An Explicit Jacobian for Newton's Method Applied to Numerical Approximations of Nonlinear Initial Boundary Value Problems*

Fredrik Laurén[†], Oskar Ålund [†], and Jan Nordström [†‡]

Abstract. We derive an explicit form of the Jacobian for discrete approximations of nonlinear initial boundary value problems (IBVPs) on matrix-vector form. The technique is exemplified on the incompressible Navier-Stokes equations in two dimensions. The Jacobian facilitates the use of Newton's method to solve the corresponding nonlinear system of equations. Appropriate boundary conditions are weakly imposed and we show how to compute the Jacobian for those parts of the discretization as well. The convergence rate of the iterations is verified by using the method of manufactured solutions. The methodology in this paper that can be used on any numerical discretization of IBVPs on matrix-vector form.

Key words. Nonlinear initial boundary value problems, Jacobian, Newton's method, incompressible Navier-Stokes equations, summation-by-parts, weak boundary conditions.

AMS subject classifications. 65M06, 65M12

1. Introduction.

Nonlinear systems of partial differential equations are common in computational science and engineering, and present multiple challenges. Stability is needed for reliability and high accuracy for fine solution details. For fast turnaround and timely result delivery, generic systems of nonlinear equations from the discretization of the form

$$\boldsymbol{\mathcal{F}}(\boldsymbol{\phi}) = 0 \tag{1.1}$$

must be solved efficiently [15]. This is the topic of this paper. Several techniques exist to solve (1.1), for example dual-time stepping [7, 13], optimization algorithms [10] or iterative methods [15]. Among the classical iterative methods, Newton's method is an effective choice due to its quadratic convergence order. The obvious drawback with Newton's method is that the Jacobian must be known. Methods that bypass this requirement and instead approximate the Jacobian lead to lower convergence orders, a typical example is the Secant method [15]. Alternatively, by using Newton-Krylov methodologies [8], only the action of the Jacobian is required and that can be approximated by $J_{\boldsymbol{\mathcal{F}}}(\boldsymbol{\phi}^k)\delta\boldsymbol{u} \approx (\boldsymbol{\mathcal{F}}(\boldsymbol{\phi}^k + \delta\boldsymbol{u}) - \boldsymbol{\mathcal{F}}(\boldsymbol{\phi}^k))/\delta$, where δ is small and $\boldsymbol{u}$ depends on the subspaces in the Krylov iterations. The advantage of Newton-Krylov methods is that an explicit Jacobian is never required, but sophisticated preconditioners becomes necessary [2] instead.

The focus in this paper is to facilitate the use of Newton's method where the key component is an exact explicit form of the Jacobian of (1.1). To exemplify our technique, we will use finite-

*Submitted to the editors October 21, 2021.

Funding: This work was funded by the Swedish Research Council (Stokcholm) under grant number 2018-05084_VR and SESSI.

[†]Department of Mathematics, Computational Mathematics, Linköping University, SE-581 83 Linköping, Sweden (fredrik.lauren@liu.se),(oskar.alund@liu.se), (jan.nordstrom@liu.se).

[‡]Department of Mathematics and Applied Mathematics, University of Johannesburg,P.O. Box 524, Auckland Park 2006, South Africa.

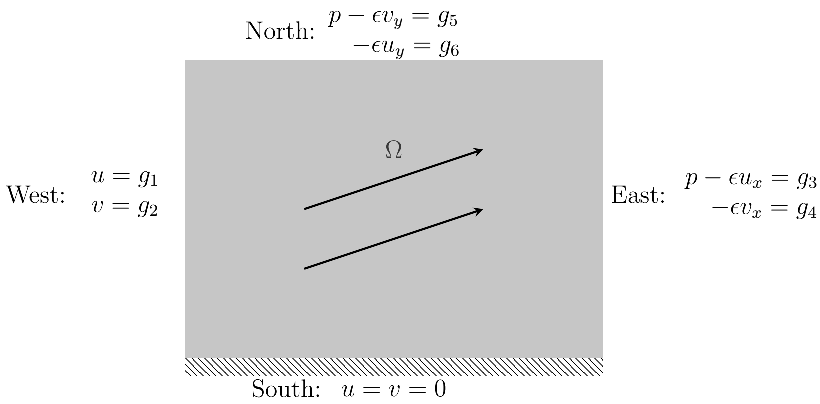

Figure 1. *Illustration of the computational domain Ω and the specific boundary conditions.*

difference operators on summation-by-parts (SBP) form [18] to discretize the incompressible Navier-Stokes (INS) equations in space. The boundary conditions will be weakly imposed via the Simultaneous Approximation Term (SAT) technique [3]. In [12], such a discretization based on the SBP-SAT technique of the nonlinear INS equations was proven to be stable, which is the key prerequisite.

Based on the formulation in [12], we show how the Jacobian can be explicitly calculated. It is also shown that the Jacobian has a block structure, where several blocks are precomputed when forming $\boldsymbol{\mathcal{F}}$, making the procedure very efficient.

To keep the paper focused on the derivation of the Jacobian, we follow [12] and consider a Cartesian grid. Exact Jacobians for numerical discretizations have recently been developed in [5] for so-called entropy stable numerical discretizations on SBP form in a periodic setting. Our new technique is not restricted to such specific discretizations, and we include the specific Jacobian related to the boundary conditions. The technique demonstrated in this paper can be used in a straightforward way on any numerical method for IBVPs that can be formulated on matrix-vector form. In addition, it can be readily extended to curvilinear grids [1], arbitrary dimensions, and other sets of linear and nonlinear equations. In principle all that is needed for the existence of the Jacobian is of course that $\boldsymbol{\mathcal{F}}$ is differentiable with respect to $\boldsymbol{\phi}$. This covers the various nonlinearities that arise in discretizations of the Navier-Stokes equations (both compressible and incompressible). However, the feasibility of explicitly deriving the Jacobian is highly dependent on the way in which $\boldsymbol{\mathcal{F}}$ is presented. As we shall see, discretizations on SBP-SAT form are particularly simple to differentiate, making Newton's method an attractive solution method.

The rest of the paper proceeds as follows. We introduce the continuous problem in Section 2 and present the semi-discrete formulation in Section 3. The Jacobian of the discretization is derived in Section 4. Implicit time integration is discussed in Section 5 and numerical experiments are performed in Section 6. Finally, conclusions are drawn in Section 7.

2. Problem formulation. As an illustrative example of our technique, we consider the scenario illustrated in Figure 1. An incompressible fluid is moving from left to right. Hence, the left side is an inflow boundary, where Dirichlet conditions are imposed, and the right side is an outflow boundary, where natural boundary conditions [14] are imposed. The lower part of the domain is a no-slip wall and the upper side is an outflow boundary, where again natural conditions are imposed. The initial-boundary value problem for the INS equations that we consider is

$$
\begin{aligned}
\tilde{I}\,\vec{w}_t + \mathcal{L}(\vec{w}) &= 0 \qquad && (x,y) \in \Omega, \qquad && t > 0 \\
\mathcal{H}\,\vec{w} &= \vec{g} && (x,y) \in \partial\Omega, && t > 0 \\
\tilde{I}\,\vec{w} &= \tilde{I}\,\vec{f} && (x,y) \in \Omega, && t = 0\,.
\end{aligned}
\tag{2.1}
$$

In (2.1), $\vec{w} = (u, v, p)^\top$, where u, v are the velocities in the x, y direction, respectively, and p is the pressure. Furthermore, $\Omega = [0,1]^2$ is the domain and $\partial\Omega$ its boundary. The initial data $\vec{f}$ and boundary data $\vec{g}$ are sufficiently smooth and compatible functions and the spatial operator is given by [12]

$$
\mathcal{L}(\vec{w}) = \frac{1}{2}\left[A\,\vec{w}_x + (A\,\vec{w})_x + B\,\vec{w}_y + (B\,\vec{w})_y\right] - \epsilon\tilde{I}[\,\vec{w}_{xx} + \vec{w}_{yy}]\,.
\tag{2.2}
$$

The matrices in (2.1) and (2.2) are

$$
A = \begin{pmatrix} u & 0 & 1 \\ 0 & u & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} v & 0 & 0 \\ 0 & v & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad \tilde{I} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}.
$$

Lastly, the explicit form of the boundary conditions $\mathcal{H}\,\vec{w} = \vec{g}$ reads

$$
\begin{aligned}
u &= g_1 & v &= g_2 & &\text{at} \quad x = 0 & &\text{(West)} \\
p - \epsilon u_x &= g_3 & -\epsilon v_x &= g_4 & &\text{at} \quad x = 1 & &\text{(East)} \\
u &= 0 & v &= 0 & &\text{at} \quad y = 0 & &\text{(South)} \\
p - \epsilon v_y &= g_5 & -\epsilon u_y &= g_6 & &\text{at} \quad y = 1 & &\text{(North)}\,.
\end{aligned}
\tag{2.3}
$$

2.1. Boundedness.

We will for completeness show how to bound the solution. For simplicity, only the south side of the domain is discussed explicitly. Details of the upcoming analysis are found in [12].

For two vector functions $\vec{\phi}$, $\vec{\psi}$ defined on Ω, we introduce the inner product and norm

$$
\langle \vec{\phi}, \vec{\psi} \rangle = \int_\Omega \vec{\phi}^{\,\top} \vec{\psi}\, d\Omega, \qquad \|\,\vec{\phi}\|^2 = \langle \vec{\phi}, \vec{\phi} \rangle\,.
$$

By multiplying (2.1) by $2\,\vec{w}^{\,\top}$ from the left and integrating over Ω, we get

$$
\frac{d}{dt}\|\,\vec{w}\|_{\tilde{I}}^2 + 2\epsilon\|\nabla\,\vec{w}\|_{\tilde{I}}^2 = BT\,,
\tag{2.4}
$$

where $\nabla\,\vec{w} = (\nabla u, \nabla v, \nabla p)^\top$, $\|\nabla\,\vec{w}\|_{\tilde{I}}^2$ is a dissipative volume term and

$$
BT = \int_{\text{South}} \vec{w}^{\,\top}(B\,\vec{w} - 2\epsilon\tilde{I}\,\vec{w}_y)\,dx
$$

contains the boundary terms evaluated at the south boundary. The other boundary terms are assumed dissipative and ignored. Imposing $u = v = 0$ results in $BT = 0$. Integrating (2.4) in time (assuming homogeneous boundary conditions on all sides) leads to

$$\|\vec{w}\|^2_{\tilde{I}}(T) + 2\epsilon \int_0^T \|\nabla \vec{w}\|^2_{\tilde{I}} dt \le \|f\|^2_{\tilde{I}}, \tag{2.5}$$

which bounds the semi-norm of the solution ($\|\vec{w}\|^2_{\tilde{I}}$) and its gradients ($\|\nabla \vec{w}\|^2_{\tilde{I}}$) for any time.

3. The semi-discrete scheme.

A brief introduction of the SBP-SAT technique is provided below and we recommend [6, 18] for extensive reviews.

We discretize the domain $\Omega = [0,1]^2$ with $N+1$ and $M+1$ grid points; $x_i = i/N$, $i = 0,\dots,N$ and $y_j = j/M$, $j = 0,\dots,M$ and let $n = (N+1)(M+1)$ denote the total number of grid points. A scalar function $q = q(x,y)$ defined on Ω is thereby represented on the grid by $\boldsymbol{q} = (q_{00},\dots,q_{0M},\dots q_{N0},\dots q_{NM})^\top$ where $q_{ij} = q(x_i, y_j)$. For the vector-valued function $\vec{w} = (u,v,p)^\top$, the approximation is arranged as $\vec{\boldsymbol{w}} = (\boldsymbol{u}^\top, \boldsymbol{v}^\top, \boldsymbol{p}^\top)^\top$. Let $\boldsymbol{D_x} = (P_x^{-1}Q_x) \otimes I_{M+1}$ and $\boldsymbol{D_y} = I_{N+1} \otimes (P_y^{-1}Q_y)$, where $\otimes$ denotes the Kronecker product. Then the approximations of the spatial derivatives are given by

$$\boldsymbol{D_x u} \approx \boldsymbol{u}_x, \quad \boldsymbol{D_y u} \approx \boldsymbol{u}_y\,.$$

The matrices $P_{x,y}$ are diagonal and positive definite, so that $\boldsymbol{P} = P_x \otimes P_y$ forms a quadrature rule that defines the norm $\|\vec{\boldsymbol{w}}\|^2_{I_3 \otimes \boldsymbol{P}} = \vec{\boldsymbol{w}}^\top (I_3 \otimes \boldsymbol{P}) \vec{\boldsymbol{w}} \approx \iint_\Omega \vec{w}^\top \vec{w} d\Omega$. We have also introduced I_k, which is the identity matrix of size k. Moreover, the matrices $Q_{x,y}$ satisfy the SBP-property

$$Q_x + Q_x^\top = E_N - E_{0x} \quad Q_y + Q_y^\top = E_M - E_{0y}\,, \tag{3.1}$$

where $E_{0x,y} = \mathrm{diag}(1,0,0,\dots,0)$ and $E_{N,M} = \mathrm{diag}(0,0,0,\dots,1)$ are matrices of appropriate sizes.

By using the notation above, the semi-discrete approximation of (2.1) becomes [12]

$$\tilde{\boldsymbol{I}} \vec{\boldsymbol{w}}_t + \boldsymbol{\mathcal{L}}(\vec{\boldsymbol{w}}) = \boldsymbol{\mathcal{S}}(\vec{\boldsymbol{w}})\,. \tag{3.2}$$

The discrete spatial operator is given by

$$\begin{aligned}\boldsymbol{\mathcal{L}}(\vec{\boldsymbol{w}}) = \frac{1}{2}\,&[\boldsymbol{A}(I_3 \otimes \boldsymbol{D_x})\vec{\boldsymbol{w}} + (I_3 \otimes \boldsymbol{D_x})\boldsymbol{A}\vec{\boldsymbol{w}} + \boldsymbol{B}(I_3 \otimes \boldsymbol{D_y})\vec{\boldsymbol{w}} + (I_3 \otimes \boldsymbol{D_y})\boldsymbol{B}\vec{\boldsymbol{w}}]\\ &- \epsilon \tilde{\boldsymbol{I}}[(I_3 \otimes \boldsymbol{D_x})^2 + (I_3 \otimes \boldsymbol{D_y})^2]\,\vec{\boldsymbol{w}}\,,\end{aligned}$$

and the block matrices are

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{U} & \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{U} & \boldsymbol{0} \\ \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \end{pmatrix}, \quad \boldsymbol{B} = \begin{pmatrix} \boldsymbol{V} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{V} & \boldsymbol{I} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \end{pmatrix}, \quad \tilde{\boldsymbol{I}} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{pmatrix},$$

where $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{n \times n}$ are diagonal matrices holding $\boldsymbol{u}, \boldsymbol{v}$, respectively. The matrices $\boldsymbol{I}$ and $\boldsymbol{0}$ are the identity and the zero matrix of size $n \times n$. Furthermore, $\boldsymbol{\mathcal{S}}(\vec{\boldsymbol{w}})$ contains penalty terms that enforce the boundary conditions.

The purpose of the SAT $\boldsymbol{\mathcal{S}}(\vec{\boldsymbol{w}})$ is i) to enforce the boundary conditions in (2.3) and ii) to stabilize the solution. One penalty term for each of the boundary conditions in (2.3) will be constructed. Let $k \in \{W, E, S, N\}$. The SAT at boundary k that enforces the boundary condition $H^k \vec{w} = \vec{g}$ has the general form

$$\boldsymbol{\mathcal{S}}^k(\vec{\boldsymbol{w}}) = (I_3 \otimes \boldsymbol{P}^{-1})\Sigma^k(I_3 \otimes \boldsymbol{P}^k)(\boldsymbol{\mathcal{H}}^k \vec{\boldsymbol{w}} - \vec{\boldsymbol{g}}) . \tag{3.3}$$

In (3.3), Σ^k is the penalty matrix to be determined for stability at boundary k. The quadratures are

$$\boldsymbol{P}^k = \begin{cases} E_{0x} \otimes P_y & \text{on the west boundary } (k = W) \\ E_N \otimes P_y & \text{on the east boundary } (k = E) \\ P_x \otimes E_{0y} & \text{on the south boundary } (k = S) \\ P_x \otimes E_M & \text{on the north boundary } (k = N) . \end{cases}$$

For the boundary conditions listed in (2.3), the penalty terms are

$$\begin{aligned}
\boldsymbol{\mathcal{S}}^W(\vec{\boldsymbol{w}}) &= (I_3 \otimes \boldsymbol{P}^{-1})\Sigma^W(I_3 \otimes \boldsymbol{P}^W) \underbrace{\begin{pmatrix} \boldsymbol{u} - \boldsymbol{g}_1 \\ \boldsymbol{v} - \boldsymbol{g}_2 \\ \boldsymbol{u} - \boldsymbol{g}_1 \end{pmatrix}}_{\boldsymbol{\mathcal{H}}^W \vec{\boldsymbol{w}} - \vec{\boldsymbol{g}}} \\
\boldsymbol{\mathcal{S}}^E(\vec{\boldsymbol{w}}) &= (I_3 \otimes \boldsymbol{P}^{-1})\Sigma^E(I_3 \otimes \boldsymbol{P}^E) \underbrace{\begin{pmatrix} \boldsymbol{p} - \epsilon \boldsymbol{D_x} \boldsymbol{u} - \boldsymbol{g}_3 \\ -\epsilon \boldsymbol{D_x} \boldsymbol{v} - \boldsymbol{g}_4 \\ \boldsymbol{0} \end{pmatrix}}_{\boldsymbol{\mathcal{H}}^E \vec{\boldsymbol{w}} - \vec{\boldsymbol{g}}} \\
\boldsymbol{\mathcal{S}}^S(\vec{\boldsymbol{w}}) &= (I_3 \otimes \boldsymbol{P}^{-1})\Sigma^S(I_3 \otimes \boldsymbol{P}^S) \underbrace{\begin{pmatrix} \boldsymbol{u} - \boldsymbol{0} \\ \boldsymbol{v} - \boldsymbol{0} \\ \boldsymbol{v} - \boldsymbol{0} \end{pmatrix}}_{\boldsymbol{\mathcal{H}}^S \vec{\boldsymbol{w}} - \boldsymbol{0}} \\
\boldsymbol{\mathcal{S}}^N(\vec{\boldsymbol{w}}) &= (I_3 \otimes \boldsymbol{P}^{-1})\Sigma^N(I_3 \otimes \boldsymbol{P}^N) \underbrace{\begin{pmatrix} -\epsilon \boldsymbol{D_y} \boldsymbol{u} - \boldsymbol{g}_6 \\ \boldsymbol{p} - \epsilon \boldsymbol{D_y} \boldsymbol{v} - \boldsymbol{g}_5 \\ \boldsymbol{0} \end{pmatrix}}_{\boldsymbol{\mathcal{H}}^N \vec{\boldsymbol{w}} - \vec{\boldsymbol{g}}} ,
\end{aligned} \tag{3.4}$$

where

$$\Sigma^W = \begin{pmatrix} -\boldsymbol{U}/2 + \epsilon \boldsymbol{D}_{\boldsymbol{x}}^\top & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{U}/2 + \epsilon \boldsymbol{D}_{\boldsymbol{x}}^\top & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & -\boldsymbol{I} \end{pmatrix}, \qquad \Sigma^E = (I_3 \otimes \boldsymbol{I})$$

$$\Sigma^S = \begin{pmatrix} -\boldsymbol{V}/2 + \epsilon \boldsymbol{D}_{\boldsymbol{y}}^\top & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{V}/2 + \epsilon \boldsymbol{D}_{\boldsymbol{y}}^\top & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & -\boldsymbol{I} \end{pmatrix}, \qquad \Sigma^N = (I_3 \otimes \boldsymbol{I}) .$$

As an example, the south penalty term can be written as

$$\boldsymbol{\mathcal{S}}^S(\vec{\boldsymbol{w}}) = (I_3 \otimes \boldsymbol{P}^{-1}) \begin{pmatrix} -\boldsymbol{V}\boldsymbol{P}^S\boldsymbol{u}/2 + \epsilon \boldsymbol{D}_{\boldsymbol{y}}^\top \boldsymbol{P}^S\boldsymbol{u} \\ -\boldsymbol{V}\boldsymbol{P}^S\boldsymbol{v}/2 + \epsilon \boldsymbol{D}_{\boldsymbol{y}}^\top \boldsymbol{P}^S\boldsymbol{v} \\ -\boldsymbol{P}^S\boldsymbol{v} \end{pmatrix}, \tag{3.5}$$

which is a more convenient notation for the derivation of the Jacobian in Section 4.

We will show in the following section that this specific choice of penalty matrices leads to nonlinear stability. The total penalty term in (3.2) becomes

$$\boldsymbol{\mathcal{S}}(\vec{\boldsymbol{w}}) = \sum_{k\in\{W,E,S,N\}} \boldsymbol{\mathcal{S}}(\vec{\boldsymbol{w}})^k . \tag{3.6}$$

3.1. Boundedness and Stability. For completeness, we also show schematically how to obtain an energy estimate (again all details can be found in [12]). Similarly to the continuous analysis, we omit all boundaries except for the south one. By mimicking the continuous path [11], we multiply (3.2) by $2\,\vec{\boldsymbol{w}}^\top(I_3 \otimes \boldsymbol{P})$ from the left and use the SBP-property (3.1) to get

$$\frac{d}{dt}\|\,\vec{\boldsymbol{w}}\|^2_{\tilde{I}\otimes\boldsymbol{P}} + 2\epsilon\|\nabla\,\vec{\boldsymbol{w}}\|^2_{\tilde{I}\otimes\boldsymbol{P}} = \boldsymbol{BT}, \tag{3.7}$$

where $\|\nabla\,\vec{\boldsymbol{w}}\|^2_{\tilde{I}\otimes\boldsymbol{P}} = (I_3 \otimes \boldsymbol{D}_{\boldsymbol{x}}\,\vec{\boldsymbol{w}})^\top(I_3\otimes\boldsymbol{P})\tilde{\boldsymbol{I}}(I_3\otimes\boldsymbol{D}_{\boldsymbol{x}}\,\vec{\boldsymbol{w}}) + (I_3\otimes\boldsymbol{D}_{\boldsymbol{y}}\,\vec{\boldsymbol{w}})^\top(I_3\otimes\boldsymbol{P})\tilde{\boldsymbol{I}}(I_3\otimes\boldsymbol{D}_{\boldsymbol{y}}\,\vec{\boldsymbol{w}})$ is the dissipative volume term corresponding to the continuous one and

$$\begin{aligned} \boldsymbol{BT} = &\underbrace{\vec{\boldsymbol{w}}^\top(I_3\otimes\boldsymbol{P}^S)\boldsymbol{B}\,\vec{\boldsymbol{w}} - 2\epsilon\,\vec{\boldsymbol{w}}^\top(I_3\otimes\boldsymbol{P}^S)\tilde{\boldsymbol{I}}(I_3\otimes\boldsymbol{D}_{\boldsymbol{y}})\,\vec{\boldsymbol{w}}}_{I} \\ &\underbrace{+2\,\vec{\boldsymbol{w}}(I_3\otimes\boldsymbol{P})\boldsymbol{\mathcal{S}}^S(\vec{\boldsymbol{w}})}_{II} \end{aligned} \tag{3.8}$$

contains all terms evaluated at the boundary.

The semi-norm of the solution ($\|\,\vec{\boldsymbol{w}}\|^2_{\tilde{I}\otimes\boldsymbol{P}}$) is bounded if the right-hand side of (3.7) is non-positive. By expanding (3.8) and using the explicit form of $\boldsymbol{\mathcal{S}}^S(\vec{\boldsymbol{w}})$ stated in (3.4), we find

$$\begin{aligned} \boldsymbol{BT} = &\underbrace{\boldsymbol{v}^\top\boldsymbol{P}^S(\boldsymbol{U}\boldsymbol{u} + \boldsymbol{V}\boldsymbol{v} + 2\boldsymbol{p} - 2\epsilon\boldsymbol{D}_{\boldsymbol{y}}\boldsymbol{v}) - 2\epsilon\boldsymbol{u}^\top\boldsymbol{P}^S\boldsymbol{D}_{\boldsymbol{y}}\boldsymbol{u}}_{I} \\ &\underbrace{-2\boldsymbol{v}^\top\boldsymbol{P}^S(\boldsymbol{U}\boldsymbol{u}/2 + \boldsymbol{V}\boldsymbol{v}/2 + \boldsymbol{p}^\top\boldsymbol{v} - \epsilon\boldsymbol{D}_{\boldsymbol{y}}\boldsymbol{v}) + 2\epsilon\boldsymbol{u}^\top\boldsymbol{P}^S\boldsymbol{D}_{\boldsymbol{y}}\boldsymbol{u}}_{II} = 0, \end{aligned}$$

where term I is obtained from the governing equation and term II from the penalty term. As in the continuous setting, the boundary terms vanish. Integrating (3.7) in time (assuming homogeneous dissipative boundary conditions at all boundaries) leads to

$$\|\,\vec{\boldsymbol{w}}\|^2_{\tilde{I}\otimes\boldsymbol{P}}(T) + 2\epsilon\int_0^T \|\nabla\,\vec{\boldsymbol{w}}\|^2_{\tilde{I}\otimes\boldsymbol{P}}dt \leq \|\boldsymbol{f}\|^2_{\tilde{I}\otimes\boldsymbol{P}},$$

which is the semi-discrete version of the estimate in (2.5).

4. Exact computation of the Jacobian. In this section, we will explicitly compute the Jacobian of $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{S}}$ in (3.2). Let $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^n$, where $n = (N+1)(M+1)$ is the total number of grid points, be a differentiable vector function. For a given vector $\boldsymbol{u} = (u_{00}, \dots u_{NM})^\top \in \mathbb{R}^n$, $\boldsymbol{h}$ outputs the vector $\boldsymbol{h}(\boldsymbol{u}) = (h_{00}, \dots h_{NM})^\top \in \mathbb{R}^n$. The Jacobian matrix $J_{\boldsymbol{h}} \in \mathbb{R}^{n\times n}$ of $\boldsymbol{h}$ is given by

$$J_{\boldsymbol{h}} = \begin{pmatrix} \frac{\partial h_{00}}{\partial u_{00}} & \cdots & \frac{\partial h_{00}}{\partial u_{NM}} \\ \vdots & \ddots & \vdots \\ \frac{\partial h_{NM}}{\partial u_{00}} & \cdots & \frac{\partial h_{NM}}{\partial u_{NM}} \end{pmatrix}.$$

We will first derive the Jacobian of the different terms in $\boldsymbol{\mathcal{L}}(\vec{\boldsymbol{w}})$ and at the end, add the terms and state the complete result. To start, consider the vector function

$$\boldsymbol{h}(\boldsymbol{u}) = \begin{pmatrix} h_{00} \\ \vdots \\ h_{NM} \end{pmatrix} = \begin{pmatrix} u_{00} \\ \vdots \\ u_{NM} \end{pmatrix} = \boldsymbol{u}\,.$$

Since

$$\begin{array}{ccccc}
\frac{\partial h_{00}}{\partial u_{00}} = 1 & \frac{\partial h_{00}}{\partial u_{01}} = 0 & \frac{\partial h_{00}}{\partial u_{02}} = 0 & \cdots & \frac{\partial h_{00}}{\partial u_{NM}} = 0 \\
\frac{\partial h_{01}}{\partial u_{00}} = 0 & \frac{\partial h_{01}}{\partial u_{01}} = 1 & \frac{\partial h_{01}}{\partial u_{02}} = 0 & \cdots & \frac{\partial h_{01}}{\partial u_{NM}} = 0 \\
 & & \vdots & & \\
\frac{\partial h_{NM}}{\partial u_{00}} = 0 & \frac{\partial h_{NM}}{\partial u_{01}} = 0 & \frac{\partial h_{NM}}{\partial u_{02}} = 0 & \cdots & \frac{\partial h_{NM}}{\partial u_{NM}} = 1\,,
\end{array}$$

the Jacobian of $\boldsymbol{h}(\boldsymbol{u}) = \boldsymbol{u}$ becomes $J_{\boldsymbol{u}} = \boldsymbol{I}$. Now let

$$\begin{aligned} \boldsymbol{h}(\boldsymbol{u}) = \begin{pmatrix} h_{00} \\ \vdots \\ h_{NM} \end{pmatrix} = \boldsymbol{D_x u} &= \begin{pmatrix} D_{0,0} & \cdots & D_{0,NM} \\ \vdots & \ddots & \vdots \\ D_{NM,0} & \cdots & D_{NM,MN} \end{pmatrix} \begin{pmatrix} u_{00} \\ \vdots \\ u_{NM} \end{pmatrix} \\ &= \begin{pmatrix} D_{0,0}u_{00} + \cdots + D_{0,NM}u_{NM} \\ \vdots \\ D_{NM,0}u_{00} + \cdots + D_{NM,MN}u_{NM} \end{pmatrix}. \end{aligned}$$

Then, in the same way

$$\begin{array}{ccccc}
\frac{\partial h_{00}}{\partial u_{00}} = D_{0,0} & \frac{\partial h_{00}}{\partial u_{01}} = D_{0,1} & \frac{\partial h_{00}}{\partial u_{02}} = D_{0,2} & \cdots & \frac{\partial h_{00}}{\partial u_{NM}} = D_{0,NM} \\
\frac{\partial h_{01}}{\partial u_{00}} = D_{1,0} & \frac{\partial h_{01}}{\partial u_{01}} = D_{1,1} & \frac{\partial h_{01}}{\partial u_{02}} = D_{1,2} & \cdots & \frac{\partial h_{01}}{\partial u_{NM}} = D_{1,NM} \\
 & & \vdots & & \\
\frac{\partial h_{NM}}{\partial u_{00}} = D_{NM,0} & \frac{\partial h_{NM}}{\partial u_{01}} = D_{NM,1} & \frac{\partial h_{NM}}{\partial u_{02}} = D_{NM,2} & \cdots & \frac{\partial h_{NM}}{\partial u_{NM}} = D_{NM,NM}\,.
\end{array}$$

Thus, $J_{\boldsymbol{D_x u}} = \boldsymbol{D_x}$.

To derive the Jacobian of the nonlinear term, $\boldsymbol{U}\boldsymbol{D_x}\boldsymbol{u}$, we let

$$\boldsymbol{h}(\boldsymbol{u}) = \boldsymbol{U}\boldsymbol{D_x}\boldsymbol{u} = \begin{pmatrix} u_{00}[D_{0,0}u_{00} + \dots + D_{0,NM}u_{NM}] \\ \vdots \\ u_{NM}[D_{NM,0}u_{00} + \dots + D_{NM,MN}u_{NM}] \end{pmatrix} = \begin{pmatrix} u_{00}(\boldsymbol{D_x}\boldsymbol{u})_{00} \\ \vdots \\ u_{NM}(\boldsymbol{D_x}\boldsymbol{u})_{NM} \end{pmatrix}.$$

By using the product rule, we get that

$$\begin{array}{llll}
\dfrac{\partial h_{00}}{\partial u_{00}} = u_{00}D_{0,0} + (\boldsymbol{D_x}\boldsymbol{u})_{00} & \dfrac{\partial h_{00}}{\partial u_{01}} = u_{00}D_{0,1} & \dots & \dfrac{\partial h_{00}}{\partial u_{NM}} = u_{00}D_{0,NM} \\
\dfrac{\partial h_{01}}{\partial u_{00}} = u_{01}D_{1,0} & \dfrac{\partial h_{01}}{\partial u_{01}} = u_{01}D_{1,1} + (\boldsymbol{D_x}\boldsymbol{u})_{01} & \dots & \dfrac{\partial h_{01}}{\partial u_{NM}} = u_{00}D_{0,NM} \\
\quad\vdots & & & \\
\dfrac{\partial h_{NM}}{\partial u_{00}} = u_{NM}D_{NM,0} & \dfrac{\partial h_{NM}}{\partial u_{01}} = u_{NM}D_{NM,1} & \dots & \dfrac{\partial h_{NM}}{\partial u_{NM}} = u_{NM}D_{NM,NM} + (\boldsymbol{D_x}\boldsymbol{u})_{NM}.
\end{array}$$

Hence,

$$\begin{aligned}
J_{\boldsymbol{U}\boldsymbol{D_x}\boldsymbol{u}} &= \begin{pmatrix} u_{00}D_{0,0} + (\boldsymbol{D_x}\boldsymbol{u})_{00} & u_{00}D_{0,1} & \dots & u_{00}D_{0,NM} \\ u_{01}D_{1,0} & u_{01}D_{1,1} + (\boldsymbol{D_x}\boldsymbol{u})_{01} & \dots & u_{01}D_{1,NM} \\ & & \vdots & \\ u_{NM}D_{NM,0} & u_{NM}D_{NM,1} & \dots & u_{NM}D_{NM,NM} + (\boldsymbol{D_x}\boldsymbol{u})_{NM} \end{pmatrix} \\
&= \underbrace{\begin{pmatrix} u_{00}D_{0,0} & u_{00}D_{0,1} & \dots & u_{00}D_{0,NM} \\ u_{01}D_{1,0} & u_{01}D_{1,1} & \dots & u_{01}D_{1,NM} \\ & & \vdots & \\ u_{NM}D_{NM,0} & u_{NM}D_{NM,1} & \dots & u_{NM}D_{NM,NM} \end{pmatrix}}_{\boldsymbol{U}\boldsymbol{D_x}} \\
&\quad + \underbrace{\begin{pmatrix} (\boldsymbol{D_x}\boldsymbol{u})_{00} & 0 & \dots & 0 \\ 0 & (\boldsymbol{D_x}\boldsymbol{u})_{01} & \dots & 0 \\ & & \vdots & \\ 0 & 0 & \dots & (\boldsymbol{D_x}\boldsymbol{u})_{NM} \end{pmatrix}}_{\underline{\boldsymbol{D_x}\boldsymbol{u}}} = \boldsymbol{U}\boldsymbol{D_x} + \underline{\boldsymbol{D_x}\boldsymbol{u}},
\end{aligned}$$

where $\underline{\boldsymbol{D_x}\boldsymbol{u}} = \mathrm{diag}(\boldsymbol{D_x}\boldsymbol{u})$.

In a similar manner, for

$$\boldsymbol{h}(\boldsymbol{u}) = \boldsymbol{D_x}\boldsymbol{U}\boldsymbol{u} = \begin{pmatrix} D_{0,0}u_{00}^2 + \dots + D_{0,NM}u_{NM}^2 \\ \vdots \\ D_{NM,0}u_{00}^2 + \dots + D_{NM,MN}u_{NM}^2 \end{pmatrix}$$

we get that

$$\begin{aligned}
&\frac{\partial h_{00}}{\partial u_{00}} = 2D_{0,0}u_{00} && \frac{\partial h_{00}}{\partial u_{01}} = 2D_{0,1}u_{01} \quad \dots \quad && \frac{\partial h_{00}}{\partial u_{NM}} = 2D_{0,NM}u_{NM}\\
&\frac{\partial h_{01}}{\partial u_{00}} = 2D_{1,0}u_{00} && \frac{\partial h_{01}}{\partial u_{01}} = 2D_{1,1}u_{01} \quad \dots \quad && \frac{\partial h_{01}}{\partial u_{NM}} = 2D_{1,NM}u_{NM}\\
&\quad\vdots && && \\
&\frac{\partial h_{NM}}{\partial u_{00}} = 2D_{NM,0}u_{00} && \frac{\partial h_{NM}}{\partial u_{01}} = 2D_{NM,1}u_{01} \quad \dots \quad && \frac{\partial h_{01}}{\partial u_{NM}} = 2D_{NM,NM}u_{NM}\,.
\end{aligned}$$

Hence, $J_{\boldsymbol{D_x U u}} = 2\boldsymbol{D_x U}$. To summarize, we have shown that

$$J_{\boldsymbol{u}} = \boldsymbol{I}, \quad J_{\boldsymbol{D_x u}} = \boldsymbol{D_x}, \quad J_{\boldsymbol{U D_x u}} = \boldsymbol{U D_x} + \underline{\boldsymbol{D_x u}}, \quad J_{\boldsymbol{D_x U u}} = 2\boldsymbol{D_x U}\,. \tag{4.1}$$

4.1. The Jacobian of the spatial operator. Having established these building blocks, we next consider the terms in $\boldsymbol{\mathcal{L}}(\vec{\boldsymbol{w}})$. Since these terms have a block structure, their Jacobians will have that as well. Let $\boldsymbol{h}^1, \boldsymbol{h}^2, \boldsymbol{h}^3 : \mathbb{R}^{3n} \to \mathbb{R}^n$ be differentiable functions of $\vec{\boldsymbol{w}}$ and define $\tilde{\boldsymbol{h}} : \mathbb{R}^{3n} \to \mathbb{R}^{3n}$ given by

$$\tilde{\boldsymbol{h}}(\vec{\boldsymbol{w}}) = \begin{pmatrix} \boldsymbol{h}^1(\vec{\boldsymbol{w}}) \\ \boldsymbol{h}^2(\vec{\boldsymbol{w}}) \\ \boldsymbol{h}^3(\vec{\boldsymbol{w}}) \end{pmatrix}.$$

Since

$$\boldsymbol{h}^1 = \begin{pmatrix} h^1_{00} \\ h^1_{01} \\ \vdots \\ h^1_{NM} \end{pmatrix} \quad \text{and} \quad \vec{\boldsymbol{w}} = \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \\ \boldsymbol{p} \end{pmatrix}$$

it follows that

$$J_{h^1_{00}} = \frac{\partial h^1_{00}}{\partial \vec{\boldsymbol{w}}} = \begin{pmatrix} \frac{\partial h^1_{00}}{\partial w_{00}} & \dots & \frac{\partial h^1_{00}}{\partial w_{3NM}} \end{pmatrix} = \begin{pmatrix} \frac{\partial h^1_{00}}{\partial \boldsymbol{u}} & \frac{\partial h^1_{00}}{\partial \boldsymbol{v}} & \frac{\partial h^1_{00}}{\partial \boldsymbol{p}} \end{pmatrix} \in \mathbb{R}^{1\times 3n}$$

and similarly for every element in $\boldsymbol{h}^1$. Therefore, the Jacobian of $\boldsymbol{h}^1$ can be expressed as

$$J_{\boldsymbol{h}^1} = \frac{\partial \boldsymbol{h}^1}{\partial \vec{\boldsymbol{w}}} = \begin{pmatrix} \frac{\partial h^1_{00}}{\partial \boldsymbol{u}} & \frac{\partial h^1_{00}}{\partial \boldsymbol{v}} & \frac{\partial h^1_{00}}{\partial \boldsymbol{p}} \\ \frac{\partial h^1_{01}}{\partial \boldsymbol{u}} & \frac{\partial h^1_{01}}{\partial \boldsymbol{v}} & \frac{\partial h^1_{01}}{\partial \boldsymbol{p}} \\ \vdots & \vdots & \vdots \\ \frac{\partial h^1_{NM}}{\partial \boldsymbol{u}} & \frac{\partial h^1_{NM}}{\partial \boldsymbol{v}} & \frac{\partial h^1_{NM}}{\partial \boldsymbol{p}} \end{pmatrix} = \begin{pmatrix} \frac{\partial \boldsymbol{h}^1}{\partial \boldsymbol{u}} & \frac{\partial \boldsymbol{h}^1}{\partial \boldsymbol{v}} & \frac{\partial \boldsymbol{h}^1}{\partial \boldsymbol{p}} \end{pmatrix} \in \mathbb{R}^{n\times 3n}\,.$$

The same holds for $\boldsymbol{h}^2$ and $\boldsymbol{h}^3$. Thus, the Jacobian of $\tilde{\boldsymbol{h}}$ is given by

$$J_{\tilde{\boldsymbol{h}}} = \begin{pmatrix} \frac{\partial \boldsymbol{h}^1}{\partial \boldsymbol{u}} & \frac{\partial \boldsymbol{h}^1}{\partial \boldsymbol{v}} & \frac{\partial \boldsymbol{h}^1}{\partial \boldsymbol{p}} \\ \frac{\partial \boldsymbol{h}^2}{\partial \boldsymbol{u}} & \frac{\partial \boldsymbol{h}^2}{\partial \boldsymbol{v}} & \frac{\partial \boldsymbol{h}^2}{\partial \boldsymbol{p}} \\ \frac{\partial \boldsymbol{h}^3}{\partial \boldsymbol{u}} & \frac{\partial \boldsymbol{h}^3}{\partial \boldsymbol{v}} & \frac{\partial \boldsymbol{h}^3}{\partial \boldsymbol{p}} \end{pmatrix} \in \mathbb{R}^{3n\times 3n}$$

and each block in $J_{\tilde{\boldsymbol{h}}}$ is of size $n \times n$.

For the first term in $\boldsymbol{\mathcal{L}}(\vec{\boldsymbol{w}})$, $\boldsymbol{A}(I_3 \otimes \boldsymbol{D_x})\vec{\boldsymbol{w}}$, we get

$$\tilde{\boldsymbol{h}}(\vec{\boldsymbol{w}}) = \begin{pmatrix} \boldsymbol{h}^1(\vec{\boldsymbol{w}}) \\ \boldsymbol{h}^2(\vec{\boldsymbol{w}}) \\ \boldsymbol{h}^3(\vec{\boldsymbol{w}}) \end{pmatrix} = \boldsymbol{A}(I_3 \otimes \boldsymbol{D_x})\vec{\boldsymbol{w}} = \begin{pmatrix} \boldsymbol{U}\boldsymbol{D_x}\boldsymbol{u} + \boldsymbol{D_x}\boldsymbol{p} \\ \boldsymbol{U}\boldsymbol{D_x}\boldsymbol{v} \\ \boldsymbol{D_x}\boldsymbol{u} \end{pmatrix} = \begin{pmatrix} \boldsymbol{U}\boldsymbol{D_x}\boldsymbol{u} + \boldsymbol{D_x}\boldsymbol{p} \\ \underline{\boldsymbol{D_x}\boldsymbol{v}}\boldsymbol{u} \\ \underline{\boldsymbol{D_x}}\boldsymbol{u} \end{pmatrix}.$$

The last identities are useful when deriving $J_{\boldsymbol{A}(I_3\otimes \boldsymbol{D_x})\vec{\boldsymbol{w}}}$. By using (4.1), we get that

$$\begin{aligned}
&\frac{\partial \boldsymbol{h}^1}{\partial \boldsymbol{u}} = \boldsymbol{U}\boldsymbol{D_x} + \underline{\boldsymbol{D_x}\boldsymbol{u}} && \frac{\partial \boldsymbol{h}^1}{\partial \boldsymbol{v}} = \boldsymbol{0} && \frac{\partial \boldsymbol{h}^1}{\partial \boldsymbol{p}} = \boldsymbol{D_x} \\
&\frac{\partial \boldsymbol{h}^2}{\partial \boldsymbol{u}} = \underline{\boldsymbol{D_x}\boldsymbol{v}} && \frac{\partial \boldsymbol{h}^2}{\partial \boldsymbol{v}} = \boldsymbol{U}\boldsymbol{D_x} && \frac{\partial \boldsymbol{h}^2}{\partial \boldsymbol{p}} = \boldsymbol{0} \\
&\frac{\partial \boldsymbol{h}^3}{\partial \boldsymbol{u}} = \boldsymbol{D_x} && \frac{\partial \boldsymbol{h}^3}{\partial \boldsymbol{v}} = \boldsymbol{0} && \frac{\partial \boldsymbol{h}^3}{\partial \boldsymbol{p}} = \boldsymbol{0}.
\end{aligned}$$

Thus,

$$J_{\boldsymbol{A}(I_3\otimes \boldsymbol{D_x})\vec{\boldsymbol{w}}} = \begin{pmatrix} \boldsymbol{U}\boldsymbol{D_x} + \underline{\boldsymbol{D_x}\boldsymbol{u}} & \boldsymbol{0} & \boldsymbol{D_x} \\ \underline{\boldsymbol{D_x}\boldsymbol{v}} & \boldsymbol{U}\boldsymbol{D_x} & \boldsymbol{0} \\ \boldsymbol{D_x} & \boldsymbol{0} & \boldsymbol{0} \end{pmatrix}.$$

Likewise for the second term in $\boldsymbol{\mathcal{L}}(\vec{\boldsymbol{w}})$, $(I_3 \otimes \boldsymbol{D_x})\boldsymbol{A}\vec{\boldsymbol{w}}$, note that

$$(I_3 \otimes \boldsymbol{D_x})\boldsymbol{A}\vec{\boldsymbol{w}} = \begin{pmatrix} \boldsymbol{D_x}\boldsymbol{U}\boldsymbol{u} + \boldsymbol{D_x}\boldsymbol{p} \\ \boldsymbol{D_x}\boldsymbol{U}\boldsymbol{v} \\ \boldsymbol{D_x}\boldsymbol{u} \end{pmatrix} = \begin{pmatrix} \boldsymbol{D_x}\boldsymbol{U}\boldsymbol{u} + \boldsymbol{D_x}\boldsymbol{p} \\ \boldsymbol{D_x}\boldsymbol{V}\boldsymbol{u} \\ \boldsymbol{D_x}\boldsymbol{u} \end{pmatrix},$$

where we have used that $\boldsymbol{V}\boldsymbol{u} = \boldsymbol{U}\boldsymbol{v}$. Hence,

$$J_{(I_3\otimes \boldsymbol{D_x})\boldsymbol{A}\vec{\boldsymbol{w}}} = \begin{pmatrix} 2\boldsymbol{D_x}\boldsymbol{U} & \boldsymbol{0} & \boldsymbol{D_x} \\ \boldsymbol{D_x}\boldsymbol{V} & \boldsymbol{D_x}\boldsymbol{U} & \boldsymbol{0} \\ \boldsymbol{D_x} & \boldsymbol{0} & \boldsymbol{0} \end{pmatrix}.$$

The next two terms in $\boldsymbol{\mathcal{L}}(\vec{\boldsymbol{w}})$ are treated in a similar manner and we get that

$$\begin{aligned}
J_{\boldsymbol{B}(I_3\otimes \boldsymbol{D_y})\vec{\boldsymbol{w}}} &= \begin{pmatrix} \boldsymbol{V}\boldsymbol{D_y} & \overline{\boldsymbol{D_y}\boldsymbol{u}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{V}\boldsymbol{D_y} + \underline{\boldsymbol{D_y}\boldsymbol{v}} & \boldsymbol{D_y} \\ \boldsymbol{0} & \boldsymbol{D_y} & \boldsymbol{0} \end{pmatrix} \\
J_{(I_3\otimes \boldsymbol{D_y})\boldsymbol{B}\vec{\boldsymbol{w}}} &= \begin{pmatrix} \boldsymbol{D_y}\boldsymbol{V} & \boldsymbol{D_y}\boldsymbol{U} & \boldsymbol{0} \\ \boldsymbol{0} & 2\boldsymbol{D_y}\boldsymbol{V} & \boldsymbol{D_y} \\ \boldsymbol{0} & \boldsymbol{D_y} & \boldsymbol{0} \end{pmatrix}.
\end{aligned}$$

Finally, the contribution to the Jacobian of the linear viscous terms simply becomes

$$J_{-\epsilon\tilde{\boldsymbol{I}}[(I_3\otimes \boldsymbol{D_x})^2+(I_3\otimes \boldsymbol{D_y})^2]\vec{\boldsymbol{w}}} = -\begin{pmatrix} \epsilon(\boldsymbol{D_x}^2 + \boldsymbol{D_y}^2) & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \epsilon(\boldsymbol{D_x}^2 + \boldsymbol{D_y}^2) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{pmatrix}.$$

Adding all terms proves the following proposition, which is the first of the two main results of this paper.

Proposition 4.1. *The Jacobian $J_{\boldsymbol{\mathcal{L}}}$ of the discrete operator $\boldsymbol{\mathcal{L}}$ in (3.2) is*

$$J_{\boldsymbol{\mathcal{L}}} = \begin{pmatrix} J_{11} & \frac{1}{2}\left(\underline{\boldsymbol{D_y u}} + \boldsymbol{D_y U}\right) & \boldsymbol{D_x} \\ \frac{1}{2}\left(\underline{\boldsymbol{D_x v}} + \boldsymbol{D_x V}\right) & J_{22} & \boldsymbol{D_y} \\ \boldsymbol{D_x} & \boldsymbol{D_y} & \boldsymbol{0} \end{pmatrix} \tag{4.2}$$

where

$$J_{11} = \frac{1}{2}\left(\boldsymbol{U D_x} + \underline{\boldsymbol{D_x u}} + 2\boldsymbol{D_x U} + \boldsymbol{V D_y} + \boldsymbol{D_y V}\right) - \epsilon(\boldsymbol{D_x^2} + \boldsymbol{D_y^2})$$

$$J_{22} = \frac{1}{2}\left(\boldsymbol{V D_y} + \underline{\boldsymbol{D_y v}} + 2\boldsymbol{D_y V} + \boldsymbol{U D_x} + \boldsymbol{D_x U}\right) - \epsilon(\boldsymbol{D_x^2} + \boldsymbol{D_y^2}).$$

4.2. The Jacobian of the penalty terms.

By following the procedure presented above, we next derive the Jacobian for $\boldsymbol{\mathcal{S}}(\vec{\boldsymbol{w}})$. To start, we rewrite $\boldsymbol{\mathcal{S}}^S(\vec{\boldsymbol{w}})$ as

$$\boldsymbol{\mathcal{S}}^S(\vec{\boldsymbol{w}}) = \begin{pmatrix} \boldsymbol{\mathcal{S}}_1^S \\ \boldsymbol{\mathcal{S}}_2^S \\ \boldsymbol{\mathcal{S}}_3^S \end{pmatrix} = (I_3 \otimes \boldsymbol{P}^{-1}) \begin{pmatrix} -\boldsymbol{V P}^S \boldsymbol{u}/2 + \epsilon \boldsymbol{D_y}^\top \boldsymbol{P}^S \boldsymbol{u} \\ -\boldsymbol{V P}^S \boldsymbol{v}/2 + \epsilon \boldsymbol{D_y}^\top \boldsymbol{P}^S \boldsymbol{v} \\ -\boldsymbol{P}^S \boldsymbol{v} \end{pmatrix} \in \mathbb{R}^{3n}. \tag{4.3}$$

The Jacobian of $\boldsymbol{\mathcal{S}}^S(\vec{\boldsymbol{w}})$ is

$$J_{\boldsymbol{\mathcal{S}}^S} = \begin{pmatrix} \frac{\partial \boldsymbol{\mathcal{S}}_1^S}{\partial \boldsymbol{u}} & \frac{\partial \boldsymbol{\mathcal{S}}_1^S}{\partial \boldsymbol{v}} & \frac{\partial \boldsymbol{\mathcal{S}}_1^S}{\partial \boldsymbol{p}} \\ \frac{\partial \boldsymbol{\mathcal{S}}_2^S}{\partial \boldsymbol{u}} & \frac{\partial \boldsymbol{\mathcal{S}}_2^S}{\partial \boldsymbol{v}} & \frac{\partial \boldsymbol{\mathcal{S}}_2^S}{\partial \boldsymbol{p}} \\ \frac{\partial \boldsymbol{\mathcal{S}}_3^S}{\partial \boldsymbol{u}} & \frac{\partial \boldsymbol{\mathcal{S}}_3^S}{\partial \boldsymbol{v}} & \frac{\partial \boldsymbol{\mathcal{S}}_3^S}{\partial \boldsymbol{p}} \end{pmatrix} \in \mathbb{R}^{3n \times 3n}.$$

The first block in $J_{\boldsymbol{\mathcal{S}}^S}$ becomes

$$\frac{\partial \boldsymbol{\mathcal{S}}_1^S}{\partial \boldsymbol{u}} = \Big(\underbrace{-\frac{\partial}{\partial \boldsymbol{u}} \boldsymbol{P}^{-1} \boldsymbol{V P}^S \boldsymbol{u}/2}_{=\boldsymbol{P}^{-1}\boldsymbol{V P}^S/2} + \underbrace{\frac{\partial}{\partial \boldsymbol{u}} \epsilon \boldsymbol{P}^{-1} \boldsymbol{D_y}^\top \boldsymbol{P}^S \boldsymbol{u}}_{=\epsilon \boldsymbol{P}^{-1}\boldsymbol{D_y}^\top \boldsymbol{P}^S} \Big) = \boldsymbol{P}^{-1}\left(-\boldsymbol{V}/2 + \epsilon \boldsymbol{D_y}^\top\right)\boldsymbol{P}^S.$$

Since $\boldsymbol{P}^S$ is diagonal, we have $\boldsymbol{V P}^s \boldsymbol{u} = \boldsymbol{U P}^S \boldsymbol{v}$ and the second block is

$$\frac{\partial \boldsymbol{\mathcal{S}}_1^S}{\partial \boldsymbol{v}} = \Big(\underbrace{-\frac{\partial}{\partial \boldsymbol{v}} \boldsymbol{P}^{-1} \boldsymbol{U P}^S \boldsymbol{v}/2}_{=\boldsymbol{P}^{-1}\boldsymbol{U P}^S/2} + \underbrace{\frac{\partial}{\partial \boldsymbol{v}} \epsilon \boldsymbol{P}^{-1} \boldsymbol{D_x}^\top \boldsymbol{P}^S \boldsymbol{u}}_{=\boldsymbol{0}} \Big) = -\boldsymbol{P}^{-1}\boldsymbol{U P}^S/2.$$

Note that $\boldsymbol{\mathcal{S}}^S$ does not depend on $\boldsymbol{p}$ and also that $\boldsymbol{\mathcal{S}}_2^S$ and $\boldsymbol{\mathcal{S}}_3^S$ are both independent of $\boldsymbol{u}$. Hence, the remaining non-zero blocks of $J_{\boldsymbol{\mathcal{S}}^S}$ are

$$\frac{\partial \boldsymbol{\mathcal{S}}_2^W}{\partial \boldsymbol{v}} = \boldsymbol{P}^{-1}(-\boldsymbol{V} + \epsilon \boldsymbol{D_y}^\top)\boldsymbol{P}^S, \qquad \frac{\partial \boldsymbol{\mathcal{S}}_3^W}{\partial \boldsymbol{v}} = -\boldsymbol{P}^{-1}\boldsymbol{P}^S,$$

where we have used that $\boldsymbol{V P}^s \boldsymbol{v} = \boldsymbol{P}^s \boldsymbol{V v}$. Therefore,

$$J_{\boldsymbol{\mathcal{S}}^S} = (I_3 \otimes \boldsymbol{P}^{-1}) \begin{pmatrix} -\boldsymbol{V}/2 + \epsilon \boldsymbol{D_y}^\top & -\boldsymbol{U}/2 & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{V} + \epsilon \boldsymbol{D_y}^\top & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{I} & \boldsymbol{0} \end{pmatrix} (I_3 \otimes \boldsymbol{P}^S).$$

For non-homogeneous boundary conditions, the boundary data $\boldsymbol{g}$ will affect the Jacobian if the SAT:s are nonlinear with respect to $\vec{\boldsymbol{w}}$. We illustrate this by considering $\boldsymbol{\mathcal{S}}_1^W$ (the first block in $\boldsymbol{\mathcal{S}}^W$), which we rewrite in a similar manner as we did for $\boldsymbol{\mathcal{S}}_1^S$ in (4.3) and get

$$\begin{aligned}\boldsymbol{\mathcal{S}}_1^W(\vec{\boldsymbol{w}}) &= \boldsymbol{P}^{-1}(-\boldsymbol{U}/2+\epsilon\boldsymbol{D}_{\boldsymbol{x}}^\top)\boldsymbol{P}^W(\boldsymbol{u}-\boldsymbol{g}_1)\\ &= -\boldsymbol{P}^{-1}\boldsymbol{U}\boldsymbol{P}^W(\boldsymbol{u}-\boldsymbol{g}_1)/2+\epsilon\boldsymbol{P}^{-1}\boldsymbol{D}_{\boldsymbol{x}}^\top\boldsymbol{P}^W(\boldsymbol{u}-\boldsymbol{g}_1)\,.\end{aligned}$$

Note that the terms $-\boldsymbol{P}^{-1}\boldsymbol{U}\boldsymbol{P}^W(\boldsymbol{u}-\boldsymbol{g}_1)/2$ and $\epsilon\boldsymbol{P}^{-1}\boldsymbol{D}_{\boldsymbol{x}}^\top\boldsymbol{P}^W(\boldsymbol{u}-\boldsymbol{g}_1)$ are nonlinear and linear with respect to $\vec{\boldsymbol{w}}$ (via $\boldsymbol{u}$), respectively. The Jacobian to the linear term simply becomes

$$\frac{\partial}{\partial\boldsymbol{u}}\left(\epsilon\boldsymbol{P}^{-1}\boldsymbol{D}_{\boldsymbol{x}}^\top\boldsymbol{P}^W(\boldsymbol{u}-\boldsymbol{g}_1)\right) = \underbrace{\frac{\partial}{\partial\boldsymbol{u}}\left(\epsilon\boldsymbol{P}^{-1}\boldsymbol{D}_{\boldsymbol{x}}^\top\boldsymbol{P}^W\boldsymbol{u}\right)}_{=\epsilon\boldsymbol{P}^{-1}\boldsymbol{D}_{\boldsymbol{x}}^\top\boldsymbol{P}^W} - \underbrace{\frac{\partial}{\partial\boldsymbol{u}}\left(\epsilon\boldsymbol{P}^{-1}\boldsymbol{D}_{\boldsymbol{x}}^\top\boldsymbol{P}^W\boldsymbol{g}_1\right)}_{=0}\,.$$

For the nonlinear term, we use that $\boldsymbol{U}\boldsymbol{P}^W=\boldsymbol{P}^W\boldsymbol{U}$ and $\boldsymbol{U}\boldsymbol{g}_1=\underline{\boldsymbol{g}_1}\boldsymbol{u}$, which yield

$$\begin{aligned}-\frac{\partial}{\partial\boldsymbol{u}}\left(\boldsymbol{P}^{-1}\boldsymbol{P}^W\boldsymbol{U}(\boldsymbol{u}-\boldsymbol{g}_1)/2\right) &= \underbrace{-\frac{\partial}{\partial\boldsymbol{u}}\left(\boldsymbol{P}^{-1}\boldsymbol{P}^W\boldsymbol{U}\boldsymbol{u})/2\right)}_{=-\boldsymbol{P}^{-1}\boldsymbol{P}^W\boldsymbol{U}} + \underbrace{\frac{\partial}{\partial\boldsymbol{u}}\left(\boldsymbol{P}^{-1}\boldsymbol{P}^W\underline{\boldsymbol{g}_1}\boldsymbol{u})/2\right)}_{=\boldsymbol{P}^{-1}\boldsymbol{P}^W\underline{\boldsymbol{g}_1}/2}\\ &= \boldsymbol{P}^{-1}(-\boldsymbol{U}+\underline{\boldsymbol{g}_1}/2)\boldsymbol{P}^W\end{aligned}$$

Since $\boldsymbol{\mathcal{S}}_1^W$ is independent of both $\boldsymbol{v}$ and $\boldsymbol{p}$, its Jacobian becomes

$$J_{\boldsymbol{\mathcal{S}}_1^W}(\vec{\boldsymbol{w}}) = \begin{pmatrix}\boldsymbol{P}^{-1}(-(\boldsymbol{U}-\underline{\boldsymbol{g}_1}/2)+\epsilon\boldsymbol{D}_{\boldsymbol{x}}^\top)\boldsymbol{P}^W & \boldsymbol{0} & \boldsymbol{0}\end{pmatrix}\in\mathbb{R}^{n\times 3n}$$

The Jacobian of the other penalty terms are derived in a similar manner and we have therefore proved the second main result of this paper.

Proposition 4.2. *The Jacobian of the total penalty term* (3.6) *is*

$$J_{\boldsymbol{\mathcal{S}}}(\vec{\boldsymbol{w}}) = \sum_{k\in\{W,E,S,N\}} J_{\boldsymbol{\mathcal{S}}^k}(\vec{\boldsymbol{w}})\,, \tag{4.4}$$

where

$$\begin{aligned}
J_{\boldsymbol{\mathcal{S}}^W}(\vec{\boldsymbol{w}}) &= (I_3\otimes\boldsymbol{P}^{-1})\begin{pmatrix}-(\boldsymbol{U}-\underline{\boldsymbol{g}_1}/2)+\epsilon\boldsymbol{D}_{\boldsymbol{x}}^\top & \boldsymbol{0} & \boldsymbol{0}\\ -(\boldsymbol{V}-\underline{\boldsymbol{g}_2})/2 & -\boldsymbol{U}/2+\epsilon\boldsymbol{D}_{\boldsymbol{x}}^\top & \boldsymbol{0}\\ -\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0}\end{pmatrix}(I_3\otimes\boldsymbol{P}^W)\\
J_{\boldsymbol{\mathcal{S}}^E}(\vec{\boldsymbol{w}}) &= (I_3\otimes\boldsymbol{P}^{-1}\boldsymbol{P}^E)\begin{pmatrix}-\epsilon\boldsymbol{D}_{\boldsymbol{x}} & \boldsymbol{0} & \boldsymbol{I}\\ \boldsymbol{0} & -\epsilon\boldsymbol{D}_{\boldsymbol{x}} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}\end{pmatrix}\\
J_{\boldsymbol{\mathcal{S}}^S}(\vec{\boldsymbol{w}}) &= (I_3\otimes\boldsymbol{P}^{-1}\boldsymbol{P}^S)\begin{pmatrix}-\boldsymbol{V}/2+\epsilon\boldsymbol{D}_{\boldsymbol{y}}^\top & -\boldsymbol{U}/2 & \boldsymbol{0}\\ \boldsymbol{0} & -\boldsymbol{V}+\epsilon\boldsymbol{D}_{\boldsymbol{y}}^\top & \boldsymbol{0}\\ \boldsymbol{0} & -\boldsymbol{I} & \boldsymbol{0}\end{pmatrix}(I_3\otimes\boldsymbol{P}^S)\\
J_{\boldsymbol{\mathcal{S}}^N}(\vec{\boldsymbol{w}}) &= (I_3\otimes\boldsymbol{P}^{-1}\boldsymbol{P}^N)\begin{pmatrix}-\epsilon\boldsymbol{D}_{\boldsymbol{y}} & \boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & -\epsilon\boldsymbol{D}_{\boldsymbol{y}} & \boldsymbol{I}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}\end{pmatrix}.
\end{aligned}$$

Remark 4.3. We see from Proposition 4.1 and Proposition 4.2 that parts of the blocks in the Jacobian of both $J_{\boldsymbol{\mathcal{L}}}$ and $J_{\boldsymbol{\mathcal{S}}}$ are obtained directly from the construction of $\boldsymbol{\mathcal{L}}$. The few remaining parts are obtained by i) matrix multiplications between a diagonal matrix and a non-diagonal one, for example $\boldsymbol{U}\boldsymbol{D_x}$ and ii) matrix additions. This leads to few new additional operations and hence efficiency.

5. The fully discrete scheme. To evolve the system (3.2) in time, we will for simplicity and ease of explanation use the implicit Backward Euler method. More accurate and efficient methods could be used in the same manner in practice. For an ordinary differential system of equations of the form

$$\mathcal{M}\boldsymbol{\phi}_t + \boldsymbol{\mathcal{H}}(\boldsymbol{\phi}) = 0\,,$$

where $\boldsymbol{\phi}$ is a function defined on the grid and $\mathcal{M}$ is a constant matrix, the backward Euler schemes becomes

$$\frac{\mathcal{M}(\boldsymbol{\phi}^{i+1} - \boldsymbol{\phi}^{i})}{\Delta t} + \boldsymbol{\mathcal{H}}(\boldsymbol{\phi}^{i+1}) = 0\,. \tag{5.1}$$

In (5.1), Δt is the size of the time step and the superindices i and $i+1$ are the solution at time level i and $i+1$, respectively.

In order to obtain $\boldsymbol{\phi}^{i+1}$, the system of nonlinear equations in (5.1) must be solved. One strategy is to first form the function in (1.1), which results in

$$\boldsymbol{\mathcal{F}}(\boldsymbol{\phi}^{i+1}) = \frac{\mathcal{M}(\boldsymbol{\phi}^{i+1} - \boldsymbol{\phi}^{i})}{\Delta t} + \boldsymbol{\mathcal{H}}(\boldsymbol{\phi}^{i+1})\,. \tag{5.2}$$

If we find a vector $\boldsymbol{\phi}^*$ such that $\boldsymbol{\mathcal{F}}(\boldsymbol{\phi}^*) = 0$, then $\boldsymbol{\phi}^{i+1} = \boldsymbol{\phi}^*$. To solve (5.2), we employ Newton's method [15], which is described in Algorithm 5.1. This allows us to solve a sequence of linear systems of equations and arrive at an approximation of $\boldsymbol{\phi}^{i+1}$.

Algorithm 5.1 Newton's method

```
  Input: φ^0 and tolerance tol
 Output: An approximation of φ*, where F(φ*) = 0
for j = 0, 1, 2, ... do
   solve J_F(φ^j) h^j = −F(φ^j)
   set φ^{j+1} = φ^j + h^j
   if ‖F(φ^{j+1})‖ < tol then
      return φ^{j+1}
   end if
end for
```

For the INS equations, $\boldsymbol{\phi} = \vec{\boldsymbol{w}}$, $\boldsymbol{\mathcal{H}}(\boldsymbol{\phi}) = \boldsymbol{\mathcal{L}}(\boldsymbol{\phi}) - \boldsymbol{\mathcal{S}}(\boldsymbol{\phi})$, and $\mathcal{M} = \tilde{\boldsymbol{I}}$. Hence, (5.2) becomes

$$\boldsymbol{\mathcal{F}}(\,\vec{\boldsymbol{w}}^{i+1}) = \frac{1}{\Delta t}\left(\begin{bmatrix}\boldsymbol{u}^{i+1}\\ \boldsymbol{v}^{i+1}\\ \boldsymbol{0}\end{bmatrix} - \begin{bmatrix}\boldsymbol{u}^{i}\\ \boldsymbol{v}^{i}\\ \boldsymbol{0}\end{bmatrix}\right) + \boldsymbol{\mathcal{L}}(\,\vec{\boldsymbol{w}}^{i+1}) - \boldsymbol{\mathcal{S}}(\,\vec{\boldsymbol{w}}^{i+1})\,. \tag{5.3}$$

Furthermore, $J_{\boldsymbol{\mathcal{H}}}(\vec{\boldsymbol{w}}) = J_{\boldsymbol{\mathcal{L}}}(\vec{\boldsymbol{w}}) - J_{\boldsymbol{\mathcal{S}}}(\vec{\boldsymbol{w}})$, which yields

$$J_{\boldsymbol{\mathcal{F}}}(\vec{\boldsymbol{w}}) = \frac{1}{\Delta t}\tilde{\boldsymbol{I}} + J_{\boldsymbol{\mathcal{L}}}(\vec{\boldsymbol{w}}) - J_{\boldsymbol{\mathcal{S}}}(\vec{\boldsymbol{w}})\,, \tag{5.4}$$

to be used in the Newton iterations. In (5.4), $J_{\boldsymbol{\mathcal{L}}}(\vec{\boldsymbol{w}})$ and $J_{\boldsymbol{\mathcal{S}}}(\vec{\boldsymbol{w}})$ are given in Proposition 4.1 and Proposition 4.2, respectively.

6. Numerical Experiments.

A simple finite-difference approximation of the Jacobian is given by [8]

$$J_{i,j} \approx \hat{J}_{i,j} = \frac{\boldsymbol{\mathcal{F}}_i(\vec{\boldsymbol{w}} + \delta_j \boldsymbol{e}_j) - \boldsymbol{\mathcal{F}}_i(\vec{\boldsymbol{w}})}{\delta_j}\,. \tag{6.1}$$

The approximation in (6.1) was used during the implementation of the analytical expression of $J_{\boldsymbol{\mathcal{F}}}$ since we expected $\|J - \hat{J}\|_\infty$ to be small. This allowed us to write unit tests ensuring that the Jacobian has been correctly implemented, by comparing it to the approximation. In (6.1), a small δ leads to a good approximation. However, note that if δ is chosen too small, the approximation will be contaminated by floating-point roundoff errors, which limits the practically achievable accuracy of J [8].

Computing difference approximations of the Jacobian also allowed us to compare the efficiency of Newton's method using approximate versus analytical Jacobians. Note that computing the approximation (6.1) requires n evaluations of $\boldsymbol{\mathcal{F}}$, resulting in $O(n^2)$ complexity, compared to the $O(n)$ complexity of evaluating the exact Jacobian. Table 1 shows the execution times for evaluating the analytical Jacobian versus computing the approximation (6.1) at increasing resolutions.

Resolution	Exact	FD approximation
5×5	0.005s	0.858s
10×10	0.006s	13.85s
15×15	0.006s	76.49s
20×20	0.007s	261.6s

Table 1

Execution times for computing the exact Jacobian of $\boldsymbol{\mathcal{F}}$ versus the finite difference approximation (6.1). Even at low resolutions, using difference approximations of the Jacobian is clearly unrealistic.

As expected, due to the large number of evaluations of $\boldsymbol{\mathcal{F}}$ needed to compute the approximation, such a strategy quickly becomes infeasible.

It is readily seen that the number of floating point operations needed to evaluate the discrete spatial operator $\boldsymbol{\mathcal{L}}$ grows linearly with the degrees of freedom n. Consider for example the term $\boldsymbol{A}(I_3 \otimes \boldsymbol{D_x})\,\vec{\boldsymbol{w}}$. The first product, $(I_3 \otimes \boldsymbol{D_x})\,\vec{\boldsymbol{w}}$, are finite difference approximations at each point in the grid, resulting in Cn operations, where C depends on the width of the difference stencil. The matrix $\boldsymbol{A}$ is a 3-by-3 block matrix with diagonal blocks, and so results in another $O(n)$ number of operations. Analogously, the remaning terms in $\boldsymbol{\mathcal{L}}$ each contribute $O(n)$ operations. The arithmetic complexity of evaluating a penalty term $\boldsymbol{\mathcal{S}}$ is $O(\sqrt{n})$ (assuming equal resolution in the horizontal and vertical directions), since $\boldsymbol{\mathcal{S}}$ acts only on the grid boundary. Hence, the arithmetic complexity of evaluating $\boldsymbol{\mathcal{F}}$ is $O(n)$.

Let us study the arithmetic complexity of evaluating the Jacobian $J_{\boldsymbol{\mathcal{F}}}$ of $\boldsymbol{\mathcal{F}}$. Inspecting the form of the Jacobian $J_{\boldsymbol{\mathcal{L}}}$ in Proposition 4.1 we see a number of terms that need to be evaluated. The partial derivatives $\underline{\boldsymbol{D_x u}}$, $\underline{\boldsymbol{D_y v}}$, etc, have already been computed as part of the evaluation of $\boldsymbol{\mathcal{L}}$, and hence can be disregarded. Similarly, terms that do not depend on the solution, such as $\boldsymbol{D_x}$, $\boldsymbol{D_x^2}$, etc, can be disregarded since they remain constant throughout the simulation. Finally we have terms of the type $\boldsymbol{U D_x}$, $\boldsymbol{D_y V}$, etc. These are all products of a diagonal matrix and a banded difference stencil matrix, and each contribute with $O(n)$ operations. Summing the terms uses $O(n)$ operations. Therefore, the arithmetic complexity of evaluating $J_{\boldsymbol{\mathcal{L}}}$ is $O(n)$. In fact, the number of operations needed to evaluate products like $\boldsymbol{U D_x}$ or $\boldsymbol{D_y V}$ do not exceed the number of operations needed to compute the discrete partial derivatives involved in $\boldsymbol{\mathcal{L}}$. Hence, the cost ratio of evaluating $J_{\boldsymbol{\mathcal{L}}}$ and evaluating $\boldsymbol{\mathcal{L}}$ is less than 1 (i.e. the additional cost of evaluating $J_{\boldsymbol{\mathcal{L}}}$ is small). As before, the arithmetic complexity of evaluating the Jacobian with respect to a boundary penalty $\boldsymbol{\mathcal{S}}$ is $O(\sqrt{n})$ since it acts only on the boundary of the grid. Thus, the total arithmetic complexity of evaluating $J_{\boldsymbol{\mathcal{F}}}$ is less than the cost of evaluating $\boldsymbol{\mathcal{F}}$.

6.1. The order of accuracy. The method of manufactured solution [16] is used to verify the implementation. In all computations in this subsection, the initial guess is the solution from the previous time step and the tolerance *tol* in Algorithm 5.1 is set to 10^{-12}. For the SBP-operators SBP21 and SBP42, the expected orders of accuracy for the system (3.2) are 2 and 3, respectively [17, 19]. The manufactured solution we have used is

$$\begin{aligned} u &= 1 + 0.1\sin(3\pi x - 0.01t)\sin(3\pi y - 0.01t) \\ v &= \sin(3\pi x - 0.01t)\sin(3\pi y - 0.01t) \\ p &= \cos(3\pi x - 0.01t)\cos(3\pi y - 0.01t)\,. \end{aligned} \tag{6.2}$$

Inserting (6.2) into (2.1) leads to a non-zero right-hand side $\vec{k}(t,x,y)$, which is evaluated on the grid and added to the right-hand side of (3.2) by the vector $\vec{\boldsymbol{k}}(t)$. Since $\vec{\boldsymbol{k}}$ is independent of $\vec{\boldsymbol{w}}$, it does not affect the Jacobian. The initial and boundary data are also taken from (6.2). The step size is chosen to be $\Delta t = 10^{-5}$ and the computations are terminated at $t = 1$. Next, we compute the pointwise error vector $\vec{\boldsymbol{e}}$ and its L_2-norm $\|\vec{\boldsymbol{e}}\|_{I_3\otimes \boldsymbol{P}}$. The spatial convergence rate for the SBP operators is given by $r = \log(\|\boldsymbol{e}\|_i/\|\boldsymbol{e}\|_j)/\log((j-1)/(i-1))$, where i and j refer to the number of grid points in both spatial dimensions. The order of accuracy in space are presented in Table 2 and agree well with theory.

Next, we consider the steady-state problem of (2.1) and (3.2), which means that the goal is to find $\vec{\boldsymbol{w}}^*$ such that

$$\boldsymbol{\mathcal{L}}(\vec{\boldsymbol{w}}^*) = \boldsymbol{\mathcal{S}}(\vec{\boldsymbol{w}}^*)\,. \tag{6.3}$$

As before, we want to find an approximation to the vector $\vec{\boldsymbol{w}}^*$ which satisfies

$$\boldsymbol{\mathcal{F}}(\vec{\boldsymbol{w}}^*) = \boldsymbol{\mathcal{L}}(\vec{\boldsymbol{w}}^*) - \boldsymbol{\mathcal{S}}(\vec{\boldsymbol{w}}^*) = 0\,. \tag{6.4}$$

The Jacobian of $\boldsymbol{\mathcal{F}}$ is $J_{\boldsymbol{\mathcal{F}}}(\vec{\boldsymbol{w}}) = J_{\boldsymbol{\mathcal{L}}}(\vec{\boldsymbol{w}}) - J_{\boldsymbol{\mathcal{S}}}(\vec{\boldsymbol{w}})$. When the iterate $\vec{\boldsymbol{w}}^k$ is far away from $\vec{\boldsymbol{w}}^*$, Newton's method may not converge and other techniques must initially be applied. We

operator	SBP21		SBP42	
N	$\|\boldsymbol{e}\|$	r	$\|\boldsymbol{e}\|$	r
21	4.13e-02	–	1.90e-02	–
41	9.73e-03	2.16	2.19e-03	3.23
61	4.17e-03	2.13	6.34e-04	3.12
81	2.28e-03	2.12	2.70e-04	3.01
Theoretical		2		3

Table 2

Error and convergence rate.

choose the SOR method [15] until $\|F(\vec{\boldsymbol{w}}^k)\|_\infty$ is sufficiently small. For SOR, the next iterate is given by $\vec{\boldsymbol{w}}^{k+1} = \vec{\boldsymbol{w}}^k(1-\alpha)+(\vec{\boldsymbol{w}}^k-\boldsymbol{h}^k)\alpha$, where $\boldsymbol{h}^k$ is the Newton step from Algorithm 5.1 and $\alpha \in (0,1]$.

To verify our procedure, we choose the steady manufactured solution to be [9]

$$\begin{aligned} u &= 1-e^{\lambda x}\cos(2\pi y) \quad & v &= \frac{1}{2\pi}\lambda e^{\lambda x}\sin(2\pi y) \\ p &= \frac{1}{2}\left(1-e^{2\lambda x}\right) \quad & \lambda &= \frac{1}{2\epsilon}-\sqrt{\frac{1}{4\epsilon^2}+4\pi^2} \end{aligned} \tag{6.5}$$

and the computational domain is changed to $\Omega = [-0.5, 1] \times [-1, 1]$ for $\epsilon = 1/20$. Inserting (6.5) into the time-independent version of (2.1) leads to $\vec{k}(t, x, y) = 0$. The initial guess is $\vec{\boldsymbol{w}}^0 = (1, 1, \ldots, 1)^\top$ and the tolerance *tol* in Algorithm 5.1 is again set to 10^{-12}. Table 3 shows the error and convergence rates, which again agrees well with theory. In Figure 2, the streamlines and the velocity field is illustrated for the converged solution on the grid containing 100×100 points. They agree well with previous results [9].

6.2. The convergence rate of the Newton iteration. Next, we will test the main development in this paper. For $\vec{\boldsymbol{w}}^k$ sufficiently close to $\vec{\boldsymbol{w}}^*$, Newton's method converges quadratically in any norm [15], which means that $e_{k+1} = Ce_k^2$, where C varies marginally between iterations and $e_k = \|\vec{\boldsymbol{w}}^k - \vec{\boldsymbol{w}}^*\|$. To verify that, we consider a grid of size 100×100 with the SBP42 operator. The exact solution $\vec{\boldsymbol{w}}^*$ is approximated by the last iterate. By the assumption that C is constant, the relation

$$\frac{e_{k+1}}{e_k} \approx \left(\frac{e_k}{e_{k-1}}\right)^p$$

is obtained for a general convergence rate p, which yields

$$p \approx \frac{\log(e_{k+1}/e_k)}{\log(e_k/e_{k-1})}.$$

The error $e_k = \|\vec{\boldsymbol{w}}^k - \vec{\boldsymbol{w}}^*\|_\infty$ is presented in Table 4 together with the estimations of p. The convergence rate agrees well with the expected theoretical one, which verifies that the Jacobian of $\boldsymbol{\mathcal{F}}$ is correct.

operator	SBP21		SBP42	
N	$\|\boldsymbol{e}\|$	r	$\|\boldsymbol{e}\|$	r
21	2.04e-01	–	4.95e-02	–
41	4.56e-02	2.16	6.86e-03	2.85
61	2.04e-02	1.98	2.20e-03	2.80
81	1.16e-02	1.97	9.76e-04	2.83
101	7.46e-03	1.97	5.16e-04	2.85
Theoretical		2		3

Table 3

Error and (accuracy) convergence rate of (6.5).

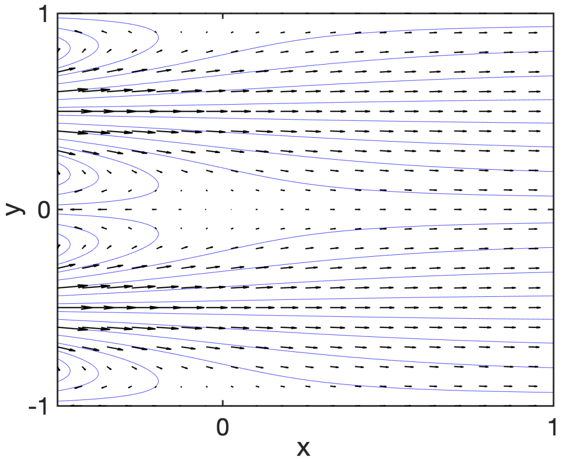

Figure 2. *Streamlines and the velocity field of* (6.5).

Table 4

Errors and the estimated (iterative) convergence rates of (6.5).

k	$\|e_k\|_\infty$	p
1	3.56e+00	–
2	1.85e+01	–
3	1.89e+00	-1.38
4	1.21e+00	0.20
5	5.53e-01	1.74
6	1.10e-01	2.07
7	3.21e-03	2.19
8	3.31e-06	1.94
9	4.14e-12	1.98
Theoretical		2

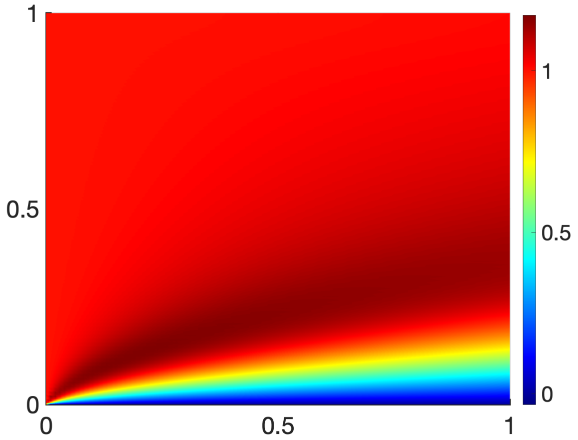

Figure 3. *Flow over a solid surface.*

Table 5
Errors and the estimated (iterative) convergence rates for the flow over a solid surface.

k	$\|e_k\|_\infty$	p
1	1.68e+00	–
2	6.17e-01	–
3	1.15e-01	1.67
4	4.37e-03	1.95
5	1.02e-05	1.85
6	5.51e-11	2.00
Theoretical		2

Next, we move on to a more realistic case where the boundary data is set to $g_1 = 1$, $g_2 = g_3 = g_4 = g_5 = g_6 = 0$ and $\epsilon = 0.01$, which will lead to a boundary layer. The computations are performed on $\Omega = [0,1]^2$ with 200×200 grid points with the SBP42 operator. Figure 3 illustrates $\boldsymbol{u}$ for the converged solution and the iterative convergence order, p, is presented in Table 5. The estimated iterative convergence order agrees well with what is theoretically expected.

In the last experiment, we consider a curved grid [1] for the incompressible Euler equation (i.e. $\epsilon = 0$). Both the south and north sides are solid surfaces, where the normal velocity is zero. The west side is an inflow boundary where $u = 1$ and $v = 0$ are specified and at the east side, $p = 0$ is imposed. We change the domain to $\Omega = [-1.5, 1.5] \times [0, 0.8]$ and include a smooth bump at the south boundary given by $y(x) = 0.0625e^{-25x^2}$ [4] . In Figure 4, the converged solution is illustrated and the estimated iterative convergence rate p is presented in Table 6 for the initial guess $(\boldsymbol{u}^0; \boldsymbol{v}^0; \boldsymbol{p}^0) = (1, \ldots, 1; 0, \ldots, 0; 1, \ldots 1)$. Again, the results agree well with the theoretical value.

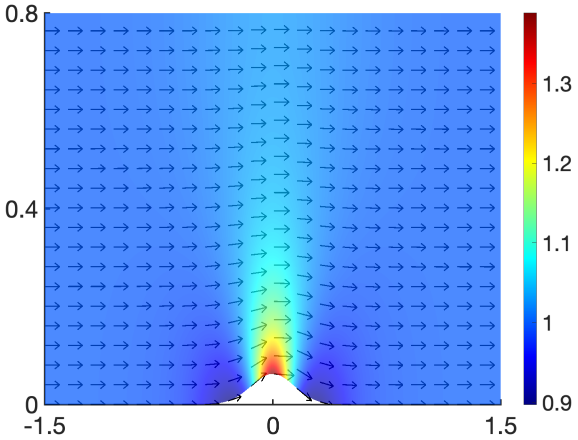

Figure 4. *Flow over a smooth bump. The plot illustrates the velocity field (arrows) and* $\boldsymbol{u}$ *(color figure) at the converged solution.*

Table 6
Errors and the estimated (iterative) convergence rates for the bump.

k	$\|e_k\|_\infty$	p
1	3.56e+00	–
2	4.91e-01	–
3	1.74e-02	1.69
4	3.33e-05	1.87
5	1.55e-10	1.96
Theoretical		2

7. Conclusions.
We derived an explicit expression for the Jacobian of a finite-difference discretization of the incompressible Navier-Stokes equations. Both the Jacobian of the system of equations and the Jacobian of the related boundary condition was computed exactly. By using the block-structure of the discretization, we showed that the Jacobian had a block structure as well, which lead to a compact and clean expression. We also showed that large parts of the Jacobian were computed by evaluating the discretization. We showed that the Jacobian could be used both in steady-state and time-dependent simulations. The numerical discretization was verified by manufactured solutions and the spatial convergence rates agreed well with the theoretical expectations. Furthermore, the computed estimates of the iterative convergence rates for Newton's method was two, and verified that the Jacobian was correctly computed. The methodology used in this paper is general and can be used in a straightforward manner for any numerical discretization of initial boundary value problems that can be written in matrix-vector form.

REFERENCES

[1] O. Ålund and J. Nordström, *Encapsulated high order difference operators on curvilinear non-conforming grids*, Journal of Computational Physics, (2019), pp. 209–224.

[2] P. N. Brown and Y. Saad, *Hybrid krylov methods for nonlinear systems of equations*, SIAM Journal on Scientific and Statistical Computing, 11 (1990), pp. 450–481.

[3] M. H. Carpenter, D. Gottlieb, and S. Abarbanel, *Time-stable boundary conditions for finite-difference schemes solving hyperbolic systems: methodology and application to high-order compact schemes*, Journal of Computational Physics, 111 (1994), pp. 220–236.

[4] Cenaero, *VI2 Smooth Gaussian bump*, 2020 (accessed December 9, 2020). https://how5.cenaero.be/content/vi2-smooth-gaussian-bump.

[5] J. Chan and C. Taylor, *Explicit Jacobian matrix formulas for entropy stable summation-by-parts schemes*, arXiv preprint arXiv:2006.07504, (2020).

[6] D. C. D. R. Fernández, J. E. Hicken, and D. W. Zingg, *Review of summation-by-parts operators with simultaneous approximation terms for the numerical solution of partial differential equations*, Computers & Fluids, 95 (2014), pp. 171–196.

[7] A. Jameson, *Time dependent calculations using multigrid, with applications to unsteady flows past airfoils and wings*, in 10th Computational Fluid Dynamics Conference, 1991, p. 1596.

[8] D. A. Knoll and D. E. Keyes, *Jacobian-free Newton–Krylov methods: a survey of approaches and applications*, Journal of Computational Physics, 193 (2004), pp. 357–397.

[9] L. Kovasznay, *Laminar flow behind a two-dimensional grid*, Mathematical Proceedings of the Cambridge Philosophical Society, 44 (1948), pp. 58–62.

[10] J. Nocedal and S. Wright, *Numerical optimization*, Springer Science & Business Media, 2006.

[11] J. Nordström, *A roadmap to well posed and stable problems in computational physics*, Journal of Scientific Computing, 71 (2017), pp. 365–385.

[12] J. Nordström and C. La Cognata, *Energy stable boundary conditions for the nonlinear incompressible Navier–Stokes equations*, Mathematics of Computation, 88 (2019), pp. 665–690.

[13] J. Nordström and A. A. Ruggiu, *Dual time-stepping using second derivatives*, Journal of Scientific Computing, 81 (2019), pp. 1050–1071.

[14] T. C. Papanastasiou, N. Malamataris, and K. Ellwood, *A new outflow boundary condition*, International journal for numerical methods in fluids, 14 (1992), pp. 587–608.

[15] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical mathematics*, vol. 37, Springer Science & Business Media, 2010.

[16] P. J. Roache, *Code verification by the method of manufactured solutions*, J. Fluids Eng., 124 (2002), pp. 4–10.

[17] M. Svärd and J. Nordström, *On the order of accuracy for difference approximations of initial-boundary value problems*, Journal of Computational Physics, 218 (2006), pp. 333–352.

[18] M. Svärd and J. Nordström, *Review of summation-by-parts schemes for initial–boundary-value problems*, Journal of Computational Physics, 268 (2014), pp. 17–38.

[19] M. Svärd and J. Nordström, *On the convergence rates of energy-stable finite-difference schemes*, Journal of Computational Physics, 397 (2019), p. 108819.