Society for
Mathematical
Biology

EDUCATION

Check for
updates

VisualPDE: Rapid Interactive Simulations of Partial Differential Equations

Benjamin J. Walker[2,3] · Adam K. Townsend[1] · Alexander K. Chudasama[1] ·
Andrew L. Krause[1]

Abstract

Computing has revolutionised the study of complex nonlinear systems, both by allowing us to solve previously intractable models and through the ability to visualise solutions in different ways. Using ubiquitous computing infrastructure, we provide a means to go one step further in using computers to understand complex models through instantaneous and interactive exploration. This ubiquitous infrastructure has enormous potential in education, outreach and research. Here, we present VisualPDE, an online, interactive solver for a broad class of 1D and 2D partial differential equation (PDE) systems. Abstract dynamical systems concepts such as symmetry-breaking instabilities, subcritical bifurcations and the role of initial data in multistable nonlinear models become much more intuitive when you can play with these models yourself, and immediately answer questions about how the system responds to changes in parameters, initial conditions, boundary conditions or even spatiotemporal forcing. Importantly, VisualPDE is freely available, open source and highly customisable. We give several examples in teaching, research and knowledge exchange, providing high-level discussions of how it may be employed in different settings. This includes designing web-based course materials structured around interactive simulations, or easily crafting specific simulations that can be shared with students or collaborators via a simple URL. We envisage VisualPDE becoming an invaluable resource for teaching and research in mathematical biology and beyond. We also hope that it inspires other efforts to make mathematics more interactive and accessible.

✉ Andrew L. Krause
andrew.krause@durham.ac.uk

1 Department of Mathematical Sciences, Durham University, Upper Mountjoy Campus, Stockton Road, Durham DH1 3LE, UK

2 Department of Mathematical Sciences, University of Bath, Bath BA2 7AY, UK

3 Department of Mathematics, University College London, London WC1E 6BT, UK

✿ Springer

1 Introduction

This paper introduces VisualPDE, a web-based tool for interactive simulations of time-dependent partial differential equations (PDEs) in one and two spatial dimensions. We would highly encourage a reader to browse to https://visualpde.com/ to directly experience this tool, including exhaustive documentation and direct access to all code used. For each of the examples and figures presented in this paper, we have included links to the VisualPDE website in the form of clickable footnotes so that the reader can interact with the content, live in their browser. Rather than provide detailed documentation of VisualPDE's features and examples (which can both be better experienced on the website), this paper provides wider context to the design and anticipated use cases of this tool. Our discussion here includes aspects of the overall design philosophy, technical achievements, and how we imagine it being used for teaching, research, knowledge exchange, and outreach activities involving partial differential equations.

We first give broad historical, pedagogical, and technical context to this project in Sect. 2. In Sect. 3 we give a high-level overview of the technical frameworks used, with a particular eye towards demonstrating why key design choices were made, and in what ways others can extend this project or develop similar tools using the technologies underlying VisualPDE. Our user interface design is a key aspect of this work, described in Sect. 4 along with ways to share and reuse VisualPDE. In Sect. 5 we give examples of using this tool within our own teaching, research, and outreach activities, suggesting ways it can be incorporated by others in future activities. We encourage readers to peruse these sections independently, and especially to spend time playing with the website, to get an overall feel for the project. Lastly, in Sect. 6, we summarise VisualPDE and our vision for how it may enhance how we communicate and interact with PDEs from diverse areas of mathematics.

The website is intended to be living and constantly updated with new features and examples. As a result, figures and snapshots in this manuscript may not always reflect the precise content of the site, but should nevertheless remain illustrative of VisualPDE, its scope, and the authors' aims and ideas. Similarly, pages linked to via the URLs provided in this manuscript may evolve slightly over time, but we do not envisage this impeding the reader.

2 Historical and Pedagogical Context

Widespread access to computation has had many profound impacts on science, and society more broadly. The development of numerical scientific modelling, and the implementation of models on increasingly sophisticated hardware, has led to enormous advances across science and engineering (Gustafsson 2018). Even relatively advanced and technical mathematical models, such as partial differential equations, have become widely used in numerous fields in part due to the development of numerical methods (Thomée 2001) and to the development of high-performance and personal computing machines. Computing has vastly increased the sophistication of modelling, enlarging the kinds of systems that we can understand through simulation that are not amenable to pen-and-paper calculations.

In the past few decades, the field of 'ubiquitous computing' has emerged to describe the accessibility of computation in many aspects of life (Meshram et al. 2016), with widespread use of smartphones and tablet computers being the most obvious examples. In contrast to high-performance scientific computing, the emphasis here is not on technical capabilities, but on accessibility and everyday usability. This is readily apparent in the changing landscape of mobile and web computing, where we see an increasing emphasis on user experience (Benyon 2019), in addition to the availability of raw compute power in mobile devices.

Some areas of scientific computing have transitioned to embracing this 'ubiquitous' nature, with increasingly widespread use of online tools by students and educators for computation and visualisation. Importantly, it has also led to the development of more dynamic and interactive learning environments. For example, there is a growing literature on the use of web-based mathematical visualisation software applied to a variety of mathematical concepts. Such interactive and accessible tools aide students in being able to explore ideas on their own, including designing their own exercises and solutions, which has been shown to have substantial pedagogical benefits (Korucu and Cakir 2018). Examples of such tools include GeoGebra (Sangwin 2007; Arbain and Shukor 2015), Desmos (Ebert 2014; King 2017), and WolframAlpha (Dimiceli et al. 2010; Necesal and Pospıšil 2012). Related to this are more computationally-focused web tools such as CodeRunner (Lobb and Harlow 2016) for rapid unit testing of student-written programs, as well as the growing use of Jupyter Notebooks (Cardoso et al. 2019) for exploring a variety of different areas of scientific computing in interactive ways. Jupyter Notebooks have even been developed to explore areas of computational fluid dynamics (Castilla and Peña 2023) and reaction–transport processes (Golman 2019), showcasing the versatility of these web-based tools even for complex scientific computing tasks such as solving partial differential equations. We direct the interested reader to the work of Engelbrecht et al. (2020) for a broad overview on the role of internet technologies in transforming educational environments.

A major technical roadmark that enabled more immersive web-based tools was the transition from early web multimedia (e.g. Flash and Java applets) to more browser-native HTML5 'canvas' elements (Fulton and Fulton 2013) and related technologies. An example of such a technology-enabled website is 'Complexity Explorables' (Brockmann 2023), a collection of interactive simulations covering a wide range of topics in cellular automata, complex network theory, and beyond. Another advance in this area is the release of a range of graphics-oriented libraries that enable high level, platform agnostic development of interactive web-based applications that make use of graphics processing units (GPUs) to do large-scale calculations in real time (Angel and Shreiner 2014), typically targeting 2D and 3D graphics. Libraries based on the popular WebGL framework include Three.js (Dirksen 2013), Babylon.js (Catuhe et al. 2014) and Abubu.js (Kaboudian et al. 2019a), which each allow for high-level interaction with GPUs within the context of a webpage and have been used in a variety of physics education scenarios (McCauley 2017; Zatarain-Cabada et al. 2023). Abubu.js in particular has been used to develop a range of mathematical visualisations, with a particular focus on models of cardiac electrophysiology (Kaboudian et al. 2021, 2019b). One can find a huge range of examples of this technology online by simply searching for 'WebGL physics' and 'WebGL fluid simulation'.

Such tools have enabled crucial changes in the landscape of higher education instruction, particularly in the context of the life sciences. We note in particular the recent effort in teaching dynamical systems modelling to biology students by Garfinkel et al. (2022) through the development of a new course 'Calculus for Life Sciences', based on dynamical systems. This course makes extensive use of Python/Jupyter Notebooks, allowing for an accessible and interactive approach to student-driven instruction. As described by Garfinkel et al. (2022), there has been a growing need to modernise many aspects of undergraduate teaching, particularly in light of the rapid pace of technological and scientific advancement. This is especially true in the life sciences, where the disconnect with the more 'classical mechanics' training of the mathematical, physical and engineering sciences has grown with the development of these fields (Woodin et al. 2010). Teaching across and encouraging interaction between the biological sciences and mathematical modelling also poses unique challenges in terms of subject matter and cultural differences within each field (Reed 2004). Jupyter Notebooks and other high-level programming interfaces are an important aid to overcoming these barriers, though we feel that even more can be done to connect modelling work in the mathematical and physical sciences more directly with everyday experiences in the life sciences.

Interactive and accessible simulations can play an important role in bridging this divide by providing immediate access to application-relevant simulations, without the need to become experts in underlying foundational aspects (e.g. modelling, numerical analysis and especially programming). In particular, we believe that there would be significant merit to a platform that abstracts away the intricacies of numerical methods and their implementation, with a user thereby free to play directly with concepts and models that are typically only accessible through simulation. Of course, depending on the goal of the learner, such a platform could also be used as a way to motivate learning about the more foundational topics rather than completely circumventing them. From this perspective, accessible and interactive simulations can provide a crucial tool for people to gain an understanding of more advanced topics without the need to build an expert knowledge of this from scratch. These ideas have already been pioneered in teaching coding skills by using 'unplugged programming' methods that do not employ computers or traditional code (Sun et al. 2021; Munasinghe et al. 2023).

VisualPDE aims to be such a platform, representing a significant advancement on the state-of-the-art of visualising PDE systems, especially those of relevance to modelling in mathematical biology and related fields. Existing solutions described above (e.g. Jupyter Notebooks or Mathematica) either still look and feel like programming, or they have a more point-and-click interface but can only handle a limited class of problems. In contrast, VisualPDE uses the extensible interactive design language of websites like Desmos, and applies this to PDE visualisation. It does this with no requirement for familiarity with numerical analysis or programming and, in addition, allows for an unprecedented range of freedom in terms of PDE systems, boundary and initial conditions, and other complex modelling and visualisation features. In addition, it is open source and by default coupled with a tutorial library that serves as a guide for any user looking for further instruction. In short, VisualPDE's accessible interface bridges the gap between the online software many students are (increasingly) familiar

with, and more powerful methods of visualising solutions that typically have a higher barrier for entry.

In the next two sections, we will describe aspects of the implementation and design of VisualPDE, informed by the context described above. We will return to questions of using the software in education, research, and knowledge exchange in Sect. 5, demonstrating the impact that VisualPDE can have in these arenas.

3 Implementing VisualPDE

VisualPDE is designed to be a flexible, plug-and-play PDE solver that runs in a web browser on a user's device. In this section, we give an overarching description of the equations that VisualPDE can solve, the numerical methods that underlie this and the aspects of the implementation that enable this to happen rapidly and interactively on widely available computing devices, including mobile phones.

3.1 The PDEs

When first designing VisualPDE, we were motivated by reaction–diffusion systems of the form

$$\frac{\partial u}{\partial t} = \nabla \cdot (D_u \nabla u) + f_u, \tag{1a}$$

$$\frac{\partial v}{\partial t} = \nabla \cdot (D_v \nabla v) + f_v, \tag{1b}$$

where u, v are scalar fields defined on a domain $\Omega \subset \mathbb{R}^2$, D_u, D_v are given diffusion coefficients (often constants in classical applications) and f_u, f_v are given functions of u and v. An example of such a system is the Gray–Scott model[1] (Gray and Scott 1984). This model received huge interest from scientists and artistic amateurs alike following numerical experiments by Pearson (1993), which demonstrated a striking range of spatial and spatiotemporal phenomena by changing only two parameters. We note in particular the interactive WebGL simulators (pmneila 2012; Sims 2022), among others, that served as inspiration for VisualPDE. Our initial goal was to allow the user to type in values of the diffusion coefficients and kinetics and, hence, explore a larger class of reaction–diffusion systems, rather than hand code the WebGL as in these cited examples.

From this simple yet dynamically rich beginning, VisualPDE has been significantly extended. Currently, coupled systems or subsystems of four unknowns (u, v, w, q) of the following general form can be posed in VisualPDE:

---

[1] https://visualpde.com/nonlinear-physics/gray-scott.html

$$\frac{\partial u}{\partial t} = \nabla \cdot (D_{uu}\nabla u + D_{uv}\nabla v + D_{uw}\nabla w + D_{uq}\nabla q) + f_u,$$

One of $\begin{cases} \dfrac{\partial v}{\partial t} = \nabla \cdot (D_{vu}\nabla u + D_{vv}\nabla v + D_{vw}\nabla w + D_{vq}\nabla q) + f_v, \\ v = \nabla \cdot (D_{vu}\nabla u + D_{vw}\nabla w + D_{vq}\nabla q) + f_v, \end{cases}$

One of $\begin{cases} \dfrac{\partial w}{\partial t} = \nabla \cdot (D_{wu}\nabla u + D_{wv}\nabla v + D_{ww}\nabla w + D_{wq}\nabla q) + f_w, \\ w = \nabla \cdot (D_{wu}\nabla u + D_{wv}\nabla v + D_{wq}\nabla q) + f_w, \end{cases}$ $\quad$ (2)

One of $\begin{cases} \dfrac{\partial q}{\partial t} = \nabla \cdot (D_{qu}\nabla u + D_{qv}\nabla v + D_{qw}\nabla w + D_{qq}\nabla q) + f_q, \\ q = \nabla \cdot (D_{qu}\nabla u + D_{qv}\nabla v + D_{qw}\nabla w) + f_q. \end{cases}$

Each of the variables v, w and q can either satisfy their own time-dependent PDE or can be specified directly in terms of the other variables. The elements D_{ij} form a matrix of diffusivities, and the f_i are given functions. Moreover, f_i and D_{ij} can be a function of the unknowns, the coordinates of the spatial domain, time, and any user-defined parameters. The functions f_i can also depend on first and second spatial derivatives of $\boldsymbol{u}$ (mixed spatial derivatives are not currently supported). This flexibility in form entails that VisualPDE is able to solve a broad class of differential–algebraic equations, including systems with nonlinear cross diffusion and advection. By exploiting zeros and algebraic variables, one can construct systems with up to 8th order spatial or 4th order temporal derivatives. Many intricate, highly nonlinear PDEs can be cast in this general form. Examples on VisualPDE include the inhomogeneous wave equation[2]

$$\frac{\partial^2 u}{\partial t^2} = \nabla \cdot (f(x,y)\nabla u) \iff \begin{cases} \dfrac{\partial u}{\partial t} = v, \\ \dfrac{\partial v}{\partial t} = \nabla \cdot (f(x,y)\nabla u), \end{cases}$$ $\quad$ (3)

and Keller–Segel chemotaxis models[3] (Horstmann 2003)

$$\frac{\partial u}{\partial t} = \nabla^2 u - \nabla \cdot (\chi(u)\nabla v) + f_u(u),$$
$$\frac{\partial v}{\partial t} = D\nabla^2 v + f_v(u,v).$$ $\quad$ (4)

We remark that the user interface (UI) allows a user to select how many equations to solve, whether or not any of the cross diffusion terms (D_{ij} for $i \neq j$) appear, and how many algebraic variables there are. In these cases the notation simplifies to represent the simpler versions of the system, including a single subscript in the case without cross diffusion. Similarly, a user may relabel any of the unknown variables to suit their preferred notation.

---

[2] https://visualpde.com/basic-pdes/inhomogeneous-wave-equation.html

[3] https://visualpde.com/mathematical-biology/keller-segel.html

3.2 Numerical Methods

To solve equations cast in the general form of Eq. (2), VisualPDE employs a central finite difference scheme for all spatial derivatives (as well as some non-centred discretisations to accommodate 'upwinding' methods) coupled to one of several explicit timestepping schemes. This non-specialised approach reflects the intended generality of VisualPDE and facilitates its plug-and-play features. A natural caveat of this generality is that some features of special systems (such as those with conserved quantities) may not be captured as well as might be achieved with bespoke numerical methods, such as symplectic or geometric integrators for models with symmetries such as Hamiltonian systems (Hairer et al. 2006). Nevertheless, we have found that the implemented scheme successfully captures key solution behaviours for a diverse range of systems currently implemented on the site, including many sensible approximations to systems with infinitely many conserved quantities. A nontrivial example is the phenomenon of solitons passing through one another in the Korteweg–De Vries equation[4] (Miura 1976),

$$\frac{\partial \phi}{\partial t} = -\frac{\partial^3 \phi}{\partial x^3} - 6\phi \frac{\partial \phi}{\partial x} \iff \begin{cases} \frac{\partial \phi}{\partial t} = -\frac{\partial^2 v}{\partial x^2} - 6v\phi, \\ v = \frac{\partial \phi}{\partial x}. \end{cases} \tag{5}$$

The use of explicit, non-symplectic timestepping means that this scheme will not preserve any of the infinitely many conserved quantities of this model and, hence, will exhibit small fluctuations as it is only an approximation to a 'true' soliton–soliton solution. Nevertheless, after passing through one another, these solitons have visually identical heights and speeds in VisualPDE, indicating a good approximation of the behaviour of this model.

The remainder of this subsection is a discussion of well-known standard finite difference methods, which we include for completeness. See LeVeque (2007) or most other texts on the numerical analysis of partial differential equations for a more complete overview.

3.2.1 Spatial Discretisation

In more detail, suppose that a user has taken the domain to be rectangular (the default for the majority of examples present on the site), with coordinates (x, y) in $\Omega = [0, L_x] \times [0, L_y]$ for $L_x, L_y > 0$. For a given spatial step size Δx, configurable by the user, we split the domain into an $(L_x/\Delta x, L_y/\Delta x)$ grid (rounding down where necessary). Spatial derivatives are computed on this grid using commonplace finite difference schemes, with first derivatives being computed using the central difference:

$$\left. \frac{\partial u}{\partial x} \right|_{(t,x,y)} \approx \frac{u(t, x + \Delta x, y) - u(t, x - \Delta x, y)}{2\,\Delta x}. \tag{6}$$

---

[4] https://visualpde.com/nonlinear-physics/kdv.html

The divergence terms of Eq. (2) are computed by approximating

$$
\begin{aligned}
\nabla \cdot (D_u \nabla u)|_{(t,x,y)} \\
&\approx \frac{1}{2\,\Delta x^2}\Big\{ D_u(t,x,y)[u(t,x-\Delta x,y) - 2u(t,x,y) + u(t,x+\Delta x,y)] \\
&\quad + D_u(t,x-\Delta x,y)[u(t,x-\Delta x,y) - u(t,x,y)] \\
&\quad + D_u(t,x+\Delta x,y)[u(t,x+\Delta x,y) - u(t,x,y)]\Big\} \\
&\quad + \frac{1}{2\,\Delta y^2}\Big\{ D_u(t,x,y)[u(t,x,y-\Delta y) - 2u(t,x,y) + u(t,x,y+\Delta y)] \\
&\quad + D_u(t,x,y-\Delta y)[u(t,x,y-\Delta y) - u(t,x,y)] \\
&\quad + D_u(t,x,y+\Delta y)[u(t,x,y+\Delta y) - u(t,x,y)]\Big\}.
\end{aligned}
\tag{7}
$$

Here, D_u can depend on space (explicitly or implicitly as a function of the variables $\boldsymbol{u}$), though the approximation reduces to a standard second-order central difference approximation if D_u is independent of space. At boundaries, ghost nodes are introduced to allow for the enforcement of boundary conditions through modification of the finite difference schemes. VisualPDE accommodates periodic, Neumann and Robin boundary conditions in this way. Dirichlet boundary conditions are also supported, though these do not modify the finite difference operators and are instead implemented by fixing the values at these nodes. Boundary conditions can be fully customised by the user via the VisualPDE interface, including mixed conditions on different interfaces, and arbitrary inhomogeneous boundary terms (e.g. time dependent boundary conditions are obtained just by writing some function of t when specifying the boundary condition).

Domains that are not rectangular are implemented by casting them as subsets of a rectangular domain, with individual points in the finite difference discretisation included or excluded via an indicator function. As this enables users to specify general domains that need not have smooth boundaries, boundary conditions on non-square domains involving derivatives should be interpreted with due care.

3.2.2 Timestepping

With space discretised as above, the resulting system of ordinary differential equations are solved by discretising in time and employing one of four explicit, fixed-timestep finite difference schemes: forward Euler, two-step Adams–Bashforth, the midpoint method (also called the modified Euler method) and the four-step Runge–Kutta method (often known as RK4). These schemes are each associated with benefits and drawbacks, with RK4 conferring the broadest stability, the highest order accuracy and the greatest computational cost, while forward Euler represents the computationally simplest option but scales least favourably with the timestep Δt.

Forward Euler is the default option for the majority of the examples in VisualPDE due to its simplicity and somewhat surprising reliability in practice. For completeness, this scheme approximates

$$\left.\frac{\partial u}{\partial t}\right|_{(t,x,y)} \approx \frac{u(t + \Delta t, x, y) - u(t, x, y)}{\Delta t}, \tag{8}$$

which is first-order accurate in Δt. At any point a user can select from any of the timestepping schemes available, including midway through a simulation, which might be needed to satisfy a user's requirements of accuracy, stability or curiosity. In this way, VisualPDE enables the simple, accessible exploration of the properties of numerical schemes in practice.

3.2.3 Validation

We have validated the results from VisualPDE through a number of comparisons to published results, as well as our own finite difference, spectral, and finite element implementations. We have also included a 'Numerical Methods' section on the website, which we plan to expand with further examples discussing aspects of the methods used. Currently this section consists of an example[5] that explores three equations with analytical solutions that can be quantitatively compared to numerical solutions. Importantly, this also invites the reader to explore changing the timestep or timestepping scheme, in particular in the context of the Schrödinger equation where a total mass of the wavefunction is only approximately conserved.

3.3 WebGL Implementation and Technical Considerations

Solving the large system of ordinary differential equations (ODEs) that arises from the spatial discretisation and displaying their solution presents non-trivial computational challenges. This is compounded by the desire for VisualPDE to run in browsers across a broad range of devices, including mobile phones, with a high level of interactivity and speed. To overcome this challenge, VisualPDE exploits the significant and widespread capabilities of graphics processing units (GPUs), present in some form on essentially all modern computing devices.

In essence, VisualPDE achieves this by casting the problem of timestepping ODEs as one of iterated image manipulation. Floating-point image textures are used to represent the computational domain, whereby each pixel corresponds to a point in the spatial discretisation. Taking advantage of the massively distributed parallel architecture of modern GPUs, the points in the spatial domain are updated in parallel every timestep. The speed of this image-based approach typically entails that VisualPDE can advance the solution a great many times each second, with recent devices (including mobile phones) being capable of upwards of 24,000 timesteps per second in representative simulations. This amounts to performing hundreds of timesteps every time the user's

---

[5] https://visualpde.com/numerical-methods/validating-VisualPDE.html

device requests an update to the solution being displayed on the screen, which typically occurs 30–60 times each second. We refer directly to the GitHub code (Walker et al. 2023) for further details on the implementation, which makes use of the Three.js library for interfacing with WebGL.

The parallelism that drives the responsiveness and speed of VisualPDE does, however, come with its own limitations. One significant limitation is that the independent updating of each pixel prevents the simple, efficient implementation of implicit timestepping schemes, which typically update every point at once as the solution of a coupled system of equations. This is the reason why VisualPDE only makes use of explicit timestepping schemes, though we note that the conditional numerical stability associated with these schemes is largely mitigated by the frequency at which timesteps can be taken, with small Δt thereby not detracting from the user experience.

Another barrier posed by the use of GPUs is the lack of widespread support for double-precision floating point textures. While these are available on some devices, they do not appear to be ubiquitously supported. Thus, VisualPDE makes exclusive use of single-precision arithmetic to ensure a consistent experience across devices, though we expect to transition to double-precision arithmetic as support widens. Simulations using the website do remarkably well in quantitatively matching more accurate numerical methods using standard double-precision methods, though we would advise some caution if an example demands particularly high accuracy and precision.

Providing a consistent experience for all users has presented a number of additional challenges, not least of which are vendor-specific differences in the capabilities and behaviours of various devices. For instance, many mobile devices do not automatically interpolate low-resolution textures onto the display, while many laptop and desktop computers do. To circumvent this, VisualPDE attempts to detect your device and its capabilities and implements bespoke interpolation where necessary. Despite the varied idiosyncrasies of popular devices, extensive testing suggests that the functionality of VisualPDE is maintained across many kinds of devices, with the curated examples configured to provide a smooth, interactive experience on even relatively low-end hardware.

4 Interacting with VisualPDE

4.1 Design of the User Interface

The VisualPDE user interface (UI) consists of two halves: a static website, with colourful tutorials, examples and user guides, and the interactive simulation page, where PDEs are solved in real time inside the browser window. Our guiding philosophy for both is to let intuition guide the user: to be minimal yet feature rich.

The website is written using the static site generator Jekyll (Preston-Werner 2023), which compiles HTML files from simple text-based markdown scripts. Combined with MathJax (Cervone 2012) to enable LaTeX-style mathematical typesetting, it allows pages to be written by authors without technical HTML or CSS knowledge, scaling gracefully to different device sizes.
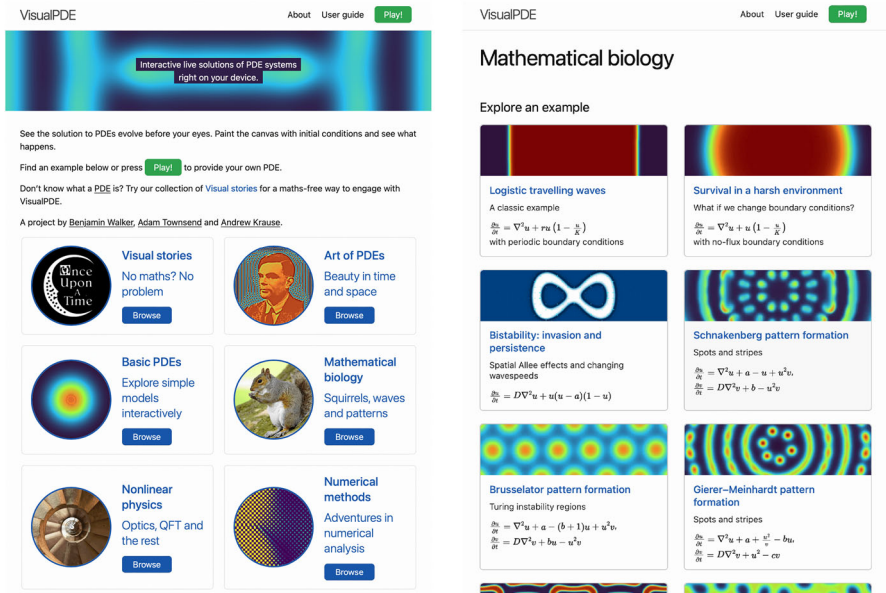
Fig. 1 The VisualPDE static website. **Left**: The homepage presents several collections of examples organised by theme. **Right**: The mathematical biology tutorial collection starts with simple models and becomes increasingly complex. Attractive simulation screenshots indicate the behaviour the user might observe. Captured July 2023

The front page of the website presents links to a number of collections that each contain guided tutorials on a particular PDE system (see Fig. 1) or several related models. These collections start with basic linear PDEs, followed by more complicated systems in mathematical biology, nonlinear physics, and numerical analysis. The curated examples are largely chosen to demonstrate different phenomena that we feel benefit from being explored through interactive visualisations, as well as to illustrate different features of VisualPDE. We also include our innovative curiosity-driven 'Visual Stories', discussed in Sect. 5. Importantly, these collections of examples are not meant to be exhaustive, and features are provided to make crafting and sharing entirely new models seamless and simple. Each tutorial contains links to the simulation page, with preset equations, parameters, and other settings. The user can then change these on the simulation page and see how the system solution changes. Importantly, every example can be transformed into every other example on the website simply by clicking and typing within the interface. This includes changing the colour scheme and other visualisation options under the Views menu at the bottom of the left column of buttons (see Fig. 2e).

The simulation page considers the entire device browser window as the domain on which to solve the PDE. Optimised for both mobile and static devices, the initial user experience is a blank coloured page with icons on the left and right. An invitation to click or tap (device depending) prompts the user to paint the screen using the cursor or their finger to set a forcing or initial condition of the system (Fig. 2a). Painting can
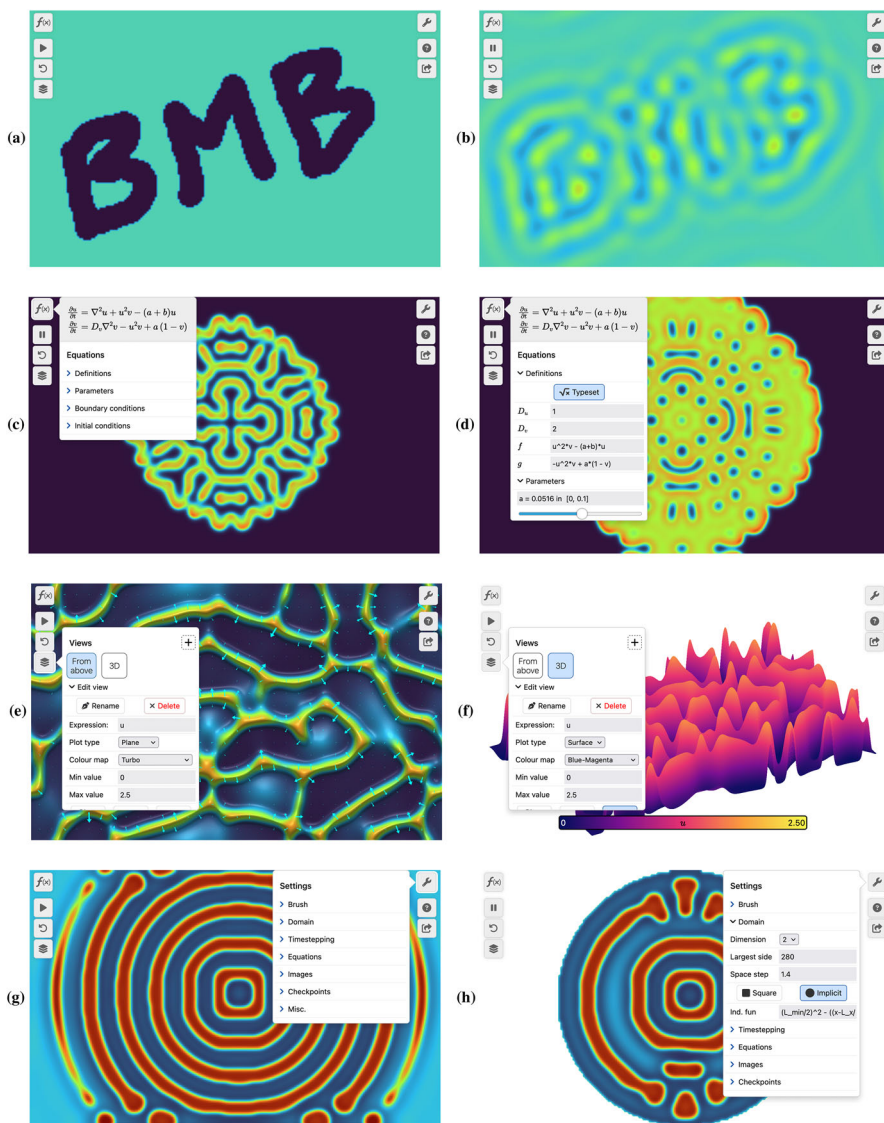
Fig. 2 The VisualPDE simulation page. **a** The user can pause the simulation and paint an initial condition on the screen using the cursor or their finger. **b** By clicking 'play', the simulation starts (here, the Brusselator). **c** The equations panel allows users to edit the equations and parameters being simulated. **d** Editing the definitions and parameters in real time allows the user to see the effect immediately (here, in the Gray–Scott model, a has been increased.) **e** The Views panel allows for selection between (potentially preset) quantities to visualise, as well as providing a selection of colour maps, scales, lighting and other options. Here, we are simulating Keller–Segel chemotaxis with a 3D lighting effect and an overlaid vector field $(-\partial u/\partial x, -\partial u/\partial y)$. Some options in the panel require scrolling down to see. **f** The same solution at the same time as **e** but viewed as a surface plot, with a different colour map and with a colour bar. **g** The settings panel allows for changes in domain shape, timestepping scheme, brush sizes when painting with the cursor/the user's finger, and any images used as input. Here we see FitzHugh–Nagumo on a rectangular domain. **h** The same system as **g** but on a circular domain. Captured July 2023 on a 800×500 px screen (Color figure online)

be done at any time, live, with a configurable brush; using the pause and play icons allows for more complex painting that evolves live when the simulation is resumed (Fig. 2b).

Although the static website links to many preset simulations, the equations are entirely configurable from within the simulation page. The equations panel (Fig. 2c, d) allows for live alteration of all the equations within the form discussed in Sect. 3.1. Expressions are instantly typeset in MathJax both for familiarity and to assure the user that VisualPDE has interpreted their expression correctly. As a user modifies a part of the equations, the relevant typeset portion of the equation is highlighted to guide the user. Sliders for parameters can be created easily by the user, similar to those seen in popular online 2D plotting software, allowing for interactive, intuitive parameter sweeps. The limits of these sliders can be fully customised so that, for example, educators can suggest ranges for students to look within. Boundary conditions for each species can be individually set to periodic, Dirichlet, Neumann, Robin or a specified combination through a drop-down menu. With these features, the UI is intended to make explorations of the impact of boundary conditions, or the presence of bifurcations, intuitive and simple to play with.

VisualPDE presents many options for visualising solutions. The Views panel (Fig. 2e, f) allows users to specify which expression they want to see mapped onto the domain: this can be customised and can include any nonlinear combination of species or even explicit time or space dependence. Although the default for 2D systems is to display the solution as a 2D image, the solution can also be visualised in 3D on a surface, or for 1D models, on a line. An adaptive colour bar can be displayed and customised, with a wide variety of colour maps available to cater for user preference and enhance accessibility (Smith and van der Walt 2015). For even richer visuals, contours and 3D-effect lighting can be added, along with custom, solution-dependent vector fields and graphical overlays. All these viewing options can be saved as preset views and given a custom name (in Fig. 2e, f, 'From above' and '3D'), so that simulations can be shared and understood easily.

Finally, the settings menu (Fig. 2e) contains slightly more advanced options. Numerical options include the dimension, step size and shape of the domain, as well as the timestepping scheme and timestep size. Presentation options include custom choices of letters for the species functions (by default u, v, w and q) to better match any source material; the shape and size of the brush for painting; labels showing the elapsed time and integral of the displayed solution. Checkpoints can be set so the user can restart the simulation to a given timestep, rather than to the beginning of the simulation. A particularly fun option here is to upload a photograph to use as a spatial function inside an equation (Fig. 3). By default, Sofya Kovalevskaya and Alan Turing are functions $I_S(x, y)$ and $I_T(x, y)$, but clicking or tapping on their photos prompts the user to either upload a different photo (on a computer) or to take a photo using their camera (on a phone). We consider this a fun hook into the software.

User feedback regarding error handling proves a particular challenge in presenting a user-friendly interface, especially for users who may be unfamiliar with the underlying numerical methods. VisualPDE provides custom messages for syntax errors, and for solutions blowing up. Numerical solutions (Sect. 3.2) can be fragile, in strong contrast to 2D plotting software that novice users may bring their intuition from. Ideally users
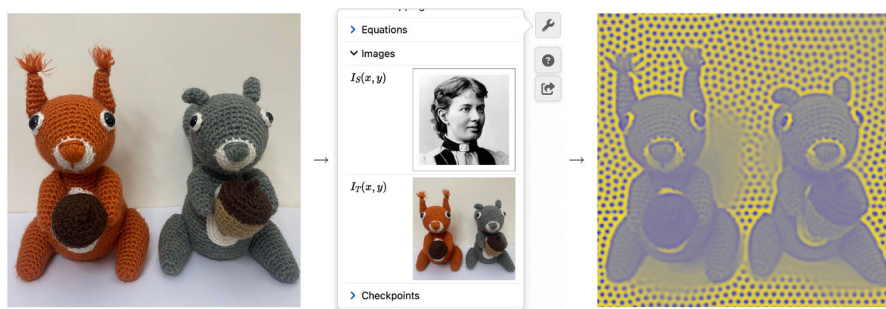
Fig. 3 Turingify your friends: Users can take photos on their phone and use them as forcing in their PDEs. Here, the Turing on Turing (https://visualpde.com/art-pdes/turing-face.html) example on VisualPDE was customised with happily coexisting red and grey squirrels. The user can also select a predefined image such as Sofya Kovalevskaya or Alan Turing (not shown). Original image of squirrels courtesy of Elizabeth Brocklebank (Color figure online)

should be told whether an infinite solution has been reached because of numerical instability or because the solution is truly exponential growth. This is beyond the scope of VisualPDE at the moment, but the omnipresent help icon and the error messages both lead to discussions on numerics that may be helpful, including practical tips for improving numerical stability. Preset simulations where parameters are constrained by sliders and where equations are uneditable (as in the Visual Stories) may therefore be more beneficial to novice users.

4.2 Sharing VisualPDE Models

There are several different ways that users can share and extend the VisualPDE website and simulator, described in detail on the VisualPDE FAQs page.[6] These include sharing a screenshot, sharing a link to a user-crafted model, embedding a model inside another website, or forking the entire VisualPDE website or simulator via the GitHub code (Walker et al. 2023). We expect that most users will design a simulation, either by modifying an existing example or writing their own, and then share this via a direct link. This can be done from within the 'share' menu, accessed by clicking the 'share' icon on the right-hand side of the screen. This will generate a URL that corresponds exactly to the current simulation given the specified initial conditions (it will not include anything drawn with the brush, timesteps taken, or any changes to the image files I_S or I_T). We expect that this functionality meets the requirements of the vast majority of users, including those creating bespoke models for teaching or research.

Individual simulations can be directly embedded within a webpage by clicking 'Embed' within the share menu, which generates a fragment of HTML that can be pasted into a user's own site. The result of this can be seen in action in the 'Visual Stories' collection on the website. One can even customise the level of UI elements which are displayed, depending on how they envision a user interacting with their

---

[6] https://visualpde.com/user-guide/FAQ.html#sharing

model. We can see this being invaluable in designing webpages to describe models in more detail, such as in research pages or interactive lecture notes.

Lastly, the entire code base is shared through a standard CC BY open source licence, meaning that users are allowed to directly fork the GitHub repository and design their own versions of the website, or even the simulator. The website has been designed with this idea of extensibility in mind, as the vast majority of it is written using high-level markdown, so that creating collections of examples like those currently on the site can be done very easily. We imagine this detailed modification being used by educators to design bespoke courses or lecture notes with VisualPDE at their core.

5 VisualPDE in Teaching, Research and Knowledge Exchange

Here, we briefly outline a few current projects and activities involving VisualPDE in different settings, and discuss the possibilities of using it much more widely.

5.1 Teaching Through Interactive Simulations

The systems presented in the basic PDEs section of the website include many of the classical linear PDEs studied in undergraduate courses, such as the heat and wave equations, in addition to relatively simple extensions including the convection–diffusion and Euler buckling models. These examples include demonstrations of the d'Alembert solution of the wave equation (see Fig. 4a, b), and the Fourier series solution of the heat equation. The overall goal is to provide some intuition for mathematical formulas obtained analytically (such as the role of the wavenumber in the decay of a cosine initial condition in the heat equation), as well as to go beyond what can be easily understood analytically, for instance exploring the impact of heterogeneous media on these simple models.
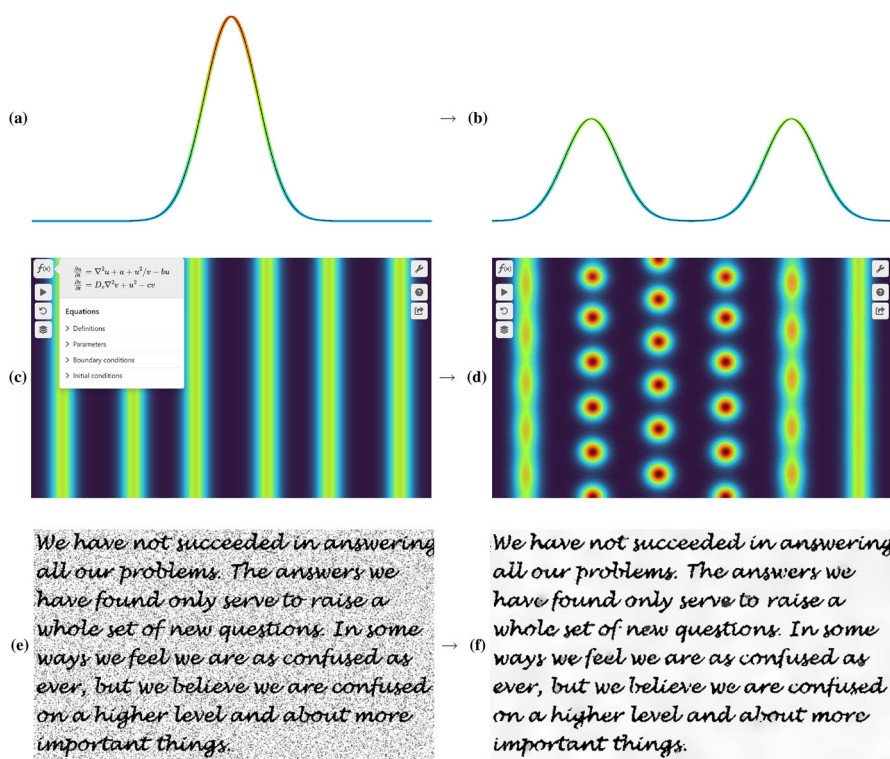
Many of the PDEs that appear in Murray's classical textbooks on mathematical biology are included on the website (Murray 2003). These books form the core of many mathematical biology courses, and cover a range of topics that can benefit from interactive visualisations. Two of the authors of this manuscript have begun making extensive use of VisualPDE in teaching a third year undergraduate mathematical biology course at Durham University, providing links and interactive demonstrations of VisualPDE simulations in lectures and via the course's content management system. Initial informal feedback from students suggests that VisualPDE has significantly enhanced students' understanding of a range of topics in the course. These include travelling waves in the Fisher–Kolmogorov equation,[7] impacts of Allee effects on spatial population invasion,[8] and pattern formation and stripe vs spot selection in the Gierer–Meinhardt system.[9]

Importantly, all of these topics illustrate theory that can be developed and understood with pen and paper, while also highlighting phenomena that are analytically

---

[7] https://visualpde.com/mathematical-biology/travelling-wave.html

[8] https://visualpde.com/mathematical-biology/bistable-travelling-waves.html

[9] https://visualpde.com/mathematical-biology/gierer-meinhardt.html

Fig. 4 Example simulations from teaching. **a** and **b** An example d'Alembert solution (https://visualpde.com/sim/?preset=waveEquation1DValidity) of the wave equation matched against the analytical solution in black. **c** and **d** An example of stripe to spot instabilities (https://visualpde.com/sim/?preset=GiererMeinhardtStripeICs) in the Gierer–Meinhardt system. **e** and **f** Image denoising (https://visualpde.com/sim/?preset=PeronaMalik) via the Perona–Malik equation

intractable at this level but easy to explore numerically. For example, we encourage students to explore how the initial mass and shape of an invasive population matters for persistence when subject to a strong Allee effect, and compare this with a spatially homogeneous model where persistence is much easier to understand due to the simplicity of an equilibrium acting as a separatrix. Figure 4c and d shows an example simulation in Gierer–Meinhardt demonstrating how stripe-like solutions are unstable to small perturbations, breaking up into spots. Such a result is difficult to understand analytically (Kolokolnikov et al. 2006), but one can develop intuition through simulations.

Along with using VisualPDE to enhance engagement and build intuition in a lecture setting, we have been including interactive exercises on homework sheets that encourage students to numerically verify their analytical calculations of, for example, thresholds for Turing instability or wavespeeds for travelling wave models. In future, we hope to further capitalise on the flexibility of VisualPDE and provide students with opportunities to exhibit creative expression in their mathematical explorations, such as by designing their own biologically meaningful systems (Woolley et al. 2021).

In the broader context of undergraduate education, we hope that VisualPDE can be integrated into a wide range of courses and incorporated into a new generation of interactive course assessments.

Beyond introductory PDEs and mathematical biology, colleagues have reported using VisualPDE to give students access to visualisations in other topics. For example, the Korteweg–De Vries equation simulation mentioned in Sect. 3.2 exhibits an important phenomenon, in which two solitons pass through one another without interacting (noting that this is only captured approximately, as described in Sect. 3.2). The Perona–Malik equation,[10] used for image denoising, particularly benefits from interactive exploration, as the ability of the model to sharpen an image can depend somewhat sensitively on the parameters (see Fig. 4e, f for a successful denoising simulation). These are just two examples of what we consider to be important lessons that can often be lost in pen-and-paper analysis or the details of numerical implementations, but which can be immediately appreciated by simply trying it out using different parameters and images in VisualPDE. As noted in Sect. 2, we view such playing at the concept level as valuable scaffolding for in-depth courses in topics such as image analysis and numerical methods.

In the UK context, it is common for lecturers to use bespoke lecture notes for a given course. In the past few years it has become standard to make these notes available to students, and it is becoming increasingly desirable to make them accessible in HTML, which has several advantages compared to PDF, especially for students with disabilities (Mejía et al. 2021). One advantage to web-based tools such as VisualPDE, GeoGebra and Desmos is that they can be natively embedded in such webpages, providing seamless connection with the material being described. There are currently several efforts to develop accessible and interactive lecture notes, and we are hopeful that VisualPDE can bring substantial value to these web-based formats. Of course, there are important challenges regarding accessibility of such visual tools themselves, and this is an ongoing area of current work.

5.2 VisualPDE in Research Development and Communication

This project was initially conceptualised in terms of teaching. However, we think there is ample potential to use it for rapid prototyping and communication of a variety of spatial models, and we are presently using it in our own research. Below we showcase some examples of this where we discuss the ease with which spatial models can be developed and shared, and highlight the potential for VisualPDE to drastically alter how we communicate and interact with mathematics during the research cycle.

5.2.1 Spatially Heterogeneous Reaction–Diffusion Dynamics

Page et al. (2003, 2005) explored models of spatially heterogeneous reaction–diffusion systems finding, among other things, that such heterogeneity can lead to spatiotemporal movement of spike solutions in 1D. This dynamic behaviour was later explored by Krause et al. (2018) and Kolokolnikov and Wei (2018) using different methods. Related

---

[10] https://visualpde.com/nonlinear-physics/perona-malik.html

to these studies, Dillon and Othmer (1999) explored mixed boundary conditions using continuation methods, and Krause et al. (2021) later justified these conditions in terms of asymptotic models of heterogeneous tissue, finding that certain combinations of mixed boundary conditions can isolate patterning away from the boundary of the domain. In all of these cases, the phenomenon was largely understood first by numerical exploration before any analytical results were available. VisualPDE allows for rapid implementations of every model studied in these papers, such as this example of dynamic bifurcations with heterogeneity,[11] including explorations of the dynamic spatiotemporal spike oscillations observed first by Page et al. (2005). While our implementation of these models in VisualPDE is some years after these papers were written, one could imagine developing these ideas substantially more quickly through the rapid prototyping and analysis made possible by VisualPDE.

5.2.2 Localised Structures

Localised patterns are steady states of PDEs with spatial structure only in a subset of the domain, with most of the domain at a homogeneous and stable steady state. They represent an interesting class of multistable solutions in PDEs, and substantial recent research has been undertaken to understand them (Burke and Knobloch 2007; Knobloch et al. 2011; Champneys et al. 2021). Recently, Hill et al. (2023) explored a class of symmetric localised solutions in two spatial dimensions in the Swift–Hohenberg equation. The lead author of the cited study was able to quickly use these results to develop a VisualPDE model capable of generating several classes of two-dimensional solutions exhibiting different symmetries, one of which is shown in Fig. 5b and can be explored on VisualPDE.[12] Once these states are found numerically within VisualPDE, it is easy to subject them to various perturbations, both in terms of perturbing the solution state but also in terms of changing the model. For instance, an example exploring advection[13] considers the effect of rotational and linear advection on these localised states. For small values of advection the states persist with minor changes but, as the advective velocity increases, they begin to change shape and lose symmetries. An example is shown in Fig. 5c–f, where a large rightward advection has changed the shape of the localised structure, with it developing a larger hexagonally-spaced 'tail'. Importantly, the solution no longer retains the D_4 symmetry used to rigorously study its existence and stability. This loss of symmetry immediately leads to questions of how to extend such analyses to consider more complicated settings where the originally assumed symmetries are not preserved in the model and, more broadly, to understand how robust such solutions are to changes in the model. Notably, extending this model within VisualPDE to include advection took only a matter of seconds, significantly less than the time needed to develop a numerical scheme to add advection in a programming language.

---

[11] https://visualpde.com/mathematical-biology/heterogeneous-dynamics.html

[12] https://visualpde.com/nonlinear-physics/swift-hohenberg.html

[13] https://visualpde.com/nonlinear-physics/advecting-patterns.html
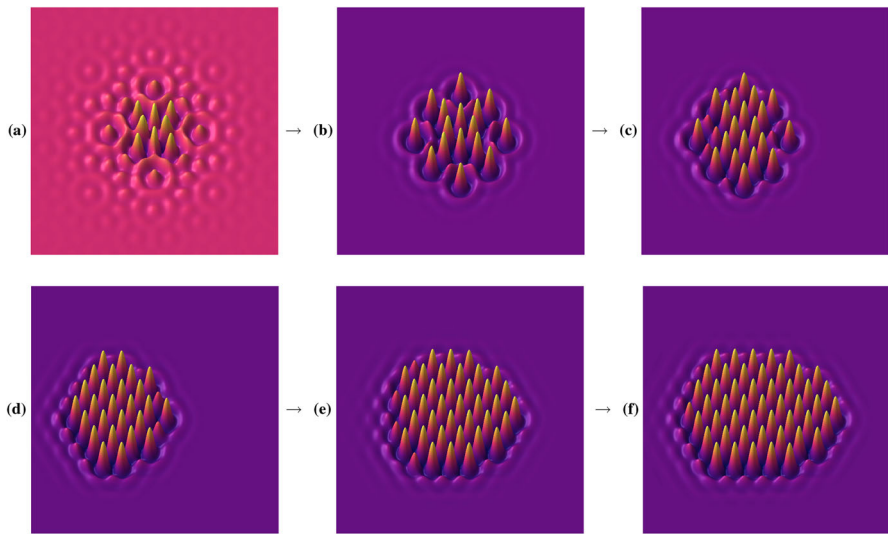
Fig. 5 Example simulations of the Swift–Hohenberg equation giving rise to translating localised solutions (https://visualpde.com/sim/?preset=swiftHohenbergRight), which can be explored with VisualPDE. **a** and **b** Localised patterns in the form from an initial condition in **a** to the stationary structure with D_4 symmetry in **b**. **c** to **f** Under sufficiently large advection to the right, the localised structure from **b** undergoes an instability forming a hexagonal pattern on its left side, breaking the D_4 symmetry. Captured July 2023 on a 1280×768 px screen

5.2.3 Spatiotemporal Dynamics

Another area in which rapid and interactive PDE simulations are extremely valuable is the domain of spatiotemporal solutions, such as solitons, travelling and spiral waves, and chaos. For such systems, 2D snapshots are often insufficient to really gain intuition about the dynamics compared to moving videos. Many such systems are also somewhat sensitive to initial conditions and other details of the model, so being able to tweak these interactively allows for a deeper understanding of the roles of such details in the resulting dynamics. Figure 6 presents some exemplar systems, including invasion in a spatially heterogeneous Fisher–Kolmogorov equation, spiral waves in a three-species 'rock–paper–scissors' Lotka–Volterra system (Reichenbach et al. 2007), spatiotemporal chaos in the Kuramoto–Sivashinsky equation (Kalogirou et al. 2015), coarsening in the Cahn–Hilliard equation (Miranville 2019), Turing wave bifurcations in hyperbolic reaction–diffusion systems (Ritchie et al. 2022), and irregular vegetation patterns in the Klausmeier model (Klausmeier 1999). These examples have many connections to contemporary research questions where we feel the lightning-fast interactivity of VisualPDE can be helpful in stimulating discussion and understanding broad qualitative features of different models.
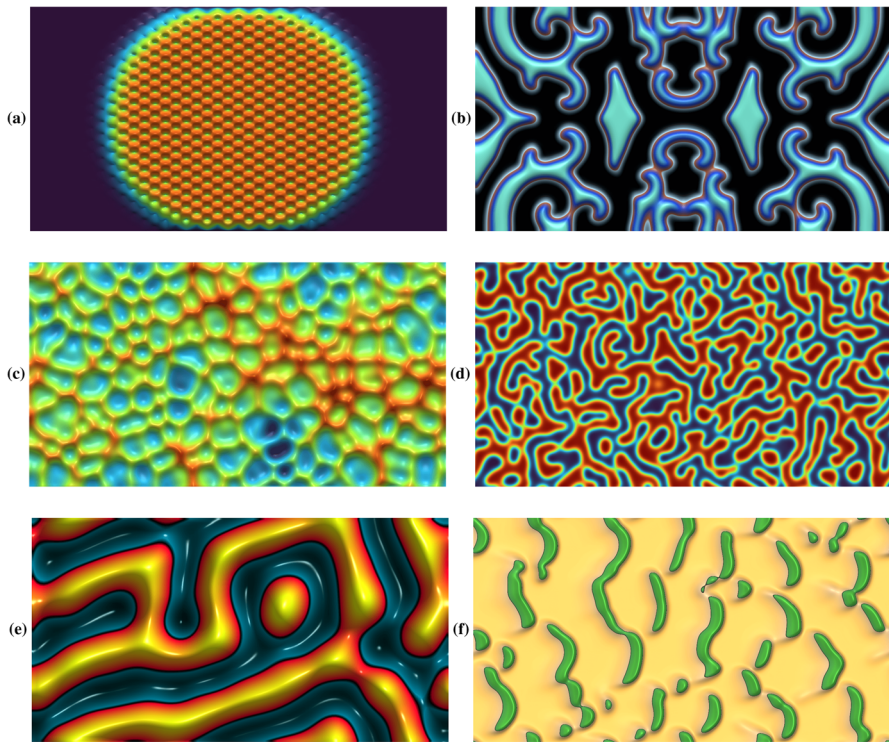
Fig. 6 Example simulations of spatiotemporal solutions. **a** A heterogeneous Fisher–Kolmogorov equation (https://visualpde.com/sim/?preset=InhomogeneousFisherKPP). **b** A three-species Lotka–Volterra system (https://visualpde.com/sim/?preset=cyclicCompetition). **c** The Kuramoto–Sivashinsky equation (https://visualpde.com/sim/?preset=KuramotoSivashinsky). **d** An example of coarsening in the Cahn–Hilliard equation (https://visualpde.com/sim/?preset=CahnHilliard). We recommend playing with the timescale r in this simulation. **e** Turing wave bifurcations (https://visualpde.com/sim/?preset=TuringWaveFHN2D) in a hyperbolic reaction–diffusion system. **f** Irregular vegetation patterns in the Klausmeier model (https://visualpde.com/sim/?preset=KlausmeierModel). Captured July 2023 on a 1280×768 px screen

5.2.4 Research Communication

Besides the examples above, the authors have also found the use of VisualPDE in interacting with collaborators, colleagues and industrial partners invaluable. Being able to rapidly test an idea and share a link directly to a simulation (with all parameters and functional forms easily accessible) has been greatly helpful in speeding up our own research efforts. A use case that we envisage becoming more and more common is for demonstrating prototypes to industrial and interdisciplinary partners, with VisualPDE enabling solution and visualisation within minutes of a model first being conceived. Even in reviewing papers, we have found the ability to quickly corroborate qualitative results useful. We suspect that as tools like VisualPDE become more widespread, it will become much more routine to send direct links to open-source implementations of models at high levels. We also hope that, as in this article, it will become commonplace
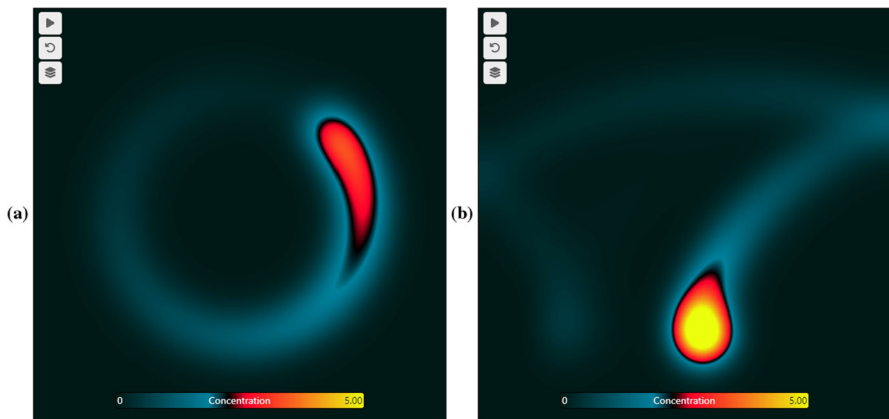
Fig. 7 Example simulations of the concentration of virus particles from a rotating source, from an adaptation of the model of Lau et al. (2022). **a** A circling source only. **b** A circling source subject to advection to the right due to an air conditioner. See the Visual Story[27] for more details

to include links to interactive simulations in research articles, allowing the research community to engage with published science in an entirely new way.

5.3 Knowledge Exchange Through 'Visual Stories'

Beyond teaching or research, we think VisualPDE has marked potential for broad knowledge exchange and outreach activities. We have begun developing a 'maths-free' collection of examples that focuses on models as ways to explore phenomena, doing so purely through interactive simulation with VisualPDE and a guiding narrative aimed at the general public. These can be found in the 'Visual Stories' collection[14] on the website.

One example of the use of VisualPDE for knowledge exchange is in exploring airborne virus transmission within a room, incorporating the effects of circulating airflow. Lau et al. (2022) developed a model of this situation, which eventually led to significant interaction with policymakers during the Covid-19 pandemic and the development of a web-based airborne virus risk calculator.[15] With support of the authors of the cited study, we extended this interface using VisualPDE to allow for more detailed spatial probabilities of infection in a Visual Story on virus transmission. This innovative medium complements the existing calculator by providing an interactive way for diverse audiences to engage with the various features of the model and develop their own intuition for complex mathematics. See Fig. 7 for some snapshots of this Story, or interact with the Story[16] yourself.

Notably, creating this accessible exposition of complex, cutting-edge mathematics was straightforward using VisualPDE. In particular, producing visually striking sim-

---

[14] https://visualpde.com/visual-stories.html

[15] https://people.maths.ox.ac.uk/griffit4/Airborne_Transmission/index.html

[16] https://visualpde.com/visual-stories/airborne-infections.html

ulations that are woven into the narrative is simple, with interactive elements being embedded directly into the article. We hope that this inspires other researchers to explore this medium as a way to communicate their research and knowledge to a broad audience, with engagement and interaction at the heart of the discourse.

6 Summary

In this paper, we have presented a web-based interactive PDE simulator capable of solving a large range of time-dependent PDEs in real time. We have discussed the historical and pedagogical context behind this tool, its technical and user-facing design philosophy, and examples of its use in teaching, research, and knowledge exchange, which we hope illustrate its broad, multi-audience potential. Looking ahead, we hope that VisualPDE proves to be useful across many mathematical communities, and that it facilitates and inspires interdisciplinary connections through interactive, accessible, shareable computing. We also hope that, alongside tailored numerical codes, future research articles across disciplines might include links to interactive, representative simulations like those presented in this article, engaging audiences in what we believe is an entirely new way. We are excited to continue developing VisualPDE and exploring its potential for changing how we interact with and communicate science and mathematics. We are hopeful that VisualPDE inspires further efforts in making mathematics more interactive, which we see as an exciting and important frontier in how we communicate and conceptualise science.

Data Availability There is no data present in this paper. All code associated with the project can be found on GitHub (Walker et al. 2023).

References

Angel E, Shreiner D (2014) Interactive computer graphics with WebGL. Addison-Wesley Professional
Arbain N, Shukor NA (2015) The effects of GeoGebra on students achievement. Proc Soc Behav Sci 172:208–214
Benyon D (2019) Designing user experience. Pearson
Brockmann D (2023) Complexity explorables. https://www.complexity-explorables.org/

Burke J, Knobloch E (2007) Homoclinic snaking: structure and stability. Chaos Interdiscip J Nonlinear Sci 17(3)

Cardoso A, Leitão J, Teixeira C (2019) Using the Jupyter notebook as a tool to support the teaching and learning processes in engineering courses. In: The challenges of the digital transformation in education: proceedings of the 21st international conference on interactive collaborative learning (ICL2018)-volume 2. Springer, pp 227–236

Castilla R, Peña M (2023) Jupyter notebooks for the study of advanced topics in fluid mechanics. Comput Appl Eng Educ 31(4):1001–1013

Catuhe D, Rousseau M, Lagarde P, Rousset D (2014) Babylon.js a 3D engine based on WebGL and Javascript. https://www.babylonjs.com/

Cervone D (2012) MathJax: a platform for mathematics on the Web. Notices AMS 59(2):312–316

Champneys AR, Al Saadi F, Breña-Medina VF, Grieneisen VA, Marée AF, Verschueren N, Wuyts B (2021) Bistability, wave pinning and localisation in natural reaction-diffusion systems. Physica D Nonlinear Phenom 416:132735

Dillon R, Othmer HG (1999) A mathematical model for outgrowth and spatial patterning of the vertebrate limb bud. J Theor Biol 197(3):295–330

Dimiceli VE, Lang AS, Locke L (2010) Teaching calculus with Wolfram Alpha. Int J Math Educ Sci Technol 41(8):1061–1071

Dirksen J (2013) Learning Three.js: the JavaScript 3D library for WebGL. Packt Publishing Ltd

Ebert D (2014) Graphing projects with Desmos. Math Teach 108(5):388–391

Engelbrecht J, Llinares S, Borba MC (2020) Transformation of the mathematics classroom with the internet. ZDM 52:825–841

Fulton S, Fulton J (2013) HTML5 canvas: native interactivity and animation for the web. O'Reilly

Garfinkel A, Bennoun S, Deeds E, Van Valkenburgh B (2022) Teaching dynamics to biology undergraduates: the UCLA experience. Bull Math Biol 84(3):43

Golman B (2019) A set of Jupyter notebooks for the analysis of transport phenomena and reaction in porous catalyst pellet. Comput Appl Eng Educ 27(3):531–542

Gray P, Scott SK (1984) Autocatalytic reactions in the isothermal, continuous stirred tank reactor: oscillations and instabilities in the system $A + 2B \rightarrow 3B$; $B \rightarrow C$. Chem Eng Sci 39(6):1087–1097

Gustafsson B (2018) Scientific computing: a historical perspective, vol 17. Springer

Hairer E, Hochbruck M, Iserles A, Lubich C (2006) Geometric numerical integration. Oberwolfach Rep 3(1):805–882

Hill DJ, Bramburger JJ, Lloyd DJ (2023) Approximate localised dihedral patterns near a turing instability. Nonlinearity 36(5):2567

Horstmann D (2003) From 1970 until present: the Keller–Segel model in chemotaxis and its consequences I. Jahresber Deutsch Math-Verein 105:103–165

Kaboudian A, Cherry EM, Fenton FH (2019) Large-scale interactive numerical experiments of chaos, solitons and fractals in real time via GPU in a web browser. Chaos Solit Fractals 121:6–29

Kaboudian A, Cherry EM, Fenton FH (2019) Real-time interactive simulations of large-scale systems on personal computers and cell phones: toward patient-specific heart modeling and other applications. Sci Adv 5(3):eaav6019

Kaboudian A, Cherry EM, Fenton FH (2021) Real-time interactive simulations of complex ionic cardiac cell models in 2D and 3D heart structures with GPUs on personal computers. In: 2021 Computing in cardiology (CinC), vol 48. IEEE, pp 1–4

Kalogirou A, Keaveny EE, Papageorgiou DT (2015) An in-depth numerical study of the two-dimensional Kuramoto–Sivashinsky equation. Proc R Soc A Math Phys Eng Sci 471(2179):20140932

King A (2017) Using Desmos to draw in mathematics. Aust Math Teach 73(2):33–37

Klausmeier CA (1999) Regular and irregular patterns in semiarid vegetation. Science 284(5421):1826–1828

Knobloch J, Lloyd DJ, Sandstede B, Wagenknecht T (2011) Isolas of 2-pulse solutions in homoclinic snaking scenarios. J Dyn Differ Equ 23:93–114

Kolokolnikov T, Wei J (2018) Pattern formation in a reaction-diffusion system with space-dependent feed rate. SIAM Rev 60(3):626–645

Kolokolnikov T, Sun W, Ward M, Wei J (2006) The stability of a stripe for the Gierer–Meinhardt model and the effect of saturation. SIAM J Appl Dyn Syst 5(2):313–363

Korucu AT, Cakir H (2018) The effect of dynamic web technologies on student academic achievement in problem-based collaborative learning environment. Malays Online J Educ Technol 6(1):92–108

Krause AL, Klika V, Woolley TE, Gaffney EA (2018) Heterogeneity induces spatiotemporal oscillations in reaction–diffusion systems. Phys Rev E 97(5):052206

Krause AL, Klika V, Maini PK, Headon D, Gaffney EA (2021) Isolating patterns in open reaction–diffusion systems. Bull Math Biol 83(7):1–35

Lau Z, Griffiths IM, English A, Kaouri K (2022) Predicting the spatio-temporal infection risk in indoor spaces using an efficient airborne transmission model. Proc R Soc A 478(2259):20210383

LeVeque RJ (2007) Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems. SIAM

Lobb R, Harlow J (2016) Coderunner: a tool for assessing computer programming skills. ACM Inroads 7(1):47–51

McCauley T (2017) A browser-based event display for the CMS experiment at the LHC using WebGL. J Phys Conf Ser 898:072030

Mejía P, Martini LC, Grijalva F, Larco JC, Rodríguez JC (2021) A survey on mathematical software tools for visually impaired persons: a practical perspective. IEEE Access 9:66929–66947

Meshram V, Meshram V, Patil K (2016) A survey on ubiquitous computing. ICTACT J Soft Comput 6(2):1130–1135

Miranville A (2019) The Cahn–Hilliard equation: recent advances and applications. SIAM

Miura RM (1976) The Korteweg–de Vries equation: a survey of results. SIAM Rev 18(3):412–459

Munasinghe B, Bell T, Robins A (2023) Unplugged activities as a catalyst when teaching introductory programming. J Pedagog Res 7(2):56–71

Murray JD (2003) Mathematical biology: II: spatial models and biomedical applications, 3rd edn. Springer

Necesal P, Pospíšil J (2012) Experience with teaching mathematics for engineers with the aid of Wolfram Alpha. Proc World Congr Eng Comput Sci 1:271–274

Page KM, Maini PK, Monk NAM (2003) Pattern formation in spatially heterogeneous Turing reaction–diffusion models. Physica D 181(1–2):80–101

Page KM, Maini PK, Monk NAM (2005) Complex pattern formation in reaction–diffusion systems with spatially varying parameters. Physica D 202(1–2):95–115

Pearson JE (1993) Complex patterns in a simple system. Science 261(5118):189–192

pmneila (2012) Reaction diffusion system (Gray–Scott model). https://pmneila.github.io/jsexp/grayscott/

Preston-Werner T (2023) Jekyll. https://jekyllrb.com

Reed MC (2004) Why is mathematical biology so hard? Notices AMS 51(3):338–342

Reichenbach T, Mobilia M, Frey E (2007) Mobility promotes and jeopardizes biodiversity in rock-paper-scissors games. Nature 448(7157):1046–1049

Ritchie JS, Krause AL, Van Gorder RA (2022) Turing and wave instabilities in hyperbolic reaction–diffusion systems: the role of second-order time derivatives and cross-diffusion terms on pattern formation. Ann Phys 444:169033

Sangwin C (2007) A brief review of GeoGebra: dynamic mathematics. MSor Connect 7(2):36

Sims K (2022) Reaction–diffusion tutorial. https://karlsims.com/rd.html

Smith N, van der Walt S (2015) A better default colormap for matplotlib. SciPy2015 836

Sun D, Ouyang F, Li Y, Zhu C (2021) Comparing learners' knowledge, behaviors, and attitudes between two instructional modes of computer programming in secondary education. Int J STEM Educ 8(1):1–15

Thomée V (2001) From finite differences to finite elements: a short history of numerical analysis of partial differential equations. In: Numerical analysis: historical developments in the 20th century. Elsevier, pp 361–414

Walker BJ, Townsend AK, Krause AL (2023) GitHub repository: VisualPDE. https://github.com/Pecnut/visual-pde

Woodin T, Carter VC, Fletcher L (2010) Vision and change in biology undergraduate education, a call for action-initial responses. CBE-Life Sci Educ 9(2):71–73

Woolley TE, Krause AL, Gaffney EA (2021) Bespoke Turing systems. Bull Math Biol 83(5):1–32

Zatarain-Cabada R, Barrón-Estrada ML, Cárdenas-Sainz BA, Chavez-Echeagaray ME (2023) Experiences of web-based extended reality technologies for physics education. Comput Appl Eng Educ 31(1):63–82