

EFFICIENT COMPUTATION OF TRANSPARENT BOUNDARY CONDITIONS FOR WAVES TRAVELING IN SHEAR FLOW

Algirdas Grybas
School of Engineering and Science
Jacobs University Bremen

Bachelor of Science thesis supervised by Prof. Dr. Marcel Oliver

May, 2007

Abstract

Based on a paper written by Prof. Dr. Marcel Oliver and Prof. Oliver Bühler, this paper examines the efficiency of computation of transparent boundary conditions for waves traveling in shear flow, represented as grid points on a truncated lattice. An improved Gradient-Penalty method is suggested over the QR decomposition, what, in the particular linear system whose matrix is similar to a unitary matrix, significantly reduces the computational costs. Theoretical comparison of computational costs and a pseudocode for implementing the Gradient-Penalty method are provided.

Acknowledgements

I am deeply indebted to my supervisor Prof. Dr. Marcel Oliver for suggesting the topic and guiding my research throughout the year. He provided a very useful insight into underlying physical concepts and motivated further research by clearly formulating research aims. His constant support and help in directing the paper significantly facilitated my research which would not have been as successful otherwise.

Keywords: transparent boundary condition, Gradient method, Penalty method, Ornstein-Uhlenbeck process

Contents

1	Introduction	3
2	Set-up	3
3	Least-norm solution	5
3.1	QR decomposition	6
3.2	Gradient method	6
3.3	Penalty method	7
3.4	Combined Gradient-Penalty method	8
3.5	Performance comparison	8
3.6	Other methods	9
	Appendices	11
A	Computational costs	11
B	Ornstein-Uhlenbeck process	11
B.1	Review	11
B.2	Brownian motion	12
B.3	Ornstein-Uhlenbeck process	13
C	Penalty-Gradient pseudocode	15
	References	17

1 Introduction

This paper is based on a working paper being written by Prof. Dr. Marcel Oliver and Prof. Oliver Bühler [1], in which they develop a notion of transparent boundary conditions for traveling waves in shear flow. Results are then tested using a time-dependent velocity field in which Fourier coefficients of the solution to the lattice ODE are driven by stochastic Ornstein-Uhlenbeck (OU) processes. By doing so, one can try to examine the relationship between the energy dissipation rate and the exit time of the OU process. Although the latter analysis is beyond the scope of this paper, a short overview of theory on stochastic processes leading to the definition and some properties of the OU process is introduced in the Appendix B as the reader may not be familiar with this particular stochastic process due to its relatively specific applications.

The paper primarily consists of two parts. First of all, a motivation for researching efficient ways of computing the transparent boundary conditions and a brief overview of the mathematical concepts behind the physical setting of waves traveling in shear flow are given. We introduce the lattice ODE being analyzed, its solution is presented in the Fourier transform leading to the formulation of the transparent boundary conditions. Efficient computation of the latter is the primary focus of this paper. Therefore, in the second part various methods suitable for computing the transparent boundary conditions are presented. The QR decomposition, currently employed in [1] to calculate the transparent boundary conditions, is discussed and its drawbacks are pointed out. Then, a new method - the combined Gradient-Penalty method - is introduced and its advantages over the QR decomposition are outlined. It turns out that in the particular linear system being analyzed, a single iteration of the Gradient method is sufficient to find a good approximation of the solution to the system. This allows one to replace the QR decomposition with a Gradient(-Penalty) method, thus significantly reducing the computational cost. Theoretical estimations of the computational costs are provided and confirm the significant improvement. Finally, other methods that could have been used are also overviewed and reasons for not choosing them are given. The reader will also find a pseudocode for the implementation of the improved method in the Appendix C.

2 Set-up

Due to the importance and wide range of applications of waves in various fields, the concept of traveling waves in a shear flow has been extensively studied in numerous settings (see, for example, [4], [11], [12]). In this paper, we take a different approach and discretize the traveling waves as grid points on an infinite lattice. This allows for wider mathematical analysis without requiring a lot of attention and knowledge of underlying physical concepts. In order to analyze the traveling waves on a computational domain and simulate the process numerically, however, dealing with infinite lattices is clearly not feasible. Since computer memory, processor speed and time available are always finite, one would never be able to numerically simulate waves on an infinite lattice. A truncated lattice must be considered instead, including as many known grid points as possible. The next ones (beyond the grid) need to be extrapolated from the available data. In doing so, two issues arise. First, one needs to decide *how* to extrapolate the off-grid points. In this paper, we apply linear extrapolation using the theoretical solutions of the lattice ODE. Second, one has to keep in mind that should the boundary exist in reality, some waves (in other words, the energy carried by them) would be reflected. This would be caused by the artificial truncation of the lattice what is not a part of the initial setting. Therefore, one must ensure the energy is *not* reflected at the boundary and the entire set-up remains unchanged after the introduction of the artificial

boundary. In this paper, we try to do this by formulating transparent boundary conditions which make sure that (at least the fastest) waves travel through the boundary without any reflection. Since the conditions must be satisfied for every sample wave being considered at every time step, computation of them involves sufficiently many operations to become one of the bottlenecks of the whole simulation. Efficient computation of the transparent boundary conditions is therefore of high importance for the analysis and requires particular focus.

We consider the advection equation

$$\partial_t \theta + \mathbf{u} \cdot \nabla \theta = 0 \quad (1)$$

on a doubly-periodic domain $\mathbb{T}^2 \equiv [0, 2\pi]^2$ with a velocity field

$$\mathbf{u}(x) = \begin{pmatrix} u(y) \\ 0 \end{pmatrix} \quad (2)$$

Writing (1) in Fourier representation,

$$\partial_t \theta_{kl} + ik \sum_{m+n=l} u_m \theta_{kn} = 0, \quad (3)$$

where $f_k = \frac{1}{2\pi} \int_0^{2\pi} e^{-ikx} f(x) dx$. Let $k = 1$ and assume the solution of (3) is of the form

$$\theta_n = e^{i\omega t} e^{i\xi(l-m)} = e^{i\omega t} e^{i\xi l} e^{-i\xi m}, \quad (4)$$

where an explicit formula for $\omega(\xi)$ is known *a priori*, and $m+n=l$ as above. Thus, θ_n , treated further as grid values or coordinates, is a function of ξ . At a particular moment in time (t fixed), denoting $\kappa = e^{-i\xi}$ and fixing ξ , we note the one-to-one correspondence between θ_n and κ^m . We can rewrite (3) in the form

$$\dot{\theta}_l = \sum_{j=-M}^M c_j \theta_{l-j} \quad (5)$$

where $c_j = -iku_j$. Substituting (4) into (5) leads to the *dispersion relation*

$$\omega(\xi) = -i \sum_{j=-M}^M c_j \kappa^{-j} \quad (6)$$

the derivative of which, $\omega'(\xi)$, is the *group velocity* at which the energy is traveling along the lattice¹. By varying m in (3), we get S different solutions which correspond to S sample points taken before the boundary that need to be linearly extrapolated to $M < S$ points after the boundary. By considering S^{out} waves going in one direction (the situation with the rest of the waves going to the other direction is analogous, so we disregard those for now), we obtain $S^{out} \times S$ solutions that can be represented in a matrix form

$$B = \begin{pmatrix} \kappa_1^1 & \cdots & \kappa_1^S \\ \vdots & & \vdots \\ \kappa_{S^{out}}^1 & \cdots & \kappa_{S^{out}}^S \end{pmatrix} \quad (7)$$

Taking sufficiently many sample points, one can assume that vectors $v_i = (\kappa_i, \kappa_i^2, \dots, \kappa_i^S)$ are linearly

¹For a more detailed introduction to the concept of group velocity, please refer to [15], pp. 113-136

independent and span the entire space of solutions θ_n , and thus of all wavenumbers ξ . They form a basis and the matrix B can be used to linearly extrapolate all points before the boundary to those after the boundary, as follows:

$$\text{Find } z \text{ such that } B^H z = \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_S \end{pmatrix}; \quad (8)$$

$$\text{Compute } \begin{pmatrix} \theta_{S+1} \\ \vdots \\ \theta_{S+M} \end{pmatrix} = C^H z, \text{ where } C = \begin{pmatrix} \kappa_1^{S+1} & \cdots & \kappa_1^{S+M} \\ \vdots & & \vdots \\ \kappa_{S^{out}}^{S+1} & \cdots & \kappa_{S^{out}}^{S+M} \end{pmatrix} \quad (9)$$

Equations (8) and (9) together formulate the **transparent boundary conditions** and efficient computation of the system in (8) is the primary focus of this paper². Since the Fourier representation of the solution of the advection equation (1) involves complex numbers, we have stated the transparent boundary condition in the complex plane as well, superscript H denoting a conjugate transpose matrix. Clearly, the size of the system is determined by S^{out} and S values. There are two ways one can specify them. On one hand, one can fix S^{out} different wave numbers, then let $S = S^{out}$ to obtain a square matrix B and thus make the entire system solvable exactly. However, not knowing S *a priori* is not desirable programming-wise as this gives rise to index confusion, especially when extending the setup to higher dimensions. Alternatively, one can specify S , the number of sample points, and let S^{out} be sufficiently large ($S^{out} > S$) so that the system is solvable³. In this case, one has to deal with an underdetermined $S \times S^{out}$ system in (8) which has infinitely many solutions. We are thus looking for a *minimal* z in the *least-norm* sense:

$$\text{Minimize } \|z\|_W^2 = z^H W z \text{ such that } B^H z = \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_S \end{pmatrix} \quad (10)$$

where $W = \text{diag}(|\omega'(\xi_1)|^{-1}, \dots, |\omega'(\xi_{S^{out}})|^{-1})$ is the weight matrix of group velocities accounting for dispersive waves traveling at different speeds. It is constructed in such a way that the waves having the smallest speed (and, thus, least energy) are penalized most. Using such a weighted norm, at the boundary only the waves with large energy, which contribute most to the energy reflection, are extracted. Then, if the transparent boundary condition is satisfied for them, the obtained set-up is already rather close to the ideal case.

3 Least-norm solution

There are numerous methods developed to solve underdetermined systems⁴. In this paper we assert that the combined Gradient-Penalty method is superior to the currently employed direct QR decomposition in solving the particular problem (10). In the following sections we give a brief introduction to the QR decomposition and its current implementation as well as propose the new Gradient-Penalty method. This is followed by a theoretical efficiency comparison of the two methods and an overview of potential alternatives to the proposed solution.

²For a more detailed description of the set-up, please see [1], pp. 3-9

³Discussion of other approaches can be found in [1], 4.2 - 4.4

⁴For a detailed presentation of solutions to least-square problems, please refer to [8], [7], [2], and [14]

3.1 QR decomposition

Since B is rectangular, the solution to the problem (10) can be obtained using the direct QR decomposition to solve (8). Given a matrix $B \in \mathbb{C}^{m \times n}$, $m \geq n$, find a factorization $B = QR$, where $Q \in \mathbb{C}^{m \times n}$ is orthogonal and $R \in \mathbb{C}^{n \times n}$ is upper trapezoidal. The entries of matrices Q and R can be computed using suitable transformation matrices (e.g. Givens or Householder⁵) or using the (modified) Gram-Schmidt orthogonalization⁶. To use the latter, one needs a set of linearly independent vectors (recall that v_i , rows of the matrix B , are linearly independent). Then solve the system

$$z = R^{-1}Q^H\theta \quad (11)$$

If the weight matrix of group velocities is taken into account, applied QR decomposition leads to the following solution:

$$QR = W^{-1/2}B \quad (12)$$

$$\begin{aligned} z &= W^{-1/2}W^{-1/2}B^H(BW^{-1/2}W^{-1/2}B^H)^{-1}\theta \\ &= W^{-1/2}QR(R^HQ^HQR)^{-1}\theta \\ &= W^{-1/2}QRR^{-1}R^{-H}\theta \\ &= W^{-1/2}QR^{-H}\theta \end{aligned} \quad (13)$$

In order to compute mutually orthogonal vectors, the Gram-Schmidt orthogonalization requires a lot of calculations, increasing the entire computational cost of QR decomposition to mn^2 operations⁷. Since B has a full rank, the Gram-Schmidt orthogonalization could be replaced with Cholesky factorization, reducing the number of operations by a factor of $\frac{1}{3}$. However, since $m \geq n$, this is still of order $O(n^3)$.

3.2 Gradient method

An alternative to QR decomposition in solving (10) is the iterative Gradient method used for minimization problems. In this case, we need to minimize $\phi(z) = \|z\|_W^2$. The idea behind the Gradient method is that, upon choice of an initial point, one needs to compute the direction (from the initial point) in which the given function decreases fastest, then find the minimum along that line and repeat the procedure from the newly computed minimum point. If one imagines the function as having level surfaces (a range of argument values where the function value is the same), the direction of steepest descent is orthogonal to the level surface. Thus, in our setting, if we fix z_0 , the next step should be in the direction of $\nabla\phi(z)$, as the function has its steepest descent (i.e. decreases at a fastest rate) in that direction. We first determine the direction:

$$\nabla\phi(z) = z^HW + Wz = (W^H + W)z = 0 \quad (14)$$

The next position could thus be represented as

$$z_{k+1} = z_k + \alpha_k \nabla\phi(z) = z_k + \alpha_k (W^H + W)z_k \quad (15)$$

where $\alpha_k \in \mathbb{R}$ is the stepsize (distance from the initial point to the minimum along $\nabla\phi(z)$). Using (15),

⁵[14], p. 204

⁶Details in [14], p. 83

⁷[14], p. 83

the next value of $\phi(z)$ is

$$\begin{aligned}
\phi(z_{k+1}) &= (z_k + \alpha_k(W^H + W)z_k)^H W (z_k + \alpha_k(W^H + W)z_k) \\
&= (z_k^H + z_k^H \alpha_k(W^H + W)^H) W (z_k + \alpha_k(W^H + W)z_k) \\
&= z_k^H W z_k + z_k^H W \alpha_k(W^H + W)z_k + z_k^H (W + W^H) \alpha_k W z_k \\
&\quad + z_k^H (W + W^H) \alpha_k W \alpha_k (W^H + W) z_k
\end{aligned} \tag{16}$$

In order to find the minimum of $\phi(z_{k+1})$ along $\nabla\phi(z)$, we differentiate $\phi(z_{k+1})$ in terms of α_k and get

$$\begin{aligned}
\frac{\partial\phi}{\partial\alpha_k}(z_{k+1}) &= z_k^H W (W^H + W) z_k + z_k^H (W + W^H) W z_k + z_k^H (W + W^H) W \alpha_k (W^H + W) z_k + \\
&\quad z_k^H (W + W^H) \alpha_k W (W^H + W) z_k
\end{aligned} \tag{17}$$

$\phi(z_{k+1})$ is minimal if (17) is equal to 0. We can therefore determine the expression for α_k in terms of available data, W and z_k :

$$\begin{aligned}
\alpha_k &= -\frac{z_k^H W (W^H + W) z_k + z_k^H (W + W^H) W z_k}{z_k^H (W + W^H) W (W^H + W) z_k + z_k^H (W + W^H) W (W^H + W) z_k} \\
&= -\frac{z_k^H W (W^H + W) z_k + z_k^H (W + W^H) W z_k}{2z_k^H (W + W^H) W (W^H + W) z_k}
\end{aligned} \tag{18}$$

The computational cost of a single iteration of the Gradient method is of order $O(n^2)$ ⁸, but many iterations may be needed if the initial guess is bad (or the matrix B is ill-conditioned)⁹. However, in Section 3.5 we argue that the matrix B is well-conditioned and so it is reasonable to expect a good initial guess so that a single iteration of the Gradient method outputs a result rather close to the solution. If this is the case, the Gradient method has a significant edge over the currently implemented QR decomposition.

3.3 Penalty method

Recall that the problem of minimizing $\phi(z)$ stemmed from the transparent boundary conditions (8) and (9). Thus, whatever z is determined using the Gradient method, it must still satisfy (8). In general, this is termed as a *constrained optimization* problem:

$$\text{Minimize the function } f(x) \text{ under the constraint that } g(x) = 0 \tag{19}$$

Penalty method is an iterative method that allows one to transform the constrained optimization problem into an unconstrained one by requiring to minimize a newly constructed *Lagrangian function*

$$\mathcal{L}(x_k, \delta_k) = f(x_k) + \frac{1}{2} \delta_k \|g(x_k)\|_2^2 \tag{20}$$

By including the *penalty parameter* δ_k , the Penalty method penalizes for not satisfying the constraint condition. On the other hand, if the constraint is satisfied, i.e. $g(x_k) = 0$, minimization of the Lagrangian function is equivalent to minimizing the given function $f(x_k)$. As δ_k increases in every iteration step, the

⁸[14], p. 123

⁹Please refer to Section 3.6 for more details.

algorithm used to minimize $f(x_k)$ is forced to satisfy the constraint condition, as otherwise minimization of the Lagrangian function is ever harder to achieve.

3.4 Combined Gradient-Penalty method

In our set-up, the minimization of $\|z\|_W^2$ is under the constraint that $B^H z = \theta$. Applying the Penalty method, we transform the constrained optimization problem to an unconstrained one by defining the Lagrangian function

$$\mathcal{L}(z_k, \delta_k) = z_k^H W z_k + \frac{1}{2} \delta_k \|B^H z_k - \theta\|_2^2 \quad (21)$$

In Section 3.5 we will argue that a single iteration of the Gradient-Penalty method suffices to achieve a good approximation to the solution. Nevertheless, in order to emphasize the iterative nature of the method, we keep the indices k for all variables that in general change in every iteration. We now apply the Gradient method to minimize this Lagrangian function. We once again determine the required elements of the Gradient method:

$$\begin{aligned} \mathcal{L}(z_k, \delta_k) &= z_k^H W z_k + \frac{1}{2} \delta_k (z_k^H B B^H z_k - \operatorname{Re}(z_k^H B \theta) - \operatorname{Re}(\theta^H B^H z_k) - \theta^H \theta) \\ &= z_k^H (W + \frac{1}{2} \delta_k B B^H) z_k - \delta_k \theta^H B^H z_k - \frac{1}{2} \delta_k \theta^H \theta \end{aligned}$$

Denote $A = W + \frac{1}{2} \delta_k B B^H$ and $b = \delta_k B \theta$. Then

$$\begin{aligned} \mathcal{L}(z_k, \delta_k) &= z_k^H A z_k - b^H z_k - \frac{1}{2} \delta_k \theta^H \theta \\ \nabla \mathcal{L}(z_k) &= (A^H + A) z_k - b \\ z_{k+1} &= z_k + \alpha_k \nabla \mathcal{L}(z_k) \\ \frac{\partial \mathcal{L}(z_{k+1})}{\partial \alpha_k} &= \nabla \mathcal{L}(z_{k+1}) \cdot \nabla \mathcal{L}(z_k) \\ &= d_k^H (A^H + A) z_k + \alpha_k d_k^H (A^H + A) d_k, \text{ where } d_k = \nabla \mathcal{L}(z_k) \\ \alpha_k &= -\frac{d_k^H (A^H + A) z_k}{d_k^H (A^H + A) d_k} \end{aligned}$$

3.5 Performance comparison

Before the performance comparison of the two methods - QR decomposition and Gradient-Penalty - are discussed, we note some specific features of the matrix B that make the system being investigated rather special. First, we note that B , as defined in (7), consists of entries $\kappa^j = e^{-ij\xi}$. In the particular case when $S = S^{out}$, B contains all coefficients of the discrete Fourier representation of the solution (4) to the lattice ODE, what makes it (a multiple of) a unitary matrix. In that case,

$$\|B\|_2 = \sqrt{B^H B} = \sqrt{I} = 1 \quad (22)$$

since $B^H = B^{-1}$ and so the condition number

$$K_2(B) = \|B\|_2 \|B^{-1}\|_2 = 1 \quad (23)$$

Since in general $S \leq S^{out}$, B is not unitary. However, note that every entry $e^{-ij\xi} = \cos(-j\xi) + i \sin(-j\xi)$

has norm 1. Thus, up to normalization, $\|B\|\|B^H\| \approx 1$, implying that B^H is (up to normalization) a pseudoinverse of B . Thus, its structure is close to that of a unitary matrix and so one would expect its condition number to be close to 1 as well¹⁰. Whatever the actual structure of B and its exact condition number, we can claim it is well-conditioned. Consequently, the system does not have irregular or rather stretched level surfaces (that could lead to a bad initial guess, as discussed in Section 3.6). Thus, independent of the initial point chosen, the direction orthogonal to the level surface at that point should point quite close to the minimum of the function. Therefore, after one iteration we can expect to be rather close to the actual solution. This proves to be very important in the following performance comparison.

It is also worth mentioning that the entire system is being observed at discrete and relatively short time intervals. Therefore, the solution to the new system (in the next time step) should be close to the previous one (the waves travel continuously). Thus, we can take the approximate minimum point, computed in a single iteration in the previous time step, use it as the initial guess in the next time step, and be confident that it is a good guess and is rather close to the new minimum. Even if the computation in the very first time step does not bring us very close to the solution, computations in successive time steps definitely will, as the system is not changing much.

QR decomposition is a direct method (in contrast to the Gradient method which is iterative), so it is possible to determine the exact number of operations needed for any size of the system. In our setting, to solve the $S^{out} \times S$ system, QR decomposition would require $S^{out} \times S^2$ operations¹¹. Since in general $S^{out} \geq S$, the computational cost is of order $O(S^3)$. In comparison, a single iteration of the Gradient method solving the given system would require around $8(S^{out})^2 + 2S \cdot S^{out} + 5S^{out}$ operations¹², i.e. the computational cost is of order $O((S^{out})^2)$. So, if we can replace the QR decomposition with a single iteration of the Gradient-Penalty method and obtain relatively good solution (and we have proven above that we can), the computational cost decreases from $O(S^3)$ to $O((S^{out})^2)$, what is a significant improvement and the main result of this paper.

3.6 Other methods

Apart from the direct QR decomposition and iterative Gradient and Penalty methods, discussed above, there are numerous other methods that could be used to compute the transparent boundary conditions. In what follows, a short overview of the most renowned alternatives is given and reasons for not using them are outlined. The standing argument is that the given system is relatively well-conditioned and the iterative procedure needs to be run only once.

As an alternative to the suggested Gradient method, its more advanced counterpart, called the (preconditioned) *Conjugate Gradient (CG) method*, could have been used. When solving a general system $Ax = b$, the CG method builds on the Gradient method, but uses pairwise A-orthogonal directions along which the minimum of the function is searched for afterwards. In case of a bad initial guess (i.e. where the level surfaces of the function are widely stretched), the simple Gradient method would involve numerous iterations, since the directions normal to the level surfaces would not point anywhere close to the actual solution. Thus, although still convergent, Gradient method would perform badly for large systems with stretched level surfaces. CG method would have a significant edge in such a case, because no matter how bad the first calculated direction is, the next one would be A-orthogonal to it and thus would have to point relatively close to the solution. If the first choice was good, CG method would behave similarly to

¹⁰For more details, please refer to [1], p. 12

¹¹[14], p. 83

¹²Please refer to Appendix A for details.

the Gradient method and would be relatively close to the solution already after the first iteration. If the initial system is ill-conditioned but the matrix has a specific structure (e.g. symmetric positive definite), a preconditioning matrix P can be used to transform the initial problem $Ax = b$ into $P^{-1}Ax = P^{-1}b$, limiting the ill-conditioning of the system and improving performance of the CG method. However, as proven above, in our setting the matrix B is rather well-conditioned. Therefore, there is no reason to expect low performance from the basic Gradient method. Thus, by taking previously-computed solution as the initial guess in the next iteration, we can expect to be already close to the sought-after solution where one iteration should suffice to obtain a relatively good minimum point. The errors introduced by stopping the iteration after one loop are already relatively small compared to other major sources of error - discretization of the dispersion relation and positioning of the system close to the wave source. In theory, the dispersion relation (6) involves a continuum of ξ values. However, for computational reasons we immediately discretize it by introducing the Fourier representation of the solution. Moreover, we then consider only a relatively small number S of sample grid points to use for the extrapolation which are not necessarily representative of all the grid points. The second major error arises by considering the system near the wave source. Due to computational constraints, one cannot consider a huge sample so that S sample grid points closest to the boundary are far from the source. Therefore, one needs to start extrapolating close to the source, where the system is not stable and so potentially significant fluctuations can distort the extrapolation procedure. Also, the fact that the system is considered close to the source limits the number of sample points S that can be taken, increasing the error even more. Taking these two sources of error into account, we can claim that the error introduced by approximate computation of the minimum point is relatively small and does not influence the efficiency of the entire system much. Thus, the benefits of the CG method provided during a large number of iterations solving ill-conditioned problems should not be significant in this particular setting. We realize the general superiority of the CG method, but do not think that due to additional computational cost it is worth implementing in this case.

The suggested Penalty method could be replaced by a more sophisticated *Lagrange multiplier method*. The latter adds an additional term $\lambda_k^H h(x)$ to the Lagrangian function, requiring the minimization of the *augmented Lagrangian function*

$$\mathcal{G}_{\delta_k}(x, \lambda_k) = f(x) + \lambda_k^H h(x) + \frac{1}{2}\delta_k \|h(x)\|_2^2, \quad (24)$$

where $\lambda_k \in \mathbb{C}^n$. Once again, the satisfied constraint condition $h(x) = 0$ would lead to the initial minimization of the function $f(x)$. If the constraint is not satisfied, however, the Lagrange multiplier method provides an additional degree of freedom λ_k . One can thus fine-tune the penalization more accurately and adjust the process in case of some specific features of the constraint function. Moreover, the Lagrange multiplier method is said to converge more than linearly if $\delta_k \rightarrow \infty$ as $k \rightarrow \infty$ (linear convergence if δ_k bounded). Also, contrary to the Penalty method, $\delta_k \rightarrow \infty$ condition is no more required (one can fine-tune using λ_k as well), so ill-conditioning of the problem is limited. Finally, the convergence rate of the Lagrange multiplier method is independent of the growth rate of the penalty parameter¹³. However, although the Lagrange multiplier method clearly has numerous advantages over the Penalty method, they are not expected to bring significant performance boost in our setting. As mentioned above, the given system $B^H z = \theta$ is relatively well-conditioned, so the additional parameter to tune the penalization procedure is not necessary. The same argument makes the limitation of ill-conditioning of the problem of little use. Finally, since the Penalty method is to be applied only once (only one iteration

¹³For a more detailed discussion of the Lagrange multiplier method, please refer to [14], pp. 317-319

of the Gradient method is performed), the rate of convergence does not matter much. Once again, we conclude that although the Lagrange multiplier method may be superior for general situations, in our case the additional computational cost would exceed the performance gains.

Appendix A Computational costs

In the following tables the computational costs of matrix multiplications, used in the Gradient-Penalty method, are given explicitly. As in the entire paper, $B \in \mathbb{C}^{S^{out} \times S}$, $z_k \in \mathbb{C}^{S^{out} \times 1}$, $W \in \mathbb{R}^{S^{out} \times S^{out}}$, $A \in \mathbb{C}^{S^{out} \times S^{out}}$, $b \in \mathbb{C}^{S^{out} \times 1}$, $\theta \in \mathbb{C}^{S \times 1}$, $\delta_k \in \mathbb{R}$.

Operation	No. of additional operations
BB^H	$(2S - 1)(S^{out})^2$
$\delta_k BB^H$	S^{out}
$W + \delta_k BB^H$	$(S^{out})^2$
$A^H + A$	$(S^{out})^2$
$\nabla \mathcal{L}(z_k)$	$2(S^{out})^2$
$d_k^H (A^H + A) z_k$	$2(S^{out})^2 + S^{out} - 1$
$d_k^H (A^H + A) d_k$	$2(S^{out})^2 + S^{out} - 1$

So, the computational cost of A is $2S(S^{out})^2 + S^{out}$; cost of b is $2S \cdot S^{out}$; of stepsize α_k - $4(S^{out})^2 + 2S^{out}$. In order to compute the next z_k one needs $2S^{out}$ operations more. Thus, the total cost of a single Gradient-Penalty iteration is $(2S + 7)(S^{out})^2 + 2S \cdot S^{out} + 5S^{out}$. Since A can be precomputed before the iteration, the most costly computation of BB^H is taken out of the iteration, reducing the cost to $8(S^{out})^2 + 2S \cdot S^{out} + 5S^{out}$, what is of order $O((S^{out})^2)$.

Appendix B Ornstein-Uhlenbeck process

B.1 Review

Definition 1. A *probability space* is a triple (Ω, \mathcal{A}, P) consisting of an arbitrary non-empty set Ω (sample/phase space), a σ -algebra \mathcal{A} of subsets of Ω , called events, and a probability measure P on \mathcal{A} , s.t. $\forall A \in \mathcal{A}, P(A) \in [0, 1]$.

In what follows Ω is assumed to be \mathbb{R}_+ .

Definition 2. A *stochastic process* $X = \{X(t), t \in T\}$ is a collection of random variables on a common probability space (Ω, \mathcal{A}, P) indexed by $t \in T \subset \mathbb{R}$ (time).

If T is countable, X is said to be a discrete(-time) stochastic process; if T is an interval, X is said to be a continuous(-time) stochastic process. Distinct stochastic processes $X = \{X(t), t \in T\}$ and $Y = \{Y(t), t \in T\}$ with the same distribution functions are equivalent.

Definition 3. A *probability density function* for a random variable X is a non-negative function $f(x)$ such that

$$P\{a < X \leq b\} = \int_a^b f(x) dx \text{ for } -\infty < a < b < \infty \quad (25)$$

Definition 4. The *distribution function* of the random variable X is the function $F(x)$ defined as

$$F(x) = P\{X \leq x\}, \quad -\infty < x < \infty \quad (26)$$

If X has a probability density function $f(x)$, then X is continuous and its distribution function

$$F(x) = \int_{-\infty}^x f(\xi) d\xi, \quad -\infty < x < \infty \quad (27)$$

Definition 5. The *joint distribution function* of random variables X, Y is the function F_{XY} of two real variables given by

$$F_{XY}(x, y) = F(x, y) = P \{X \leq x \text{ and } Y \leq y\} \quad (28)$$

F_{XY} is said to possess a (joint) probability density if there exists a function f_{XY} for which

$$F_{XY}(x, y) = \int_{-\infty}^x \int_{-\infty}^y f_{XY}(\xi, \eta) d\eta d\xi \quad \forall x, y \in \mathbb{R} \quad (29)$$

A random vector X_1, \dots, X_n is said to have a joint normal distribution, if every linear combination $\alpha_1 X_1 + \dots + \alpha_n X_n$, $\alpha_i \in \mathbb{R}$, has a univariate normal distribution. If X and Y are jointly distributed, $E[X + Y] = E[X] + E[Y]$.

B.2 Brownian motion

Definition 6. *Brownian motion* initiated at x with diffusion coefficient σ^2 on a probability space (Ω, \mathcal{A}, P) is a stochastic process $\{B(t); t \geq 0\}$ such that

- $B(0) = x$ almost everywhere and $B(t)$ is continuous as a function of t .
- Every increment $B(t + s) - B(t)$ is normally distributed with mean zero and variance $\sigma^2 t$; $\sigma^2 > 0$ is a fixed parameter.
- $\forall 0 = t_0 < t_1 < \dots < t_n$ the increments $B(t_n) - B(t_{n-1}), \dots, B(t_1) - B(t_0)$ are independent and normally distributed with mean $E[B(t_i) - B(t_{i-1})] = 0$ and variance $E[(B(t_i) - B(t_{i-1}))^2] = t_i - t_{i-1}$.

Standard Brownian motion is obtained by dividing $B(t)$ by σ . Then $\sigma^2 = 1$. Note that each path of a continuous Brownian motion stochastic process is (almost surely) differentiable at no point.

Definition 7. Let T be an abstract set and $\{X(t); t \in T\}$ a stochastic process. $\{X(t); t \in T\}$ is a **Gaussian process** if for every $n = 1, 2, \dots$ and every finite subset $\{t_1, \dots, t_n\}$ of T , the random vector $(X(t_1), \dots, X(t_n))$ has a joint normal distribution.

Every Gaussian process is uniquely described by its mean and covariance function:

$$\mu(t) = E[X(t)] \text{ for } t \in T \quad (30)$$

$$\Gamma(s, t) = E[\{X(s) - \mu(s)\} \{X(t) - \mu(t)\}] \text{ for } s, t \in T \quad (31)$$

The covariance function is positive definite: $\forall n \in \mathbb{N}$, $\alpha_i \in \mathbb{R}$, $t_i \in T$,

$$\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j \Gamma(t_i, t_j) \geq 0 \quad (32)$$

Definition 8. Let $\{B(t); t \geq 0\}$ be a standard Brownian motion process ($\sigma^2 = 1$), and let μ and $\sigma > 0$ be fixed. The **Brownian motion with drift parameter μ and variance parameter σ^2** is the process

$$X(t) = \mu t + \sigma B(t) \text{ for } t \geq 0 \quad (33)$$

In the field of Finance, μ is the (risk-free) rate of return, and σ is the volatility of the portfolio.

Definition 9. A function $G(t)$ is called nonanticipating function of t if for all $s > t$, $G(t)$ is statistically independent of $W(s) - W(t)$.

Lemma 1. If $G(t)$ and $H(t)$ are arbitrary continuous nonanticipating functions, then

$$E\left[\int_{t_0}^t G(t')dW(t') \int_{t_0}^t H(t')dW(t')\right] = \int_{t_0}^t dt E[G(t')H(t')] \quad (34)$$

Proof. Notice that

$$\begin{aligned} E\left[\sum_i G_{i-1}\Delta W_i \sum_j H_{j-1}\Delta W_j\right] &= E\left[\sum_i G_{i-1}H_{i-1}(\Delta W_i)^2\right] \\ &+ E\left[\sum_{i>j} (G_{i-1}H_{j-1} + G_{j-1}H_{i-1})\Delta W_j\Delta W_i\right] \end{aligned}$$

In the second term, ΔW_i is independent of all other terms since $i > j$, and G and H are nonanticipating. Hence, we may factorise out the term $E[\Delta W_i] = 0$ so that this term vanishes. Using $E[(\Delta W_i)^2] = \Delta t_i$, we obtain

$$\begin{aligned} \lim_{\Delta t_i \rightarrow 0} E\left[\sum_{i=t_0}^t G_{i-1}H_{i-1}(\Delta W_i)^2\right] &= \lim_{\Delta t_i \rightarrow 0} E\left[\sum_{i=t_0}^t G_{i-1}H_{i-1}\right]E[(\Delta W_i)^2] \\ &= \lim_{\Delta t_i \rightarrow 0} E\left[\sum_{i=t_0}^t G_{i-1}H_{i-1}\right]\Delta t_i \end{aligned}$$

and so the result follows. \square

B.3 Ornstein-Uhlenbeck process

The Ornstein-Uhlenbeck process $\{V(t); t \geq 0\}$ with a drift coefficient $\beta > 0$ and a diffusion parameter σ^2 is defined in terms of a standard Brownian motion $B(t)$ by scale changes in space and time:

$$V(t) = \nu e^{-\beta t} + \frac{\sigma e^{-\beta t}}{\sqrt{2\beta}} B(e^{2\beta t} - 1) \text{ for } t \geq 0 \quad (35)$$

$V(0) = \nu$ if $B(0) = 0$.

The Ornstein-Uhlenbeck process is a continuous-state-space, continuous-time Markov process having continuous paths. It is a Gaussian process with

$$E[V(t)|V(0) = \nu] = \nu e^{-\beta t} \quad (36)$$

and

$$\begin{aligned}
\text{Var}[V(t)|V(0) = x] &= e^{-2\beta t} \frac{\sigma^2}{2\beta} \text{Var}[B(e^{2\beta t} - 1)] \\
&= \sigma^2 \left(\frac{1 - e^{-2\beta t}}{2\beta} \right)
\end{aligned} \tag{37}$$

The covariance between two Ornstein-Uhlenbeck processes is

$$\begin{aligned}
\text{Cov}[V(u), V(s)] &= E[\{V(u) - \nu e^{-\beta u}\}\{V(s) - \nu e^{-\beta s}\}] \\
&= E[(\nu e^{-\beta u} + \frac{\sigma e^{-\beta u}}{\sqrt{2\beta}} B(e^{2\beta u} - 1) - \nu e^{-\beta u})(\nu e^{-\beta s} + \frac{\sigma e^{-\beta s}}{\sqrt{2\beta}} B(e^{2\beta s} - 1) - \nu e^{-\beta s})] \\
&= E[\frac{\sigma^2 e^{-\beta(s+u)}}{2\beta} B(e^{2\beta u} - 1)B(e^{2\beta s} - 1)] \\
&= \frac{\sigma^2 e^{-\beta(s+u)}}{2\beta} E[B(e^{2\beta u} - 1)B(e^{2\beta s} - 1)] \\
&= \frac{\sigma^2 e^{-\beta(s+u)}}{2\beta} E[\int_0^u e^{\beta u} dW(u) \int_0^s e^{\beta s} dW(s)] \\
&= \frac{\sigma^2}{2\beta} e^{-\beta(s+u)} (e^{2\beta \min(u,s)} - 1)
\end{aligned} \tag{38}$$

with the last implication due to Itô isometry.

If the Ornstein-Uhlenbeck SDE is formulated as

$$dc(t) = -\alpha c(t)dt + \sqrt{2\beta}dW(t) \tag{39}$$

where $W(t)$ is the standard Brownian motion, then

Lemma 2. *Ornstein-Uhlenbeck process has an explicit solution*

$$c(t+s) = e^{-\alpha s}c(t) + \sqrt{2\beta} \int_t^{t+s} e^{\alpha(s-r)} dW(t+r) \tag{40}$$

Proof. Without loss of generality, let $s = 0$ and define $C(t) = \int_0^t c(r) dr$. Integrate (39) and obtain

$$\begin{aligned}
c(t) &= -\alpha \int_0^t c(t) dt + \sqrt{2\beta} \int dW(t) \\
&= -\alpha C(t) + c(0) + \sqrt{2\beta}W(t_n)
\end{aligned} \tag{41}$$

where in the last implication we made use of Itô stochastic integral:

$$\int_{t_0}^t G(t')dW(t') = \lim_{n \rightarrow \infty} \left(\sum_{i=1}^n G(t_{i-1})(W(t_i) - W(t_{i-1})) \right) \tag{42}$$

Multiply both sides by $e^{\alpha t}$:

$$e^{\alpha t}c(t) + \alpha e^{\alpha t}C(t) = e^{\alpha t}c(0) + \sqrt{2\beta}e^{\alpha t}W(t) \tag{43}$$

Integrate again and multiply by α :

$$\alpha e^{\alpha t} C(t) = c(0)(e^{\alpha t} - 1) + \alpha \sqrt{2\beta} \int_0^t e^{\alpha r} W(r) dr \quad (44)$$

Subtract (44) from (43):

$$e^{\alpha t} c(t) = e^{\alpha t} c(0) + \sqrt{2\beta} (e^{\alpha t} W(t) - \alpha \int_0^t e^{\alpha r} W(r) dr)$$

and note from Itô formula that

$$e^{\alpha t} W(t) - \alpha \int_0^t e^{\alpha r} W(r) dr = \int_0^t e^{\alpha r} dW(r) \quad (45)$$

Substitute this into the previous equation and obtain

$$e^{\alpha t} c(t) = e^{\alpha t} c(0) + \sqrt{2\beta} \int_0^t e^{\alpha r} dW(r) \quad \square \quad (46)$$

Appendix C Penalty-Gradient pseudocode

A pseudocode for implementing the Penalty-Gradient method is given below. The function `f` finds the minimum point `z` for the function at every time step `j`, extrapolates the given grid points to the off-grid values `theta` and concatenates them to the currently available grid points in `yy`. Then, an ODE solver (e.g. Runge-Kutta 4) is used to obtain the values of the grid at a next time step. One should also note that, although throughout the paper we considered S^{out} grid points traveling in one direction, in actual computation we have to consider waves traveling in both directions away from the source. Therefore, in the code we distinguish between the left-going waves (`ltheta`) and right-going waves (`rtheta`) as well as the minimum points at left and right boundaries - `z1` and `zr`, respectively. W , A , and b have been defined in Section 3.4, which also includes the Gradient-Penalty method run inside the function `f`.

```
Initialize yy
```

```
yy = ode_rk4(f,yy)
```

```
function ode_rk4(f,yy):
```

```
    Initialize z1, zr
```

```
    Compute W
```

```
    Compute A, b
```

```
    for i = 1 To m
```

```
        yy[i+1] = yy[i]
```

```
        for j = 1 To n
```

```
            k1 = f(yy[i+1], z1, zr)
```

```
            k2 = f(yy[i+1] + k1*timestep/2, z1, zr)
```

```
            k3 = f(yy[i+1] + k2*timestep/2, z1, zr)
```

```
            k4 = f(yy[i+1] + k3*timestep, z1, zr)
```

```
            yy[i+1] = yy[i+1] + (k1 + 2*k2 + 3*k3 + k4) * timestep/6
```

```
        store z1, zr for next run
    return yy

function f(yy, z1, zr):
    Use Gradient to update z1, zr
    Extrapolate ltheta, rtheta
    yy = concatenate (ltheta, yy, rtheta)
    return yy
```

References

- [1] Bühler O., Oliver M. *Transparent Boundary Conditions as Dissipative Subgrid Closures for the Spectral Representation of Scalar Advection by Shear Flows* Working paper, 2007
- [2] Björck A. *Least Squares Methods: Handbook of Numerical Analysis Vol. 1 Solution of Equations in \mathbb{R}^N* . Elsevier North Holland, 1988
- [3] Colonius T., Rowley C. W. *Discretely nonreflecting boundary conditions for linear hyperbolic systems*. 2000
- [4] Eckhardt B., Faisst H. *Traveling Waves in Pipe Flow*. American Physical Society, 2003
- [5] Gardiner C. W. *Handbook of Stochastic Methods for Physics, Chemistry and Natural Sciences*. Springer-Verlag, Heidelberg, 2002
- [6] Givoli D., Keller J. B. *Exact non-reflecting boundary conditions*. 1989
- [7] Golub G., Loan C. V. *Matrix Computations*. The John Hopkins University Press, Baltimore and London, 1989
- [8] Hanson R., Lawson C. *Solving Least Squares Problems*. Prentice-Hall, Englewood Cliffs, New York, 1974
- [9] Karlin S., Taylor H. M. *An introduction to stochastic modelling*. Academic Press, USA, 1998
- [10] Krylov N. V. *Introduction to the Theory of Random Processes*. American Mathematical Society, USA, 2002
- [11] Li G. *Traveling-Wave Patterns in Binary Mixture Convection with Through Flow*. 2001
- [12] Ropchan C. B., Swaters G. E. *The Role of Negative Energy Waves in Linear and Nonlinear Shear Flow Instability in Hyperelastic Fluid-filled Tubes*. 1992
- [13] Ross S. M. *Stochastic processes*. John Wiley & Sons, USA, 1996
- [14] Quarteroni A., Sacco R., Saleri F. *Numerical Mathematics*. Springer-Verlag, New York, 2000
- [15] Trefethen L. N. *Group velocity in finite difference schemes*. 1982