, the results are bounded by those shown using the same value of α for both domains, as shown in Figure 10.

For non-linear problems, the convergence of Newton's method at each time step is important for overall speed. Figure 12 shows that fast convergence is retained throughout the spring's cycle. This is as expected, since we used the actual Jacobian in the iteration.

Both the synchronous and asynchronous integrators achieved second-order convergence rates for positions, as shown in Figure 13. For the Lagrangian integrator, we computed velocities using a second-order finite difference, yielding a second-order convergence rate. For the Hamiltonian integrator, we computed velocities directly from the momentum. The synchronous integrator achieves second-order convergence for velocities, but the asynchronous integrator achieves only first-order convergence for velocities.

Examining how time refinement affects the error, we see in Figure 14 that refining on the stiff side reduces the error by half, whereas refining on the non-stiff side has little effect. This indicates that we can take larger steps on the non-stiff side without affecting the accuracy, and smaller steps on the stiff side to gain the accuracy required. The greatest change in error is adding a single substep from 1:1 to 2:1, so that even a moderate number of substeps may be sufficient to attain the desired accuracy.

4.2. Mesh of springs

To extend the results to a system with many degrees of freedom, we applied the asynchronous Hamiltonian integrator to a mesh of springs on the unit square with 147 degrees of freedom, shown

DOI: 10.1002/nme

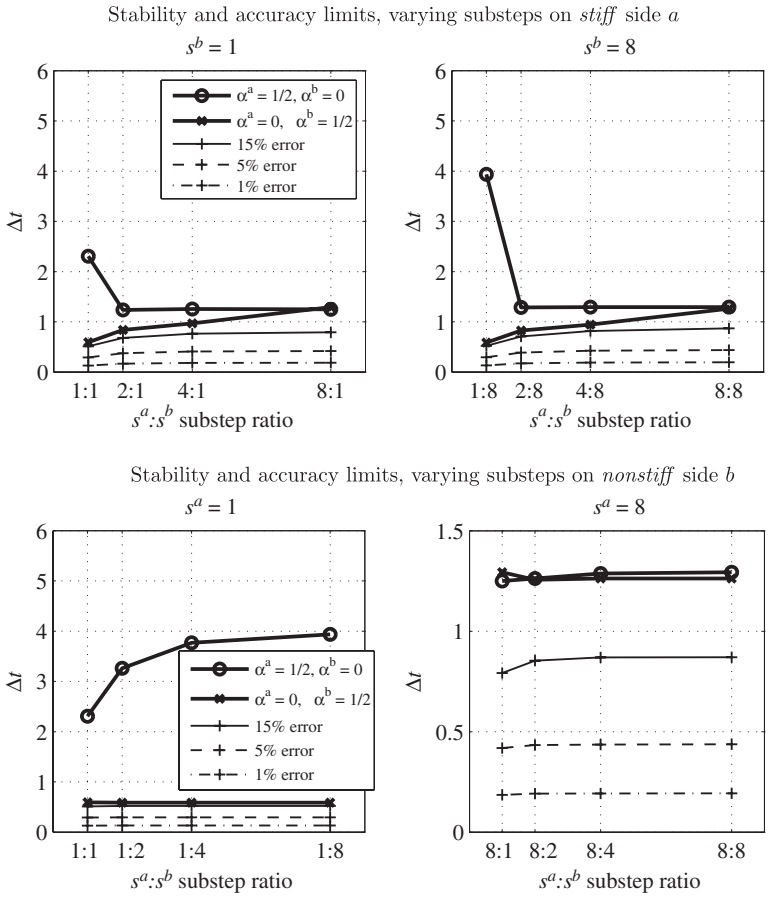

Figure 10. Results for the mixed explicit–implicit integrator are bounded by results in Figures 8 and 9 with the same value of α for both domains.

in Figure 15. Meshes of springs are used, for example, in computer graphics to simulate cloth, where Runge–Kutta or backward Euler integration is often employed [37, 38]. The spring force between node i and node j is

$$\mathbf{f}(\mathbf{x}^i, \mathbf{x}^j) = -c \tanh\left(\frac{\|\mathbf{w}\| - l_0}{2d}\right) \frac{\mathbf{w}}{\|\mathbf{w}\|} \tag{40}$$

where $\mathbf{x}^i = [x^i \ y^i \ z^i]^{\mathrm{T}}$ is the position of node i, $\mathbf{w} = \mathbf{x}^i - \mathbf{x}^j$, and l_0 is the spring's rest length. The left domain a has stiff springs with $c^a = 45$ and $d^a = 0.5$, whereas the right domain b has non-stiff springs with $c^b = 3$ and $d^b = 0.5$. Each node has a mass of $M = 0.1$. Nodes on the boundary are homogeneous Dirichlet. The initial condition is a bubble function,

$$z^i = \sin(\pi x^i) \sin(\pi y^i) \tag{41}$$

Solutions at various time steps are shown in Figure 16.

DOI: 10.1002/nme

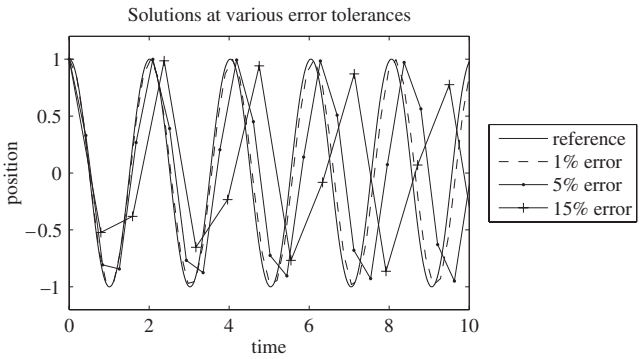

Figure 11. A comparison of solutions at various error tolerances, showing poor quality of solutions at higher error tolerances. One would undoubtedly choose a time step smaller than that used for 15% error solution, which is still under the stability limit. Here we use the percent error in a period. Solutions were computed using $\alpha = \frac{1}{2}, s^a = 8, s^b = 1$.

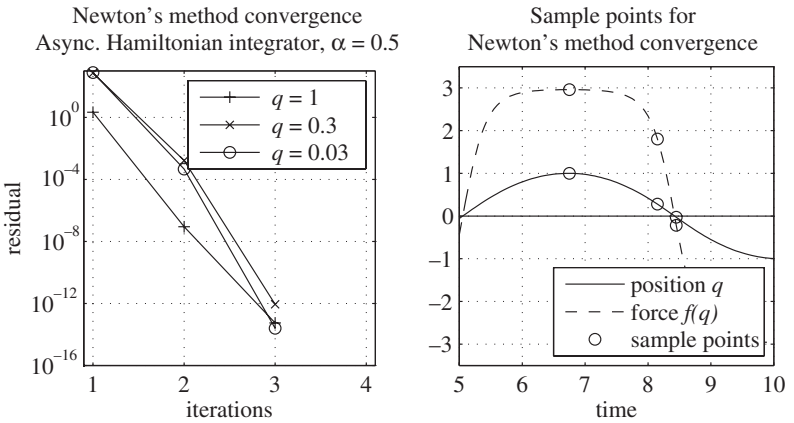

Figure 12. Newton's method exhibits fast convergence throughout the whole spring cycle. The graph on the left shows the convergence for three sample positions; the graph on the right depicts the position and force at those sample positions. Results for $s^a = 8$, $s^b = 1$ ratio are shown. The residual is normalized relative to the initial residual of first step, $R = \|g_n^{(i)}\| / \|g_0^{(0)}\|$.

The stability results in Figure 17 bear out the conclusions from the single degree-of-freedom case. For the $\alpha = \frac{1}{2}$ integrator, the stability limit initially decreases but then levels out as the stiff side is refined. For the $\alpha = 0$ integrator, the stability limit doubles as the stiff side is refined. For this problem, the eigenvalue analysis of the linearized system did not yield useful information about the stability limit; hence, we based our stability analysis on running for at least 2000 time steps and testing whether any position exceeded a bounding volume.

The error measurements in Table I indicate that for this problem, the accuracy limit is again below the stability limit. For instance, to realize an accuracy of 5% would require choosing a Δt about half the stability limit for $\alpha = 0$, and significantly below the stability for $\alpha = \frac{1}{2}$. Because

DOI: 10.1002/nme

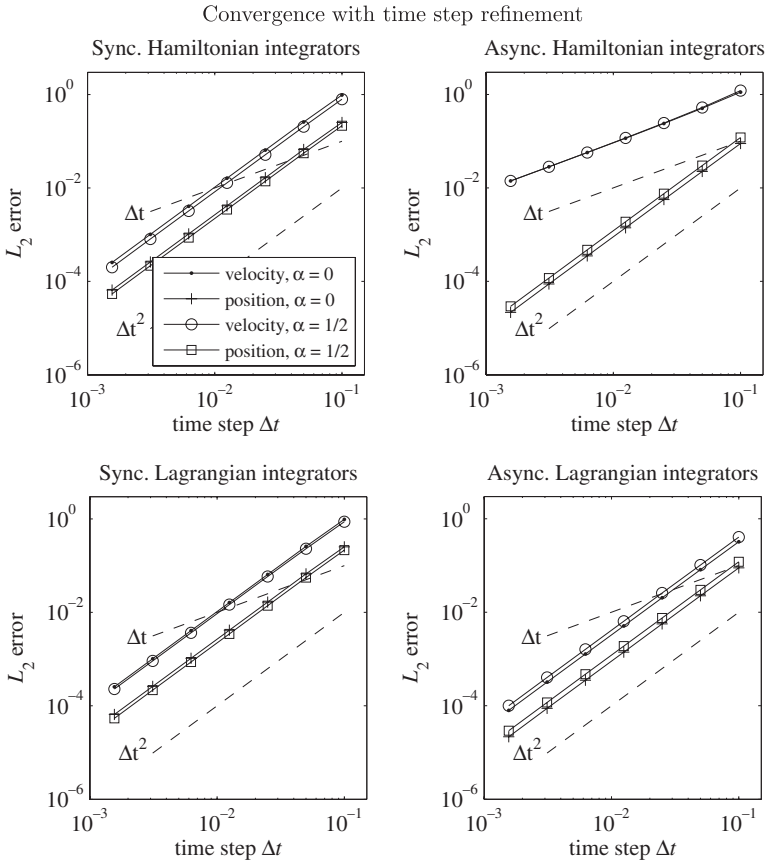

Figure 13. The Hamiltonian and Lagrangian integrators both achieve the second-order convergence for positions. The Hamiltonian integrators, on top, become first-order accurate for velocities when going from synchronous to asynchronous, whereas the Lagrangian integrators, on bottom, retain the second-order accuracy for velocities. Asynchronous results shown are for $s^a = 8$, $s^b = 1$ ratio. Similar results are obtained for different ratios. The error is $\|q - q_{\text{ref}}\|_{L_2}$.

there is no readily identifiable period for this problem, we used the L_2 error measure, $\|\mathbf{q} - \mathbf{q}_{\text{ref}}\|_{L_2}$, integrated from the initial time to the final time. This represents a cumulative global error, which grows with time. The final time of integration is arbitrary; we chose $t_{\text{F}} = 100$ so that the system would go through several major periods. For a reference solution we used Verlet without domain decomposition, with $\Delta t = 0.0002$.

As before, the integrators achieve the second-order accuracy for positions shown in Figure 18. However, now the asynchronous integrator also achieved the second-order accuracy for velocities, instead of only first-order accuracy as in the first test problem (Figure 13). From this, we conjecture that velocities at the interface are less accurate, but do not pollute the rest of the domain. A more rigorous investigation is required. We also observed that the extra momentum constraint (37g) reduces the velocity error by a constant factor, but does not affect the convergence rate.

DOI: 10.1002/nme

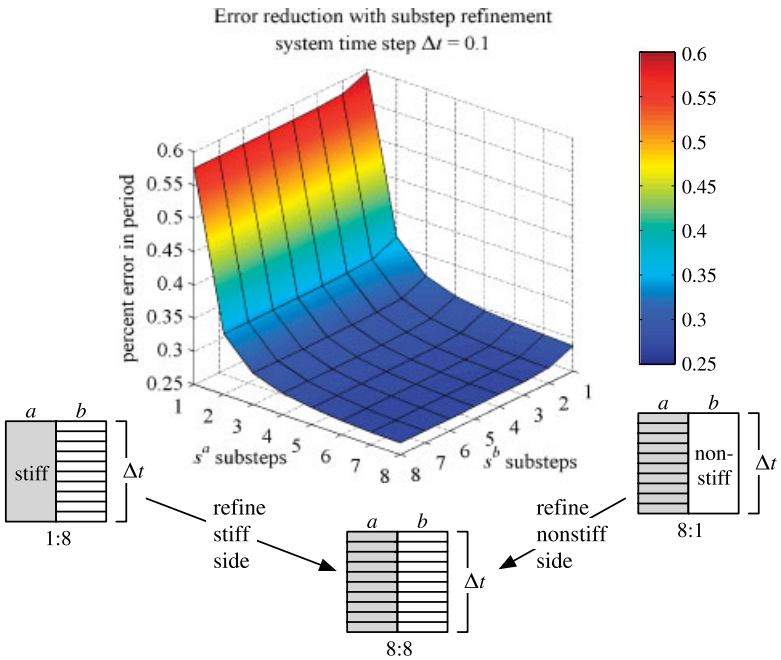

Figure 14. The change in error as time steps for stiff domain a and non-stiff domain b are refined, with fixed system time step Δt. As the stiff time step is refined, along the left axis, error is reduced by half. As the non-stiff time step is refined, along the right axis, there is little change in error. With this system time step Δt, all errors are less than 1%. Results shown are for the asynchronous Hamiltonian integrator with $\alpha = \frac{1}{2}$.

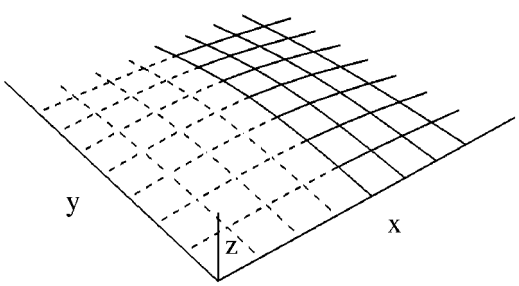

Figure 15. Mesh of springs on a unit square, containing 112 springs and 147 degrees of freedom. Domain a, with $0 \leqslant x \leqslant \frac{1}{2}$, has stiff springs in dashed line. Domain b, with $\frac{1}{2} \leqslant x \leqslant 1$, has non-stiff springs in solid line. Boundary nodes are Dirichlet.

5. CONCLUSIONS

In this paper, we have shown how to derive both synchronous and asynchronous integrators with domain decomposition in both the Lagrangian and Hamiltonian frameworks. For the Lagrangian

DOI: 10.1002/nme

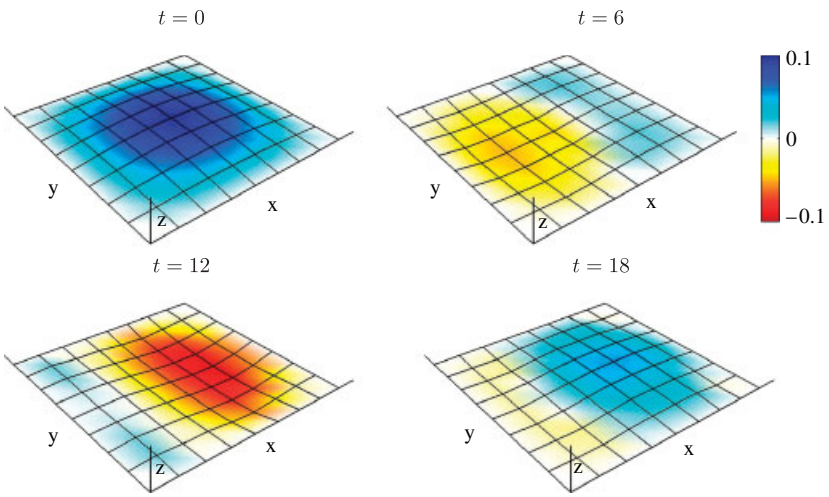

Figure 16. The initial condition and solution at selected times. For visualization purposes, the mesh is drawn as a solid surface and colored by z height. Black lines indicate springs.

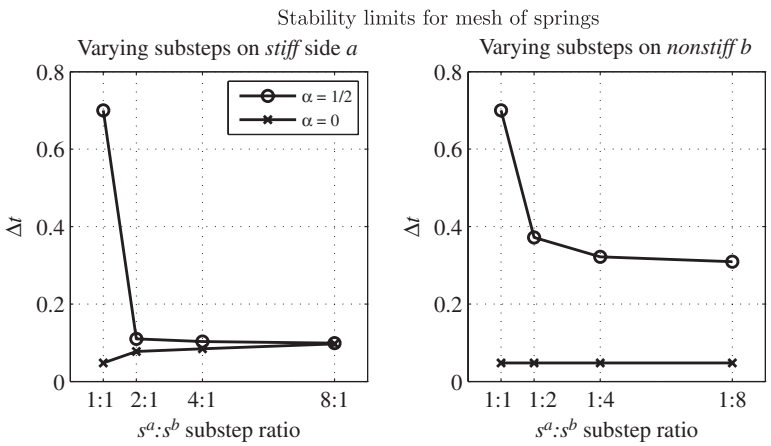

Figure 17. For $\alpha = \frac{1}{2}$ integrator, the stability limit drops when taking asynchronous steps, but then levels off. For $\alpha = 0$ integrator, refining the stiff side doubles the stability limit, whereas refining the non-stiff side does not affect the stability limit.

integrator we enforce the continuity of position at the interface, whereas for the Hamiltonian integrator we enforce the continuity of both position and velocity at the interface. Each domain can be computed in parallel, making the method suitable for parallel computing. The similarity of implementation to the Newmark method makes our integrators easy to incorporate into the existing codes.

As we show by various examples, these integrators have stability properties superior to traditional integrators for long-term time integration. Experiments on a non-linear problem have shown that our methods are symplectic and approximately preserve energy, such that the energy oscillates

DOI: 10.1002/nme

Table I. The relative percent error $\|\mathbf{q}-\mathbf{q}_{\text{ref}}\|_{L_2}/\|\mathbf{q}_{\text{ref}}\|_{L_2}$ at time $t=100$ for various integrators.

	$\alpha=0$				$\alpha=\frac{1}{2}$			
Δt	Verlet	1:1 Ratio	8:1 Ratio	1:8 Ratio	Midpoint	1:1 Ratio	8:1 Ratio	1:8 Ratio
0.16	—	—	—	—	19	18	—	12
0.08	—	—	12	—	11	11	9.6	9.6
0.04	8.1	7.8	4.9	3.0	12	11	8.1	7.4
0.02	3.0	2.9	1.7	1.6	5.0	4.9	3.1	3.0
0.01	1.4	1.3	0.50	1.2	2.1	2.0	0.92	1.7
0.005	0.44	0.42	0.13	0.39	0.87	0.84	0.23	0.79
0.0025	0.11	0.11	0.032	0.10	0.23	0.22	0.059	0.21
0.00125	0.028	0.027	0.0088	0.025	0.058	0.056	0.015	0.053

Blank entries are numerically unstable runs. To realize error less than 5%, Δt must be chosen less than the stability limit. In addition, note that the error does not increase when changing from undecomposed Verlet and midpoint integrators to synchronous decomposed 1:1 integrators.

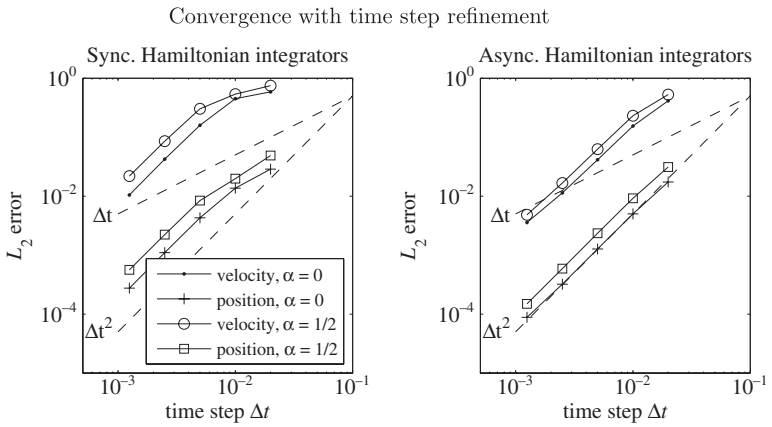

Figure 18. Both the synchronous and asynchronous Hamiltonian integrators achieved the second-order convergence for positions and velocities. Asynchronous results are for the 8:1 ratio. The error is $\|\mathbf{q}-\mathbf{q}_{\text{ref}}\|_{L_2}/\|\mathbf{q}_{\text{ref}}\|_{L_2}$.

about a fixed value. The stability limit becomes independent of the time step ratio, and in practice the system time step is limited by the accuracy requirement rather than by the stability requirement.

There are a number of directions for future work. We are interested in further investigating the asynchronous scheme to improve its order of accuracy and stability, particularly the loss of unconditional stability. One potential direction is to investigate other interpolations of the system interface than the linear interpolation we used here. Another improvement for the Hamiltonian integrator would be to constrain the velocity at every substep, rather than at just the system time step. Composition methods [17] also afford a means of generating variational integrators with higher-order accuracy, which may benefit the asynchronous integrators.

We plan to add external forces and dissipation, whose effects can be accurately computed by variational integrators [18]. We intend to extend the methods developed here to PDEs and investigate how to solve the resulting systems efficiently in parallel. We are also interested in extending this

DOI: 10.1002/nme

work to domains with non-matching meshes at the interface. To handle non-matching meshes, we believe that the common interface between domains that we have included in this work will be of vital importance [39]. Ultimately, we want to extend the proposed method to multi-physics problems.

ACKNOWLEDGEMENTS

This work was supported by the Center for Simulation of Advanced Rockets under contract number B523819 funded by the U.S. Department of Energy.

REFERENCES

1. Belytschko T, Mullen R. Mesh partitions of explicit–implicit time integration. In *Formulations and Computational Algorithms in Finite Element Analysis*, Bathe K, Oden J, Wunderlich W (eds). MIT Press: New York, 1976; 673–690.
2. Belytschko T, Mullen R. Stability of explicit–implicit mesh partitions in time integration. *International Journal for Numerical Methods in Engineering* 1978; **12**:1575–1586.
3. Hughes TJR, Liu W. Implicit–explicit finite elements in transient analysis: stability theory. *Journal of Applied Mechanics* 1978; **45**:371–374.
4. Hughes TJR, Liu W. Implicit–explicit finite elements in transient analysis: implementation and numerical examples. *Journal of Applied Mechanics* 1978; **45**:375–378.
5. Belytschko T, Yen H, Mullen R. Mixed methods for time integration. *Computer Methods in Applied Mechanics and Engineering* 1979; **17–18**:259–275.
6. Belytschko T. Partitioned and adaptive algorithms for explicit time integration. In *Nonlinear Finite Element Analysis in Structural Mechanics*, Wunderlich W, Stein E, Bathe KJ (eds). Springer: Berlin, 1981; 572–584.
7. Neal MO, Belytschko T. Explicit–explicit subcycling with non-integer time step ratios for structural dynamic systems. *Computers and Structures* 1989; **31**:871–880.
8. Smolinski P, Sleith S. Explicit multi-time step methods for structural dynamics. *PVP-Vol. 246/AMD-Vol. 143*, *New Methods in Transient Analysis*. ASME: New York, 1992; 1–4.
9. Smolinski P, Sleith S, Belytschko T. Stability of an explicit multi-time step integration algorithm for linear structural dynamics equations. *Computational Mechanics* 1996; **18**:236–244.
10. Daniel WJT. A study of the stability of subcycling algorithms in structural dynamics. *Computer Methods in Applied Mechanics and Engineering* 1998; **156**:1–13.
11. Combescure A, Gravouil A. A numerical scheme to couple subdomains with different time-steps for predominantly linear transient analysis. *Computer Methods in Applied Mechanics and Engineering* 2002; **191**:1129–1157.
12. Gravouil A, Combescure A. Multi-time-step explicit–implicit method for non-linear structural dynamics. *International Journal for Numerical Methods in Engineering* 2001; **50**:199–225.
13. Prakash A, Hjelmstad KD. A FETI-based multi-time-step coupling method for Newmark schemes in structural dynamics. *International Journal for Numerical Methods in Engineering* 2004; **61**:2183–2204.
14. Veselov AP. Integrable discrete-time systems and difference operators. *Functional Analysis and its Applications* 1988; **22**:83–93.
15. Veselov AP. Integrable Lagrangian correspondences and the factorization of matrix polynomials. *Functional Analysis and its Applications* 1991; **25**:112–122.
16. Wendlandt JM, Marsden JE. Mechanics integrators derived from a discrete variational principle. *Physica D* 1997; **106**:223–246.
17. Marsden JE, West M. Discrete mechanics and variational integrators. *Acta Numerica* 2001; **10**:357–514.
18. Kane C, Marsden JE, Ortiz M, West M. Variational integrators and the Newmark algorithm for conservative and dissipative mechanical systems. *International Journal for Numerical Methods in Engineering* 2000; **49**:1295–1325.
19. Newmark NM. A method of computation for structural dynamics. *Journal of the Engineering Mechanics Division* 1959; **85**:67–94.
20. Lew A, Marsden JE, Ortiz M, West M. Asynchronous variational integrators. *Archive for Rational Mechanics and Analysis* 2003; **167**:85–146.

DOI: 10.1002/nme

21. Lew A, Marsden JE, Ortiz M, West M. Variational time integrators. *International Journal for Numerical Methods in Engineering* 2004; **60**:153–212.
22. Kale KG, Lew AJ. Parallel asynchronous variational integrators. *International Journal for Numerical Methods in Engineering* 2007; **70**:291–321.
23. Farhat C, Roux FX. A method of finite element tearing and interconnecting and its parallel solution algorithm. *International Journal for Numerical Methods in Engineering* 1991; **32**:1205–1227.
24. Farhat C, Crivelli L, Roux FX. A transient FETI methodology for large-scale parallel implicit computations in structural mechanics. *International Journal for Numerical Methods in Engineering* 1994; **37**:1945–1975.
25. Farhat C, Chen P, Mandel J. A scalable Lagrange multiplier based domain decomposition method for implicit time-dependent problems. *International Journal for Numerical Methods in Engineering* 1995; **38**:3831–3858.
26. Park KC, Felippa CA, Gumaste U. A localized version of the method of Lagrange multipliers and its applications. *Computational Mechanics* 2000; **24**:476–490.
27. Park KC, Felippa CA, Rebel G. A simple algorithm for localized construction of non-matching structural interfaces. *International Journal for Numerical Methods in Engineering* 2002; **53**:2117–2142.
28. Brezzi F, Marini L. The three-field formulation for elasticity problems. *GAMM Mitteilungen* 2005; **28**:124–153.
29. Hairer E, Lubich C, Wanner G. *Geometric Numerical Integration* (2nd edn). Springer: Berlin, 2006.
30. Verlet L. Computer experiments on classical fluids. *Physical Review* 1967; **159**:98–103.
31. Ryckaert J, Ciccotti G, Berendsen H. Numerical integration of the Cartesian equations of motion of a system with constraints. *Journal of Computational Physics* 1977; **23**:327–341.
32. Chan T, Mathew T. Domain decomposition algorithms. *Acta Numerica* 1994; **3**:61–143.
33. Anderson H. RATTLE: a velocity version of the SHAKE algorithm for molecular dynamics calculations. *Journal of Computational Physics* 1983; **52**:24–34.
34. Cook RD, Malkus DS, Plesha ME, Witt RJ. *Concepts and Applications of Finite Element Analysis* (4th edn). Wiley: New York, 2002.
35. Hughes TJR. *The Finite Element Method* (2nd edn). Dover: New York, 2000.
36. Skeel R, Srinivas K. Nonlinear stability analysis of area-preserving integrators. *SIAM Journal on Numerical Analysis* 2000; **38**:129–148.
37. Baraff D, Witkin A. Large steps in cloth simulation. *Computer Graphics* 1998; **32**:43–54.
38. Ng HN, Grimsdale RL. Computer graphics techniques for modeling cloth. *IEEE Computer Graphics and Applications* 1996; **16**(5):28–41.
39. Jiao X, Heath MT. Common-refinement-based data transfer between non-matching meshes in multiphysics simulations. *International Journal for Numerical Methods in Engineering* 2004; **61**:2402–2427.

DOI: 10.1002/nme