Journal of Computational Physics 371 (2018) 651–662

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Bandwidth-based mesh adaptation in multiple dimensions

Elliott S. Wise *, Ben T. Cox, Bradley E. Treeby

Department of Medical Physics and Biomedical Engineering, University College London, Gower Street, London, WC1E 6BT, United Kingdom

ARTICLE INFO

Article history:
Received 13 September 2017
Received in revised form 16 February 2018
Accepted 1 June 2018
Available online xxxx

Keywords:
Adaptive moving mesh method
Pseudospectral method
Bandwidth

ABSTRACT

Spectral methods are becoming increasingly prevalent in solving time-varying partial differential equations due to their fast convergence properties. However, they typically use regular computational meshes that do not account for spatially varying resolution requirements. This can significantly increase the overall grid density when resolution requirements vary sharply over the modelled domain. Moving mesh methods offer a remedy for this, by allowing the position of mesh nodes to adapt to the simulated model solution. In this paper, a mesh specification is presented that is based on a local measure of the spatial bandwidth of the model solution. This addresses the rate of decay of the model solution's frequency components by producing high-sampling rates when this decay is slow. The spatial bandwidth is computed using a combination of the original solution and its Riesz transformed counterparts. It is then integrated into a Fourier spectral moving mesh method, using the parabolic Monge–Ampère equation for mesh control. This method is used to solve a multidimensional version of the viscous Burgers equation, and a heterogeneous advection equation. The performance of bandwidth-based mesh adaptation is compared with arclength- and curvature-based adaptation, and against a static mesh. These numerical experiments show that the bandwidth-based approach produces superior convergence rates, and hence requires fewer mesh nodes for a given level of solution accuracy.

1. Introduction

Spectral methods are increasingly being used for the numerical solution of differential equations [1]. To do this, they use globally-defined basis functions and, when compared with low-order, local methods, often produce high accuracy with relatively coarse spatial discretisations [2]. The sampling rate for spectral methods is related to the rate of decay of the solution's frequency components: functions that vary rapidly require denser sampling than those that vary more smoothly [1]. To meet these sampling requirements, spectral meshes typically used fixed, standardised meshes whose spacing is chosen to ensure that some maximum frequency component is supported. For example, the Fourier collocation method is usually applied using equispaced meshes whose spacing ensures at least two points per minimum wavelength of interest—a choice arising from the Shannon–Nyquist sampling theorem. Any frequency components beyond this will not be supported by the mesh. For many problems of interest, spatial resolution requirements are not uniform throughout the simulated domain. For example, in [3–5] a Fourier collocation method was used to simulate high-intensity focussed ultrasound fields. These contain tightly localised shock fronts that require dense computational meshes, but most of the field only has power at low

* Corresponding author.
E-mail addresses: elliott.wise.14@ucl.ac.uk (E.S. Wise), b.cox@ucl.ac.uk (B.T. Cox), b.treeby@ucl.ac.uk (B.E. Treeby).

https://doi.org/10.1016/j.jcp.2018.06.009

frequencies. As the meshes used in those works were uniform, the strict shock front sampling requirement was applied everywhere, and large-scale, high-performance computing resources were needed to store and process field variables at each time-step.

To accommodate varying resolution requirements, spectral methods can be implemented within a moving mesh method framework [6]. These dynamically adapt mesh node positions throughout a simulation in a solution-dependent manner. This allows both temporally- and spatially-varying resolution requirements to be met. Monitor functions are used to link the model solution to the mesh, and hence guide mesh adaptation. The choice of monitor function is critical to the performance of moving mesh methods. There have been a number of past multidimensional spectral moving mesh methods, including Fourier, Galerkin, and Chebyshev types [7–11]. These all used arclength-like monitor functions that cluster mesh nodes where the gradient of the model solution is large. In all cases, these methods were applied to problems whose solutions were characterised by steep fronts, and so it is unsurprising that gradient-based mesh adaptation was effective. However, many problems include features for which gradient-based adaptation is not an obvious choice and, additionally, gradient-based mesh adaptation has not been theoretically justified in the context of spectral methods.

A mesh adaptation approach that is tailored to spectral collocation methods was presented in [12] and applied with success to one-dimensional problems. It used a high-pass filter to find regions with large high-frequency solution components, and increased the mesh node density there accordingly. A weakness of this method is that the high-pass filtering step requires parameter choices that are problem- and interpolant-specific. Following this, the (spatially) local bandwidth was presented as a parameterless and robust approach to frequency-based mesh adaptation, and applied to a variety of one-dimensional acoustics problems [13]. In that work, the bandwidth measured the local rate of decay of the solution's frequency components, and the sampling density was chosen to be proportional to its reciprocal. When compared with arclength-based mesh adaptation, the bandwidth-based approach considerably improved the convergence rates of Chebyshev, Fourier, and even finite-difference methods. However, the algorithm presented in that work is limited to one-dimensional problems, as it uses the analytic signal to decouple the spatial phase and amplitude of the model variable.

This paper introduces a multidimensional bandwidth-based mesh adaptation method. It works by first decoupling the spatial phase and amplitude of the model solution using the *monogenic signal* [14,15]. From this, the local bandwidth of the solution is computed and used as a specification for mesh adaptation. This specification is integrated into a Fourier spectral moving mesh method, and assessed against arclength- and curvature-based mesh adaptation. To do so, a multidimensional viscous Burgers equation and a heterogeneous advection equation are used to simulate the formation and propagation of a shock front and a sharp peak.

2. Bandwidth-based mesh adaptation

2.1. Multidimensional local bandwidth

To optimally sample a function using a nonuniform grid, the (spatially-varying) spatial frequency content of that function can be used. Specifically, for a band-limited function the local sampling rate should reflect the maximum spatial frequency present at that point [16]. For functions that are not band-limited, the local sampling rate should similarly reflect the rate at which local spatial frequencies decay. In this regard, the local bandwidth has been shown to be an effective measure of this decay for one-dimensional mesh adaptation [13].

To perform local, multidimensional spatial frequency analysis, it is useful to consider the monogenic signal [14]. This is a multidimensional generalisation of the analytic signal, which contains the original signal as one component and a quadrature signal as the other. By augmenting the original signal, the monogenic signal decouples the local amplitude and phase, making local frequency analysis more straightforward. Given a d-dimensional scalar field u, the monogenic signal can be written as a vector field $\mathbf{v}$ with $d+1$ components consisting of the original scalar field and its Riesz-transformed counterparts

$$\mathbf{v} = \begin{pmatrix} u & \mathcal{R}_1 u & \cdots & \mathcal{R}_d u \end{pmatrix}^T .$$

Here, the Riesz transform $\mathcal{R}_j$ is defined in Fourier-space by

$$\mathcal{R}_j u = \mathcal{F}^{-1}\left\{ -\frac{ik_j}{\|\mathbf{k}\|} \mathcal{F}\{u\} \right\}, \tag{1}$$

where i is the imaginary unit, $\mathcal{F}$ is the Fourier transform, $\mathbf{k}$ is a vector-field of wavenumbers corresponding to $\mathbf{x}$, and j indicates the coordinate axis. Now let $\mathbf{V}(\mathbf{k})$ be the Fourier transform of $\mathbf{v}(\mathbf{x})$, and assume without loss of generality that

$$\int_{\mathbb{R}^d} \|\mathbf{V}\| \, d\mathbf{k} = 1.$$

Then, the square of the global spatial bandwidth B_j aligned with coordinate axis j is defined as the expected value of k_j^2. That is,

$$B_j^2 = \langle \mathbf{V} | k_j^2 | \mathbf{V} \rangle = \int_{\mathbb{R}^d} k_j^2 \, \|\mathbf{V}\| \, d\mathbf{k}.$$

To localise the spatial bandwidth, an operator is defined as

$$K_j = \begin{cases} \frac{1}{i} \frac{d}{dx_j} & \text{in the position representation} \\ k_j & \text{in the wavenumber representation.} \end{cases}$$

Then, the expected value can be written in either the position or wavenumber domains since

$$B_j^2 = \langle \mathbf{V} | K_j^2 | \mathbf{V} \rangle = \langle \mathbf{v} | K_j^2 | \mathbf{v} \rangle.$$

This can in turn be rearranged to give

$$\begin{aligned} B_j^2 &= \langle \mathbf{v} | K_j^2 | \mathbf{v} \rangle \\ &= \langle K_j \mathbf{v}, K_j \mathbf{v} \rangle \\ &= \int_{\mathbb{R}^d} \left\| \frac{\partial \mathbf{v}}{\partial x_j} \right\|^2 d\mathbf{x} \\ &= \int_{\mathbb{R}^d} \left\| \frac{\partial \mathbf{v} / \partial x_j}{\mathbf{v}} \right\|^2 \|\mathbf{v}\|^2 \, d\mathbf{x} \end{aligned}$$

The local spatial bandwidth b_j is then defined as the square root of the left term in this integrand, that is

$$b_j = \left\| \frac{\partial \mathbf{v} / \partial x_j}{\mathbf{v}} \right\|, \qquad \text{so that } B_j^2 = \int_{\mathbb{R}^d} b_j^2 \, \|\mathbf{v}\|^2 \, d\mathbf{x}.$$

For use as a mesh specification, it is useful to include an amplitude weighting so that resolution isn't spent on regions in which the model solution has little power. Hence, the local, amplitude-weighted bandwidths are finally defined as

$$\rho_j = \left\| \frac{\partial \mathbf{v}}{\partial x_j} \right\|.$$

As an example, ρ_1 is computed in two dimensions as

$$\rho_1 = \left[\left(\frac{\partial u}{\partial x_1} \right)^2 + \left(\frac{\partial (\mathcal{R}_1 u)}{\partial x_1} \right)^2 + \left(\frac{\partial (\mathcal{R}_2 u)}{\partial x_1} \right)^2 \right]^{\frac{1}{2}}.$$

These ρ_j are finally combined to form the bandwidth mesh density vector

$$\boldsymbol{\rho} = \begin{pmatrix} \rho_1 & \cdots & \rho_d \end{pmatrix}^T.$$

2.2. Mesh movement

In the context of moving mesh methods, a mesh is specified using a monitor function. In general, this is a matrix-valued field whose eigendecomposition specifies mesh node densities in orthogonal directions [17]. Above, the bandwidth mesh density vector $\boldsymbol{\rho}$ specifies desired mesh node densities along the coordinate axes (which are by definition orthogonal), and so a diagonal matrix of its elements could serve as a monitor function. In this work, the parabolic Monge–Ampère (PMA) equation is used to generate a mesh from a monitor function [18]. This considers the physical coordinate $\mathbf{x}$ to be related to a temporally-fixed computational coordinate $\mathbf{s}$ using a time-varying transformation $\mathbf{x} = \mathbf{x}(t, \mathbf{s})$. The PMA equation[1] is given by

$$\dot{\mathbf{x}} = \tau^{-1} \frac{\partial}{\partial \mathbf{s}} \left(\frac{|\mathbf{M}| |\mathbf{J}^{-1}|}{\theta} \right)^{1/d}, \qquad \theta = \int_{\Omega} |\mathbf{M}| d\mathbf{x}, \qquad \tau = 10^{-2}. \tag{2}$$

[1] The PMA equation is usually expressed in terms of a potential function whose gradient gives the mesh transformation. In the present work, it is expressed in terms of the mesh coordinate directly. Additionally, the Laplacian smoothing term that is usually included in the PMA equation has been replaced with a frequency filter applied to $\dot{\mathbf{x}}$. This is described and justified in §4.4.

Here $\mathbf{M}$ is a monitor function, $\mathbf{J}$ is the mesh Jacobian matrix (defined below in (6)), $|\cdot|$ indicates a matrix determinant, and Ω is the physical domain. The overset dot on $\mathbf{x}$ indicates a time-derivative in the computational domain. The PMA equation is a relaxation equation that finds a mesh that meets the equidistribution condition exactly, while minimising a functional $I[\mathbf{x}]$ measuring the overall amount of adaptation

$$I[\mathbf{x}] = \int_{\Omega_C} |\mathbf{x}(\mathbf{s}) - \mathbf{s}|^2 d\mathbf{s}.$$

Here Ω_C is the computational domain. The parameter $\tau = 10^{-2}$ controls the rate at which the PMA equation converges, and has been chosen here to reflect the time-scale of the model problems in §5.1. The PMA equation has previously been successfully applied to mesh generation for meteorological problems [19–21].

It is notable that the PMA equation ignores directional information in the monitor function, since it takes the monitor function's determinant. This might lead one to think that it would produce isotropic meshes only. However, it has been shown that this is not the case, and that the mesh anisotropy the PMA equation produces aligns with that of features in the model solution to which it's applied [22]. The lack of directional information also means that a matrix-valued monitor function is unnecessary as only its determinant is used. Instead, $|\mathbf{M}|$ can be replaced with a scalar-valued mesh density function ρ. This could simply be $\rho = |\mathbf{M}|$, where $\mathbf{M}$ is a diagonal matrix with elements given by $\boldsymbol{\rho}$, but this choice can result in no mesh nodes being placed at a location if any component of $\boldsymbol{\rho}$ is zero. An alternative is to use a measure of the bandwidth mesh density vector's length. This ensures that $\rho = 0$ only when all ρ_j are zero and u is constant locally (an expected behaviour since a constant function requires one sample over an infinite domain). This work considers the root-mean-square

$$\rho = d^{-\frac{1}{2}} \|\boldsymbol{\rho}\|. \tag{3}$$

As an example, in two dimensions this is given by

$$\rho = \frac{1}{\sqrt{2}} \left[\left(\frac{\partial u}{\partial x_1}\right)^2 + \left(\frac{\partial u}{\partial x_2}\right)^2 + \left(\frac{\partial(\mathcal{R}_1 u)}{\partial x_1}\right)^2 + \left(\frac{\partial(\mathcal{R}_1 u)}{\partial x_2}\right)^2 + \left(\frac{\partial(\mathcal{R}_2 u)}{\partial x_1}\right)^2 + \left(\frac{\partial(\mathcal{R}_2 u)}{\partial x_2}\right)^2 \right]^{\frac{1}{2}}.$$

Equation (3) incorporates information from every component of $\boldsymbol{\rho}$ and scales appropriately with dimensionality. The L^2 norm was chosen to ensure ρ is smooth when each component of $\boldsymbol{\rho}$ is smooth. This makes it appropriate for use with spectral methods.

3. Model equations

3.1. A multidimensional viscous Burgers equation

To evaluate bandwidth-based mesh adaptation, two model equations are used. The first is a viscous Burgers equation [23]. This equation generates shock fronts for which dense, anisotropic meshes are beneficial. In one dimension, it takes the standard form

$$u_t = \varepsilon \frac{\partial^2 u}{\partial x^2} - u \frac{\partial u}{\partial x}, \qquad \varepsilon = 10^{-2}.$$

In two-dimensions, an extension of this equation propagates the solution parallel to a specified unit vector $\hat{\mathbf{w}}$

$$u_t = \varepsilon \Delta u - u(\nabla u \cdot \hat{\mathbf{w}}), \qquad \varepsilon = 10^{-2}.$$

In both, the viscosity parameter ($\varepsilon = 10^{-2}$ in these examples) has been chosen to generate a steep shock front. Illustrative examples are provided in §5.1, and some numerical results are presented in §5.2 and §5.3. Note that these results are not significantly affected by the choice of propagation vector $\hat{\mathbf{w}}$.

3.2. A heterogeneous advection equation

The second model is an advection equation

$$u_t = -c(\mathbf{x}) \frac{\partial u}{\partial x_1}, \qquad c(\mathbf{x}) = [1 + 0.9\cos(x_1)]^{-1}.$$

This model describes linear advection along the x_1 coordinate axis, with a propagation speed that is slower in the middle of the domain than the edges. This causes the propagating wave to anisotropically compress and expand as it propagates, making the use of an adaptive mesh beneficial.

3.3. Mesh/model coupling

To couple the mesh and model equations, the quasi-Lagrange approach is used. This solves the two equations simultaneously by expressing the model's temporal derivative in the computational domain using the following relationship [6, p. 142]:

$$u_t = \dot{u} - \nabla u \cdot \dot{\mathbf{x}}. \tag{4}$$

Here, the subscript t indicates a time-derivative in the physical domain, and an overset dot again indicates a time-derivative in the computational domain. Equation (4) can be rearranged to yield $\dot{u}$, and solved simultaneously with the mesh equation for $\dot{\mathbf{x}}$.

4. Numerical methods

4.1. Spatial calculus

Below, the physical and computational coordinates are considered to have the same domain of periodicity, and the computational coordinate is uniformly sampled. Spatial gradients can then be computed using a Fourier collocation method, with all calculations performed relative to the computational mesh. For a scalar-field u, this takes two steps. First, a gradient is taken with respect to the computational mesh, computed using the Fast Fourier transform (FFT) $\mathcal{F}$ via

$$\frac{\partial u}{\partial \mathbf{s}} = \mathcal{F}^{-1}\left\{i\mathbf{k}\mathcal{F}\{u\}\right\}. \tag{5}$$

Here, $\mathbf{k}$ is a vector-field of wavenumbers corresponding to the computational coordinate $\mathbf{s}$. Next, derivatives are converted into the physical domain using the chain rule. To do this, the mesh Jacobian matrix is defined as

$$\mathbf{J} = \frac{\partial \mathbf{s}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial s_1}{\partial x_1} & \cdots & \frac{\partial s_1}{\partial x_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial s_d}{\partial x_1} & \cdots & \frac{\partial s_d}{\partial x_d} \end{pmatrix}, \tag{6}$$

and physical gradients are then given by

$$\nabla = \frac{\partial}{\partial \mathbf{x}} = \mathbf{J}^T \frac{\partial}{\partial \mathbf{s}}.$$

More complex differential operators are similarly computed. For instance, the divergence of a vector-field $\mathbf{v}$ is given by the Frobenius product (sum of the element-wise products)

$$\nabla \cdot \mathbf{v} = \mathbf{J}^T : \frac{\partial \mathbf{v}}{\partial \mathbf{s}} = \begin{pmatrix} \frac{\partial s_1}{\partial x_1} & \cdots & \frac{\partial s_d}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial s_1}{\partial x_d} & \cdots & \frac{\partial s_d}{\partial x_d} \end{pmatrix} : \begin{pmatrix} \frac{\partial v_1}{\partial s_1} & \cdots & \frac{\partial v_1}{\partial s_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial v_d}{\partial s_1} & \cdots & \frac{\partial v_d}{\partial s_d} \end{pmatrix}.$$

To compute the mesh Jacobian matrix using a Fourier collocation method, the following expression is used:

$$\frac{\partial \mathbf{x}}{\partial \mathbf{s}} = \frac{\partial}{\partial \mathbf{s}}(\mathbf{x} - \mathbf{s}) + \mathbf{I}. \tag{7}$$

This approach transforms the smooth, monotonic mesh transformation $\mathbf{x}(\mathbf{s})$ into a periodic function $\mathbf{x} - \mathbf{s}$ for which Fourier spectral differentiation is suitable. The mesh Jacobian matrix in (6) is then computed as the inverse

$$\mathbf{J} = \left(\frac{\partial \mathbf{x}}{\partial \mathbf{s}}\right)^{-1}.$$

Finally, integral terms may be also be expressed in the computational domain using the chain rule:

$$\int_{\Omega} u d\mathbf{x} = \int_{\Omega_C} u |\mathbf{J}^{-1}| d\mathbf{s}.$$

As the computational coordinate $\mathbf{s}$ is uniformly sampled and periodic, trapezoidal quadrature can then be used.

4.2. Computing the bandwidth mesh density vector

To compute each component of the bandwidth mesh density vector, the Fourier multiplier for the Riesz transform in (1) is combined with the gradient's Fourier multiplier in (5). This yields

$$\mathcal{F}\left\{\frac{\partial \mathbf{v}}{\partial s_j}\right\} = \begin{pmatrix} ik_j & \frac{k_j k_1}{\|\mathbf{k}\|} & \cdots & \frac{k_j k_d}{\|\mathbf{k}\|} \end{pmatrix}^T \mathcal{F}\{u\}.$$

From here, each component is transformed back out of the Fourier domain and combined into a mesh density vector $\tilde{\boldsymbol{\rho}}$ in the computational domain. This is finally transformed into the physical domain

$$\boldsymbol{\rho} = \mathbf{J}^T \tilde{\boldsymbol{\rho}},$$

and the bandwidth mesh density function is computed using (3).

In one dimension, this approach yields a similar (but not identical) mesh density function to the physical-domain expression given in [13]. The difference between the two arises from the fact that the Riesz transform (or, equivalently, the Hilbert transform in one dimension) is computed in the computational domain in the present work, and in the physical domain in [13]. The reason for this difference lies in the fact that the Hilbert and Riesz transforms are singular integrals. In [13], a technique was used to subtract out the Hilbert transform's singularity, allowing its integral definition to be easily computed. The technique that was used to do so does not readily extend to the higher-order singularity present in the multi-dimensional Riesz transforms. The Fourier-domain expressions avoid the Riesz transforms' singularities, but discrete forms require equispaced samples or a nonuniform Fourier transform (the latter has not been considered in this work).

4.3. Time-stepping

To integrate the mesh and model time-derivatives (with the latter expressed in terms of the computational coordinates, as discussed in §3.3), the method of lines (MOL) was used, with adaptive time-stepping implemented by two of Matlab's ODE solvers [24]. For the convergence analysis presented in §5.2, `ode113` was chosen. This is an Adams–Bashforth–Moulton predictor–corrector method with an adaptive order between 1 and 12. It is recommended for use when very small error tolerances are required. For the stiffness analysis presented in §5.3, `ode23` was chosen. It uses the Bogacki–Shampine $(2, 3)$ Runge–Kutta pair. This solver provides a more smoothly varying time-step size than `ode113`, and hence makes it easier to compare the relative stiffness of the systems of ODEs resulting from MOL discretisations with different mesh specifications.

4.4. Spatial smoothing

Introducing mesh adaptation into a model can significantly increase the stiffness of the resulting system [25], meaning very small timesteps are required. This is because the mesh equation typically needs to converge faster than the time-scale over which the model equation's solution changes. One way in which this stiffness can be alleviated is by discretising the mesh at a lower resolution than the model (and upsampling it when needed for gradient calculations). This is known as a two-level moving mesh method [6,26]. It exploits the observation that a highly accurate model solution doesn't require an equally accurate mesh equation solution. Here, spatial smoothing is used for the same purpose. To do so, a smoothing kernel is defined in the computational spatial frequency domain as a tensor product of one-dimensional Blackman windows. These windows are constructed to decay to zero by a particular cut-off wavenumber. They are then applied to both the mesh density function ρ, and to the mesh velocity $\dot{\mathbf{x}}$. The cut-off wavenumber ensures that the mesh coordinate will always be oversampled, since any frequency components past this wavenumber will be zeroed when the window is applied. Compared with a conventional two-level method, this approach gives a similar reduction in stiffness, but avoids the computational expense of upsampling and is algorithmically simpler to implement.

4.5. Fixed boundary nodes

As a Fourier collocation method is used to solve the model and mesh equations, periodic boundary conditions are implied. However, while the mesh transformation is periodic, the mesh nodes themselves are free to translate beyond the boundaries of the simulation. This is problematic for plotting, as it is useful to visualise the solution on a fixed domain. To fix boundary nodes in space, the normal component of the boundary nodes' velocity is subtracted from every node's velocity. This modification does not affect the overall density of nodes, but does slow convergence slightly and can produce a small amount of skewness (evident in Fig. 1).

5. Numerical results for Burgers equation

5.1. Illustrative examples

Illustrative simulations for Burgers equation conducted with two initial conditions are shown in Figs. 1 and 2. Both use the bandwidth mesh density function for mesh adaptation. The first simulation is two-dimensional, and uses a smoothed

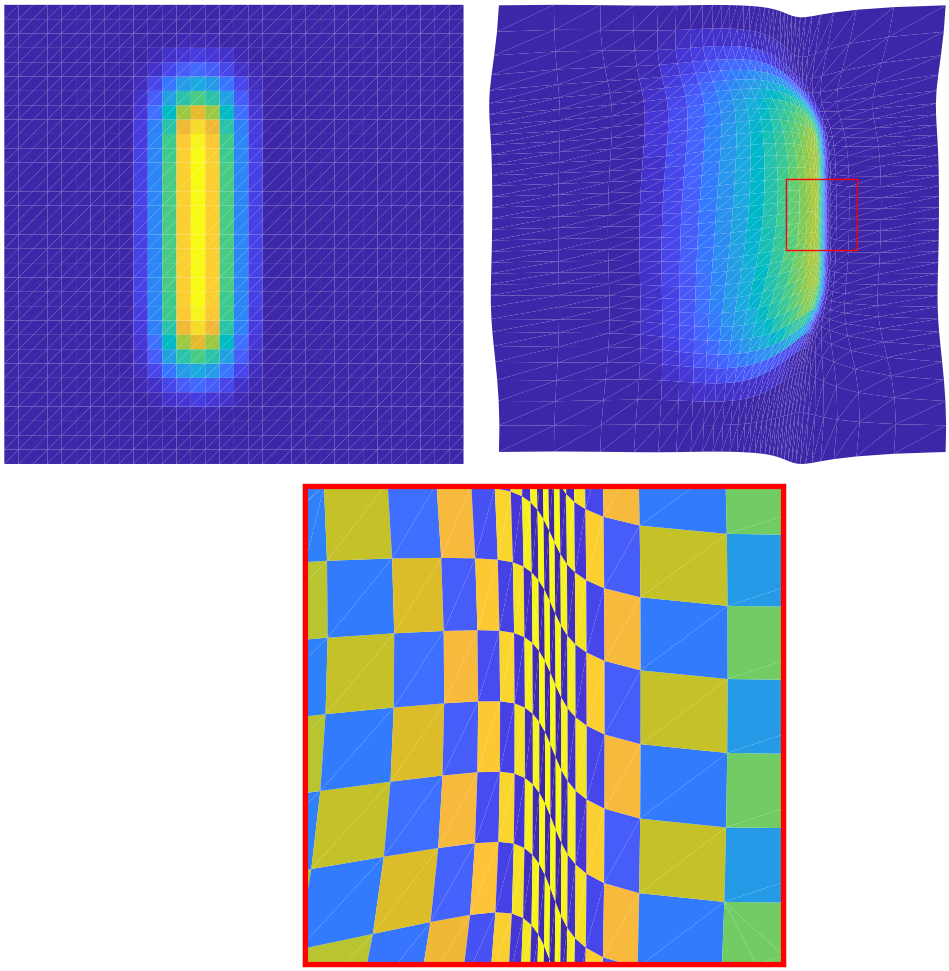

Fig. 1. Burgers equation solutions from simulations conducted using the bandwidth mesh density function (3). The model and mesh resolution were $N_u = N_{\mathbf{x}} = 32$. (Top-left) Initial condition and (top-right) final solution for a two-dimensional simulation. (Bottom) Close view of the two-dimensional mesh surrounding the shock front region indicated in the top-right subplot. Patches are centred on mesh nodes, and colour intensity indicates the determinant of the mesh Jacobian matrix (i.e. the overall mesh density at that node). Colours alternate between blue and yellow for clarity. The slight vertical distortion in the mesh arises from using fixed (but still periodic) boundary conditions in the mesh equation. (For interpretation of the colours in the figures, the reader is referred to the web version of this article.)

line as an initial condition, and a propagation vector $\hat{\mathbf{w}} = (1, 0)^T$ that is parallel to the x_1 coordinate axis. A shock front can be seen to form, and a close-up view of the mesh shows anisotropic adaptation aligned with the shock front. The second initial condition is a variation on the von Mises distribution

$$u(0, \mathbf{x}) = \exp\left[\sum_{j=1}^{d} \cos(x_i) + \prod_{j=1}^{d} \cos(x_i) - (d+1)\right], \qquad \hat{\mathbf{w}} = \left(\tfrac{1}{2} \quad \tfrac{\sqrt{3}}{2}\right),$$

where d is the number of dimensions ($d = 1, 2$). This initial condition is 2π-periodic in all dimensions, and $\hat{\mathbf{w}}$ produces off-axis propagation through the domain. The initial condition was chosen for two reasons. First, it is smoothly periodic, and so Gibbs oscillations will not appear when a Fourier pseudospectral method is applied to it. Second, it is approximately rotationally symmetric in two dimensions, so that results can be easily compared with one-dimensional simulations. As before, a single shock front forms as time progresses. The slope of this shock is maximised at approximately $t = 2.5$, at which point the simulation concludes. Again, a close-up view of the mesh shows anisotropic adaptation aligned with the shock front. This example is used in the numerical experiments presented below.

5.2. Convergence

A number of simulations based on the second example in §5.1 were conducted to examine the performance of the bandwidth mesh density function. Comparisons were made using simulations conducted without mesh adaptation, and

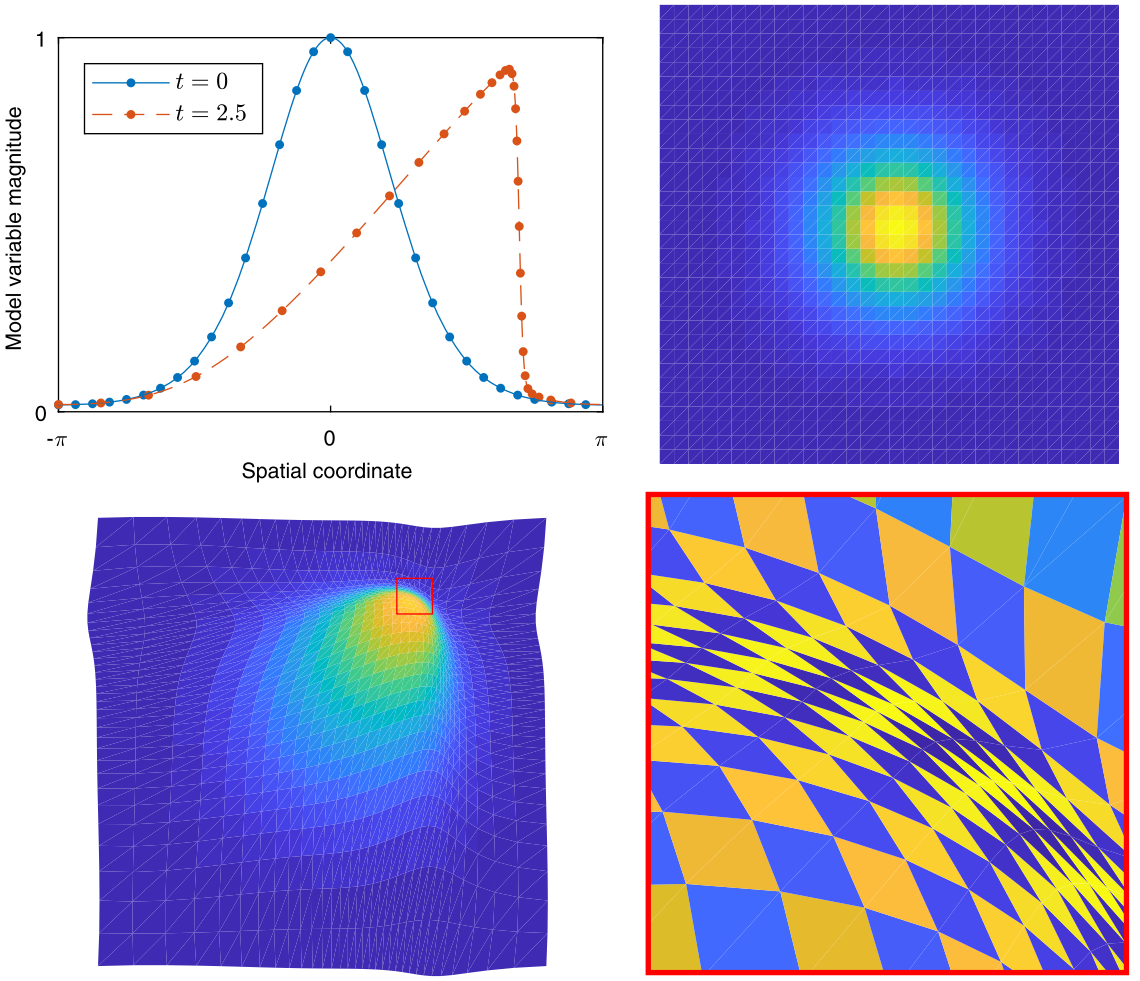

Fig. 2. Burgers equation solutions from simulations conducted using the bandwidth mesh density function (3). The model and mesh resolution were $N_u = N_{\mathbf{x}} = 32$. (Top-left) Initial condition (solid, blue) and final solution (dashed, orange) for a one-dimensional simulation. (Top-right) Initial condition and (bottom-left) final solution for a two-dimensional simulation. (Bottom-right) Close view of the two-dimensional mesh surrounding the shock front region indicated in the bottom-left subplot. Patches are centred on mesh nodes, and colour intensity indicates the determinant of the mesh Jacobian matrix (i.e. the overall mesh density at that node). Colours alternate between blue and yellow for clarity.

with meshes adapted using the arclength, and curvature mesh density functions. Here, the arclength and curvature mesh density functions are respectively defined by

$$\rho = \left(1 + \|\nabla u\|^2\right)^{\frac{d}{2}}, \qquad \rho = \left(1 + |\mathbf{H}(u)|^2\right)^{\frac{1}{4}}, \tag{8}$$

where $|\mathbf{H}(u)|$ is the determinant of the Hessian matrix of u. Note that the arclength mesh density function usually doesn't include the exponent d, but this ensures that its units and performance are consistent for any dimension. In one-dimension, an additional comparison was made with simulations conducted using the bandwidth mesh density function computed using the algorithm in [13], rather than the algorithm presented in this work. For all simulations, the model and mesh were discretised using $N_u = 32, 40, \ldots, 128$ points in each dimension, with the smoothing window chosen such that only $N_{\mathbf{x}} = 32$ mesh components were non-zero. The absolute and relative time-stepping error tolerances were $100\epsilon \approx 2.22 \times 10^{-14}$, where ϵ is double-precision machine epsilon (this is the smallest relative error tolerance Matlab's ODE solvers accept).

To measure the convergence rate resulting from each mesh specification, adjacent solutions were compared as the simulation resolution was increased. For each pair, let the low-resolution solution be labelled A, and the high-resolution solution be labelled B. First, solution A was upsampled to the resolution of solution B using zero-padding in the computational coordinate's Fourier domain. This is straightforward for the model variable. For the mesh, the quantity $\mathbf{x}_A - \mathbf{s}_A$ was upsampled, and $\mathbf{s}_B$ was added afterwards to yield $\mathbf{x}_B$. A moving mesh method was then used to interpolate the upsampled solution A onto mesh B [6, §2.6.3]. To do so, the following moving mesh problem was solved:

$$u_t = 0, \qquad \dot{\mathbf{x}} = \mathbf{x}_B - \mathbf{x}_A.$$

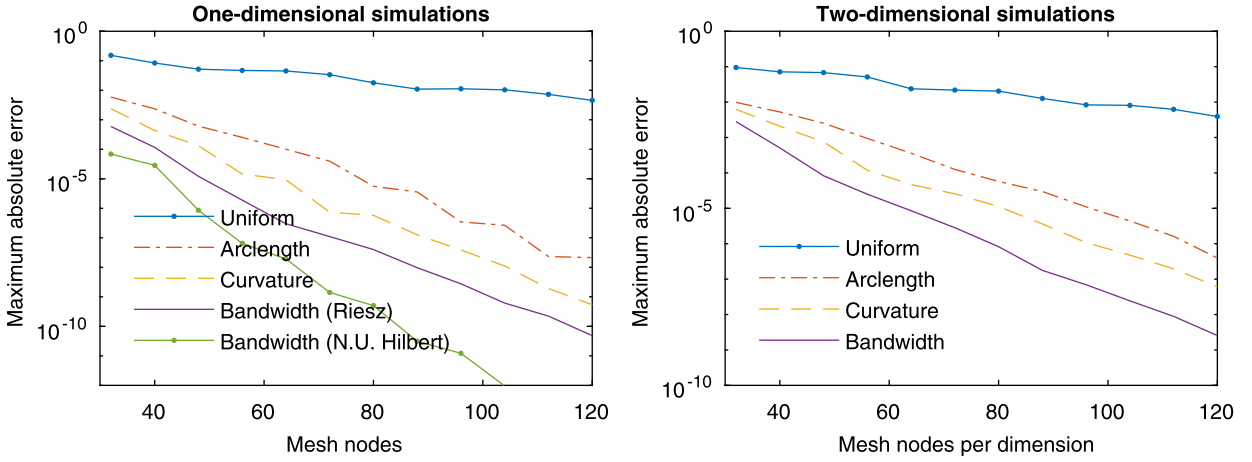

Fig. 3. Error in Burgers equation simulations conducted using various mesh specifications. In all cases, adaptive meshes outperformed the simulations conducted using a uniform mesh. The bandwidth mesh density function from [13] performs best in one-dimension. In both one- and two-dimensions, the bandwidth mesh density function from this work outperforms both the curvature and arclength mesh density functions.

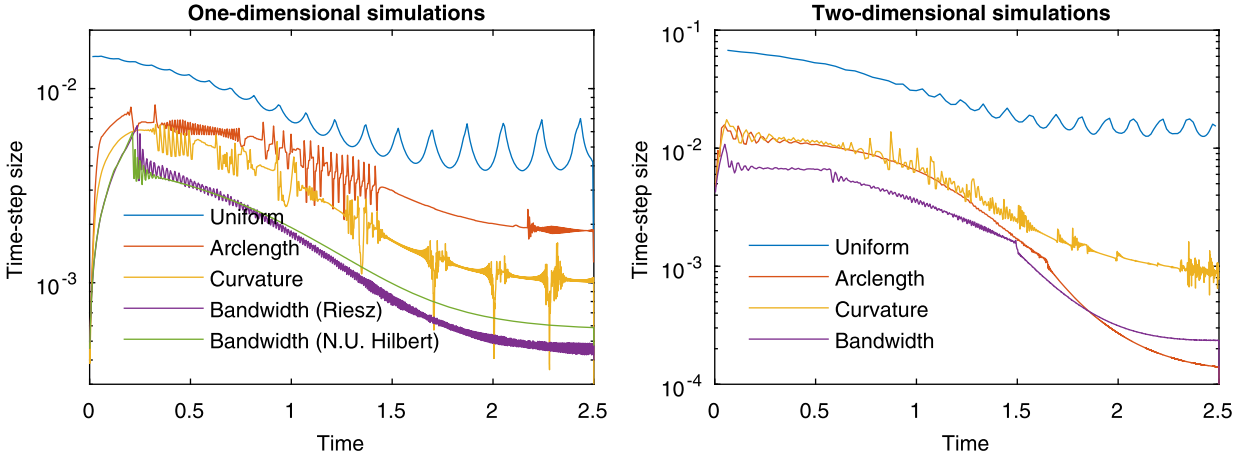

Fig. 4. Time-step sizes for Burgers equation simulations conducted using various mesh specifications. All simulations can be seen to stiffen as the shock front develops.

This model equation leaves solution B unchanged as time progresses, while accounting for mesh movement via (4). The mesh equation advects mesh nodes between the mesh A and mesh B over the interval $t \in [0, 1]$. The error in solution A was then taken to be the maximum absolute difference between it and solution B. To ensure this approach was accurate, the time-stepping error tolerances were set to equal those of the original simulations.

Fig. 3 depicts the results of this convergence analysis. In all cases, mesh adaptation improves the rate of convergence relative to simulations conducted using uniform meshes. Furthermore, the overall error typically improves by many orders of magnitude. Comparing the various mesh density functions, the bandwidth mesh density function performs best, followed by the curvature and arclength mesh density functions in turn. Comparing the one- and two-dimensional simulations, the convergence rates decrease as dimensionality increases. This is expected because the PMA equation does not allow for explicit control of directional mesh node densities, as it takes the determinant of the given monitor function. This means that some mesh density information is lost in multiple dimensions.

5.3. Stiffness

The increased convergence rates evident in Fig. 3 demonstrate the usefulness of mesh adaptation in reducing memory requirements, but total simulation time also depends on the increased stiffness that comes with mesh movement. To examine this, time-step sizes were recorded for simulations conducted without mesh adaptation, and using the arclength, curvature, and bandwidth mesh density functions. The number of solution components and non-zero mesh components were $N_u = 64$ and $N_{\mathbf{x}} = 32$. The absolute and relative time-stepping error tolerances were 10^{-6} in one-dimension, and 10^{-4} in two-dimensions, with both chosen to ensure smoothly varying time-step sizes. Fig. 4 depicts the results of these simulations. All adaptive meshes stiffen the system of equations beyond that of the uniform mesh. In the one-dimensional

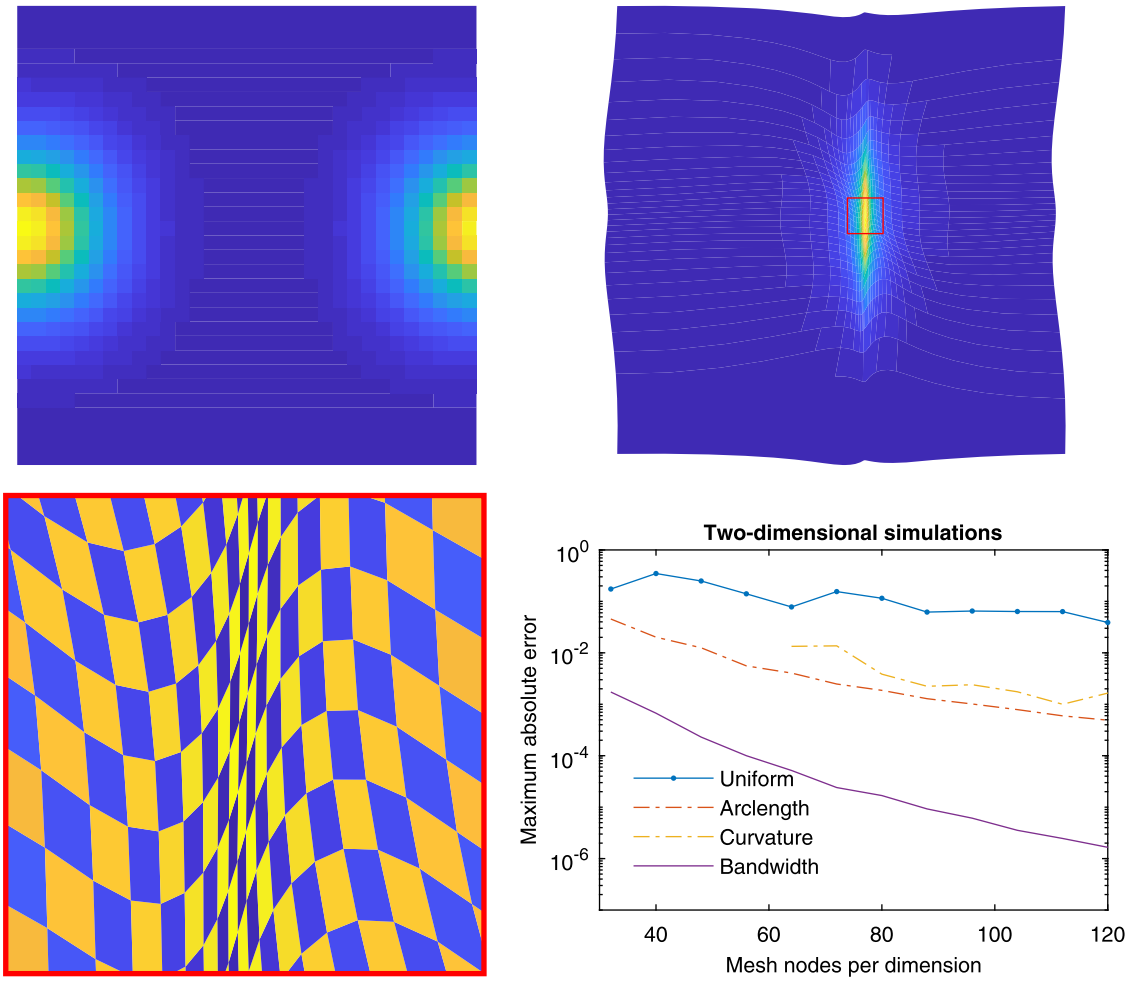

Fig. 5. (Top-left to bottom-left) Advection equation solutions from simulations conducted using the bandwidth mesh density function (3). The model and mesh resolution were $N_u = N_{\mathbf{x}} = 32$. (Top-left) Initial condition and (top-right) final solution for a two-dimensional simulation. (Bottom-left) Close view of the two-dimensional mesh surrounding the shock front region indicated in the top-right subplot. Patches are centred on mesh nodes, and colour intensity indicates the determinant of the mesh Jacobian matrix (i.e. the overall mesh density at that node). Colours alternate between blue and yellow for clarity. The slight vertical distortion in the mesh arises from using fixed (but still periodic) boundary conditions in the mesh equation. (Bottom-right) Error in the advection equation simulations conducted using various mesh specifications. In all cases, adaptive meshes outperformed the simulations conducted using a uniform mesh. The bandwidth mesh density function from this work outperforms both the curvature and arclength mesh density functions. The curvature mesh density function failed to produce stable time-stepping for small values of N_u.

simulations, the bandwidth-based simulation is stiffest, followed in turn by the curvature- and arclength-based simulations. In the two-dimensional simulations, the arclength- and bandwidth-based simulations are not consistently stiffer than one another, but both are stiffer than the curvature-based simulation.

6. Numerical results for heterogeneous advection

To illustrate the generality of bandwidth-based mesh adaptation, a two-dimensional simulation for the advection equation is shown in Fig. 5. The initial condition is a shifted version of the von Mises distribution

$$u(0, \mathbf{x}) = \exp\left[\cos(x_1 - \pi) + \cos(x_2) + \cos(x_1 - \pi)\cos(x_2) - 3\right],$$

and the bandwidth mesh density function is used for mesh adaptation. The initial condition can be seen to compress as it propagates rightwards, forming a sharp peak. As with Burgers' equation, a close-up view of the mesh shows anisotropic adaptation aligned with the sharp peak.

To assess the performance of the bandwidth mesh density function, a convergence analysis like that in §5.2 was performed. All parameters were kept the same, only the model equation was changed. The results of this analysis are presented in Fig. 5. Once again, mesh adaptation improved the rate of convergence relative to simulations conducted using uniform

meshes. Comparing the various mesh density functions, the bandwidth mesh density function performs best, followed by the arclength and curvature mesh density functions in turn.

7. Discussion

The convergence and stiffness results in §5 and §6 illustrate two aspects of moving mesh methods. These are that they aim to reduce the trade-off between accuracy and the number of mesh nodes, but tend to increase the stiffness of the resulting system of equations. The effect of moving mesh methods on the memory requirements and computation time of a simulation is more complex, and depends on problem-specific factors and the hardware that is available for computations. However, some general statements on the matter can be made. For a given number of mesh nodes, the memory usage and number of computations per time-step will be higher for a moving mesh method than a static method. This is because both a model and mesh equation need to be discretised. Reductions in overall memory usage for a given level of accuracy must therefore come from an improvement in the rate of convergence that outweighs this. Improved convergence rates can similarly reduce the number of computations that are performed per time-step. However, a reduction in the overall computation time for a simulation then requires this to outweigh any increase in the number of time-steps that are taken, since moving mesh methods are typically stiffer than static methods. Note that this discussion is not uniquely applicable to bandwidth-based mesh adaptation. For the Fourier spectral moving mesh method in this work, the computational complexity of the bandwidth mesh density function is comparable to that of the arclength and curvature mesh density functions, and so no additional computational penalty is incurred for its use.

To give a sense of what this means in practice, the two-dimensional simulations from §5.2 are considered. To achieve an absolute error of approximately 10^{-3}, the curvature-, arclength-, and bandwidth-based approaches required approximately 7, 10, and 16-times fewer nodes respectively than the static approach. This implies substantial reductions in memory were achieved, even accounting for the extra system of equations that was solved. However, the wall-clock times for the curvature-, arclength-, and bandwidth-based simulations were 5.3, 1.3, and 1.2-times longer that of the uniform simulation. This is largely explained by two factors. The first is the increasing stiffness that comes with mesh adaptation. The number of timesteps the curvature-, arclength-, and bandwidth-based simulations took were respectively 9.8, 2.9, and 3.0-times those of the static simulation. The second factor is the differing number of computations that were performed, since an extra system of equations is solved when the mesh is adaptive.

8. Conclusion

A multidimensional spectral moving mesh method is presented that is based on the local spatial bandwidth of the model solution. The approach is closely aligned with the sampling requirements of spectral methods, whose discretisations depend on the frequency content of the model solution. When applied to a viscous Burgers equation and a heterogeneous advection equation, bandwidth-based mesh adaptation produces convergence rates which exceed those arising from arclength- and curvature-based adaptation. However, mesh adaptation does increase the stiffness of the resulting system of equations relative to a static mesh. Hence, the principal benefit of a moving mesh method is that it will likely reduce the overall memory requirements of a simulation due to improved convergence rates. Any improvements in the overall computation time will require the reduced number of computations arising from smaller problem discretisations to outweigh the increased stiffness arising from mesh adaptation. Nonetheless, bandwidth-based mesh adaptation offers an effective, well-justified technique for implementing spectral moving mesh methods.

Acknowledgements

This work was supported in part by the Engineering and Physical Sciences Research Council (EPSRC), United Kingdom, grant numbers EP/L020262/1, EP/M011119/1, and EP/P008860/1.

References

[1] T.A. Driscoll, N. Hale, L.N. Trefethen, Chebfun Guide, Pafnuty Publications, 2014, http://www.chebfun.org/docs/guide/.
[2] B. Fornberg, The pseudospectral method: comparisons with finite differences for the elastic wave equation, Geophys. 52 (4) (1987) 483–501, https://doi.org/10.1190/1.1442319.
[3] J. Jaros, A.P. Rendell, B.E. Treeby, Full-wave nonlinear ultrasound simulation on distributed clusters with applications in high-intensity focused ultrasound, Int. J. High Perform. Comput. Appl. 30 (2) (2016) 137–155, https://doi.org/10.1177/1094342015581024.
[4] E. Martin, Y.T. Ling, B.E. Treeby, Simulating focused ultrasound transducers using discrete sources on regular Cartesian grids, IEEE Trans. Ultrason. Ferroelectr. Freq. Control 63 (10) (2016) 1535–1542, https://doi.org/10.1109/TUFFC.2016.2600862.
[5] P.S. Georgiou, J. Jaros, H. Payne, C. Allen, T.T. Shah, H.U. Ahmed, E. Gibson, D. Barratt, B.E. Treeby, Beam distortion due to gold fiducial markers during salvage high-intensity focused ultrasound in the prostate, Med. Phys. 44 (2) (2017) 679–693, https://doi.org/10.1002/mp.12044.
[6] W. Huang, R.D. Russell, Adaptive Moving Mesh Methods, Springer, 2011.
[7] L.S. Mulholland, W.Z. Huang, D.M. Sloan, Pseudospectral solution of near-singular problems using numerical coordinate transformations based on adaptivity, SIAM J. Sci. Comput. 19 (4) (1998) 1261–1289, https://doi.org/10.1137/S1064827595291984.
[8] W. Feng, P. Yu, S. Hu, Z. Liu, Q. Du, L. Chen, Spectral implementation of an adaptive moving mesh method for phase-field equations, J. Comput. Phys. 220 (1) (2006) 498–510, https://doi.org/10.1016/j.jcp.2006.07.013.

[9] W. Feng, P. Yu, S. Hu, Z. Liu, Q. Du, L. Chen, A Fourier spectral moving mesh method for the Cahn–Hilliard equation with elasticity, Commun. Comput. Phys. 5 (2–4) (2009) 582–599.
[10] J. Shen, X. Yang, An efficient moving mesh spectral method for the phase-field model of two-phase flows, J. Comput. Phys. 228 (8) (2009) 2978–2992, https://doi.org/10.1016/j.jcp.2009.01.009.
[11] J.J. Tapia, P. Gilberto López, Adaptive pseudospectral solution of a diffuse interface model, J. Comput. Appl. Math. 224 (1) (2009) 101–117, https://doi.org/10.1016/j.cam.2008.04.037.
[12] C.J. Subich, A robust moving mesh method for spectral collocation solutions of time-dependent partial differential equations, J. Comput. Phys. 294 (2015) 297–311, https://doi.org/10.1016/j.jcp.2015.04.003.
[13] E.S. Wise, B.T. Cox, B.E. Treeby, Mesh density functions based on local bandwidth applied to moving mesh methods, Commun. Comput. Phys. 22 (5) (2017) 1286–1308, https://doi.org/10.4208/cicp.OA-2016-0246.
[14] M. Felsberg, G. Sommer, The monogenic signal, IEEE Trans. Signal Process. 49 (12) (2001) 3136–3144, https://doi.org/10.1109/78.969520.
[15] C.P. Bridge, Introduction to the monogenic signal, https://arxiv.org/abs/1703.09199, 2017.
[16] J.J. Clark, M.R. Palmer, P.D. Lawrence, A transformation method for the reconstruction of functions from nonuniformly spaced samples, IEEE Trans. Acoust. Speech Signal Process. ASSP-33 (4) (1985) 1151–1165, https://doi.org/10.1109/TASSP.1985.1164714.
[17] W. Cao, W. Huang, R.D. Russell, A study of monitor functions for two-dimensional adaptive mesh generation, SIAM J. Sci. Comput. 20 (6) (1999) 1978–1994, https://doi.org/10.1137/S1064827597327656.
[18] C.J. Budd, J.F. Williams, Moving mesh generation using the parabolic Monge–Ampère equation, SIAM J. Sci. Comput. 31 (5) (2009) 3438–3465, https://doi.org/10.1137/080716773.
[19] C. Budd, M. Cullen, E. Walsh, Monge–Ampére based moving mesh methods for numerical weather prediction, with applications to the Eady problem, J. Comput. Phys. 236 (2013) 247–270, https://doi.org/10.1016/j.jcp.2012.11.014.
[20] P. Browne, C. Budd, C. Piccolo, M. Cullen, Fast three dimensional r-adaptive mesh redistribution, J. Comput. Phys. 275 (2014) 174–196, https://doi.org/10.1016/j.jcp.2014.06.009.
[21] H. Weller, P. Browne, C. Budd, M. Cullen, Mesh adaptation on the sphere using optimal transport and the numerical solution of a Monge–Ampère type equation, J. Comput. Phys. 308 (2016) 102–123, https://doi.org/10.1016/j.jcp.2015.12.018.
[22] C.J. Budd, R.D. Russell, E. Walsh, The alignment properties of Monge–Ampere based mesh redistribution methods: I linear features, http://arxiv.org/abs/1402.5453, 2014.
[23] J.M. Burgers, A mathematical model illustrating the theory of turbulence, Adv. Appl. Mech. 1 (1948) 171–199, https://doi.org/10.1016/S0065-2156(08)70100-5.
[24] L.F. Shampine, M.W. Reichelt, The MATLAB ODE suite, SIAM J. Sci. Comput. 18 (1) (1997) 1–22, https://doi.org/10.1137/S1064827594276424.
[25] W. Huang, R.D. Russell, Analysis of moving mesh partial differential equations with spatial smoothing, SIAM J. Numer. Anal. 34 (3) (1997) 1106–1126, https://doi.org/10.1137/S0036142993256441.
[26] W. Huang, Practical aspects of formulation and solution of moving mesh partial differential equations, J. Comput. Phys. 171 (2) (2001) 753–775, https://doi.org/10.1006/jcph.2001.6809.