Stable Advection-Reaction-Diffusion With Arbitrary Anisotropy

Theodore Kim
IBM TJ Watson Research Center

Ming Lin
University of North Carolina at Chapel Hill

ABSTRACT

Turing first theorized that many biological patterns arise through the processes of reaction and diffusion [1]. Subsequently, reaction-diffusion systems have been studied in many fields, including computer graphics. We first show that for visual simulation purposes, reaction-diffusion equations can be made unconditionally stable using a variety of straightforward methods. Second, we propose an anisotropy embedding that significantly expands the space of possible patterns that can be generated. Third, we show that by adding an advection term, the simulation can be coupled to a fluid simulation to produce visually appealing flows. Fourth, we couple fast marching methods to our anisotropy embedding to create a painting interface to the simulation. Unconditional stability to maintained throughout, and our system runs at interactive rates. Finally, we show that on the Cell processor, it is possible to implement reaction-diffusion on top of an existing fluid solver with no significant performance impact.

1. INTRODUCTION

Alan Turing posited that a wide variety of biological patterns such as whorls in leaves and tentacles on Hydras form as a result of two simple physical processes: reaction and diffusion [1]. In general, reaction diffusion systems do not have closed form solutions, so Turing appealed to numerical methods.

In computer graphics, [2] and [3] first introduced reaction-diffusion systems in 1991, and demonstrated the rich set of patterns they can produce. Recently, a variety of numerical integration schemes have become popular in computer graphics that allow the fast, stable simulation of related phenomena, such as the Navier-Stokes equations [4], the inviscid Euler equations [5], and Stokes' flow [6]. In this paper we show how these techniques can be applied to reaction-diffusion.

Using this stable formulation, we significantly extend reaction-diffusion in a variety of ways. First, we show how to embed arbitrary anisotropy functions into the diffusion operator, allowing for the creation of patterns that were not possible with previous graphics techniques (See Figure. 4). Next we show that by adding an advection term, the simulation can be naturally coupled to a fluid solver to create visually appealing flows (See Figure 1). Last, we use fast marching methods and our anisotropy embedding to creating a painting interface to advection-reaction diffusion systems. Unconditional stability is maintained throughout, and our system runs at interactive rates.

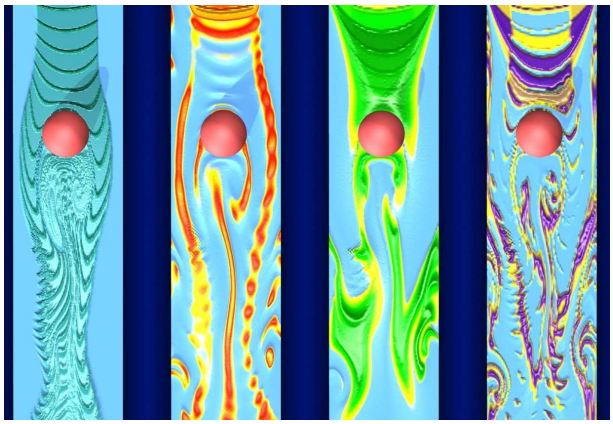


Figure 1: Advection-Reaction-Diffusion result: The leftmost column is an advection-only result, while the right three columns are coupled to various reaction-diffusion equations.

2. PREVIOUS WORK

For a general survey of procedural texture synthesis, we refer the readers to standard textbooks, such as [7], for an exhaustive survey of such techniques. The existing techniques are capable of generating easy to control, visually convincing results. However, their physical basis can be tenuous, making their extension to the patterns we present here difficult.

We use many of the same integration methods as [4] in fluid simulation. However, we will forgo an exhaustive survey of computational fluid dynamics (CFD) in computer graphics, as CFD is not the focus of this paper. We instead mention thematically similar recent works in CFD. The 'passive advection' we describe is similar to the treatment of the temperature and density fields presented by Fedkiw et al. [5].

Iham et al. [8] showed that by coupling reaction effects to the velocity field, explosion effects can be simulated. Unlike our work, their focus was on explosions, not pattern formation. Notably, they used a conditionally stable integration scheme, so our technique could be used to make their scheme more robust.

Bargteil et al. [9] used reaction-diffusion to show effective advection of texture coordinates in a fluid simulation, but the pattern formation was not clearly visible in their simulation. Most likely, this is because the simulation parameters had to be set to avoid a strict timestep restriction. The method we present here could be used to remove this restriction.

Integration of advection-reaction-diffusion equations is a well-studied problem in applied mathematics (eg. [10]), but is usually not presented in the unified form shown in this paper, because the overall convergence of the scheme is considered too low. However, for interactive applications, stability is more important than accuracy, as long as visual plausibility is maintained. To the best of our knowledge, this paper presents the first fast, stable scheme for simulating reaction-diffusion suitable for interactive design.

3. STABLE REACTION DIFFUSION

In this section, we describe the standard mathematical formulation for reaction-diffusion systems and present an unconditionally stable scheme for simulating such a system.

3.1 Reaction-Diffusion Systems

A reaction-diffusion equation for a chemical A takes the general form

$$\frac{\partial A}{\partial t} = \nabla \cdot (a \nabla A) + R \tag{1}$$

where a is a diffusion constant. The $\nabla \cdot (a \nabla A)$ term corresponds to the physical process of diffusion. The R term is the reaction term, and defines how the chemical A interacts with other chemicals. A reaction-diffusion system is obtained when several chemical interact via their reaction terms, such as the two-chemical system shown in Eqns. 2 and 3.

$$\frac{\partial A}{\partial t} = \nabla \cdot (a \nabla A) + R_A(A, B) \tag{2}$$

$$\frac{\partial B}{\partial t} = \nabla \cdot (b \nabla B) + R_B(A, B) \tag{3}$$

These equations are then integrated numerically. This step is usually accomplished by discretizing the two chemicals and replacing the differential operators with finite difference operators. Witkin and Kass [3] discretized the equations over a rectilinear grid, an approach that can suffer from distortion when the final pattern is stretched over an arbitrary model. They proposed a correction matrix that minimized this parametric distortion. Turk [2] circumvented the distortion problem by computing an irregular grid in the form of a Voronoi diagram directly on the surface of a model instead. The reaction-diffusion system was then integrated along this irregular grid. In this paper we adhere more to the approach described by [3] because integration over a rectilinear grid is a well studied problem in applied math, and we would like to be able to draw from this wealth of knowledge.

3.2 An Unconditionally Stable Scheme

For the remainder of this paper, we will employ the following notation. $A^t_{i,j}$ denotes the concentration of A at time t, at grid cell (i, j). Spatial and temporal neighbors are defined as offsets from these indices, i.e. $A^{t+1}_{i,j}$ is the concentration in the grid cell at the next timestep, and $A^t_{i-1,j}$ is the grid cell to the left of $A^t_{i,j}$. The diffusion operator for A is usually discretized as:

$$\nabla \cdot (a \nabla A) \approx \frac{a(A^t_{i-1,j} + A^t_{i,j-1} - 4A^t_{i,j} + A^t_{i+1,j} + A^t_{i,j+1})}{h^2} \tag{4}$$

where h is defined as the physical length of a single grid cell. We can insert this discretization directly into Eqn. 1 to obtain a forward Euler integration scheme (Eqn. 5).

$$\begin{aligned} A^{t+1}_{i,j} = {} & A^t_{i,j} + \Delta t R(A, B) + \\ & \frac{a\Delta t}{h^2}(A^t_{i-1,j} + A^t_{i,j-1} - 4A^t_{i,j} + A^t_{i+1,j} + A^t_{i,j+1}) \end{aligned} \tag{5}$$

The stability condition for this integration scheme is $\frac{2a\Delta t}{h^2} < 1$, where Δt is the desired timestep. This is unacceptably stringent, especially if the diffusion constant a is large, as the timestep must be very small to offset the large a. Depending on the stiffness of the reaction equation, the size of the timestep may have to be reduced further. The usual method for circumventing timestep restrictions is to use implicit integration schemes. In order to apply implicit Euler to both the reaction and diffusion operators, we split the integration into two stages (Eqns. 6 and 13). We begin by integrating the reaction component in Eqn. 6. Let $\widehat{A^t_{i,j}}$ denote an intermediate value for $A^{t+1}_{i,j}$ after reaction but before diffusion.

$$\widehat{A^t_{i,j}} = A^t_{i,j} + \Delta t R(\widehat{A^t_{i,j}}, \widehat{B^t_{i,j}}) \tag{6}$$

The reaction term in reaction-diffusion equations usually takes the form of a polynomial, in which case Eqn. 6 defines a small polynomial system. While non-linear systems are difficult to solve in general, the ones produced by reaction equations are usually small enough that they can be factored by hand and sent to a non-linear solver such as Newton-Raphson at runtime. For example, take the reaction terms for Gray-Scott reaction-diffusion [11],

$$R_A(A, B) = -AB^2 + F(1 - A) \tag{7}$$

$$R_B(A, B) = AB^2 - (F + k)B \tag{8}$$

where F and k are constants. Reaction equations usually do not require any neighbor information, so it is well-known (e.g. [10]) that they can be addressed using ODE methods. We have retained the (i, j) subscripts in the following equations purely for consistency. Substituting the Gray-Scott reaction equations into Eqn. 6, we obtain the following system:

$$\widehat{A^t_{i,j}} = A^t_{i,j} + \Delta t(-\widehat{A^t_{i,j}}\widehat{B^t_{i,j}}^2 + F(1 - \widehat{A^t_{i,j}})) \tag{9}$$

$$\widehat{B^t_{i,j}} = B^t_{i,j} + \Delta t(\widehat{A^t_{i,j}}\widehat{B^t_{i,j}}^2 - (F + k)\widehat{B^t_{i,j}}) \tag{10}$$

We can solve for $\widehat{A^t_{i,j}}$ in terms of $\widehat{B^t_{i,j}}$:

$$\widehat{A^t_{i,j}} = \frac{A^t_{i,j} + F\Delta t}{\Delta t \widehat{B^t_{i,j}}^2 + F\Delta t + 1} \tag{11}$$

Substituting this into $\widehat{B^t_{i,j}}$, we obtain:

$$\widehat{B^t_{i,j}} = B^t_{i,j} + \Delta t \left(\frac{A^t_{i,j} + F\Delta t}{\Delta t \widehat{B^t_{i,j}}^2 + F\Delta t + 1} \widehat{B^t_{i,j}}^2 - (F+k)\widehat{B^t_{i,j}} \right) \tag{12}$$

This is a cubic equation that can be solved using cubic formula, Newton-Raphson, or any other non-linear solver. Although cubic formula may appear attractive because it is an analytical solution, it requires manipulation of complex numbers (even if all the root are real) and involves a large number of arithmetic operations. We instead prefer to use Newton-Raphson because it is faster, simpler to implement, and more general. The solution is then plugged back into Eqn. 11 to complete the integration. While a new derivation is necessary for each different set of reaction equations, this method applies as long as the reaction terms do not contain any additional derivatives. Additional update rules for several common reaction equations are given in Appendix A.

It may seem expensive to perform Newton-Raphson iteration at every grid cell, but since A and B represent concentrations, their values over the lifetime of the simulation is limited to the $[0, 1]$ range. Thus, it is possible to precompute a table of values and perform a lookup at runtime. In practice, this method is faster than even forward Euler.

In the second integration stage, we apply implicit Euler to the diffusion operator thus:

$$A^{t+1}_{i,j} = \widehat{A^t_{i,j}} + \frac{\Delta t a}{h^2}(A^{t+1}_{i-1,j} + A^{t+1}_{i,j-1} - 4A^{t+1}_{i,j} + A^{t+1}_{i+1,j} + A^{t+1}_{i,j+1}) \tag{13}$$

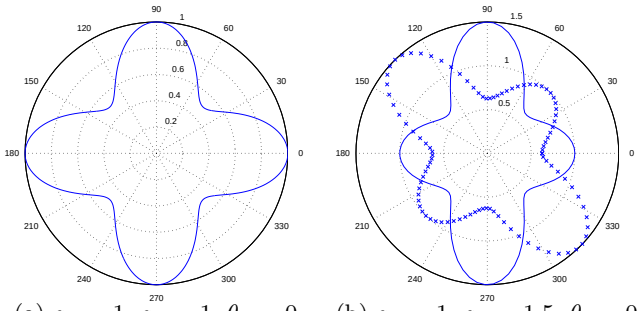
The diffusion operator is now stable for any value of Δt. Since $A^{t+1}_{i,j}$ appears on both sides of Eqn. 13, it defines a linear system that must be solved. Many efficient sparse matrix solvers exist for systems of this type. Incomplete Cholesky Conjugate Gradient (ICCG) is generally considered one of the best choices [5], as it is both fast and handles boundary conditions more naturally than multigrid. Excellent introductory texts exist for the novice [12]. We note extension to second order implicit methods is straightforward. The Crank-Nicolson can be used for the diffusion operator, and second order Adams-Moulton can be used for the reaction equations. An excellent overview of these methods is available in [13].

4. ANISOTROPIC REACTION DIFFUSION

In this section, we present a general anisotropy embedding for creating more interesting reaction-diffusion patterns. This embedding will later allow us to introduce user-specified stroke orientations into the simulation. The anisotropy embedding poses additional numerical challenges, and we show how to maintain unconditional stability in its presence.

4.1 A Generalized Anisotropy Function

Using solely Eqns. 2 and 3, we get patterns with no clear anisotropy. In order to address this limitation, Witkin and



(a) $a_1 = 1$, $a_2 = 1$, $\theta_0 = 0$ (b) $a_1 = 1$, $a_2 = 1.5$, $\theta_0 = 0$ (solid), $\theta_0 = \frac{\pi}{4}$ (dots)

Figure 2: Polar plots of the Witkin-Kass anisotropy function. In Fig. 2(a), the anisotropy strengths a_1 and a_2 are set to be equal, creating four cosine lobes of equal size. In the solid line plot in Fig. 2(b), a_2 is set to be greater, creating larger lobes over $\frac{\pi}{4} < \theta < \frac{3\pi}{4}$ and $\frac{5\pi}{4} < \theta < \frac{7\pi}{4}$. In the dotted line plot in the same figure, the function has been rotated counterclockwise by setting $\theta_0 = \frac{\pi}{4}$.

Kass suggested adding anisotropy to the diffusion operator. This was accomplished by varying the diffusion constant a according to the local orientation. Two separate diffusion values are defined, a_1 for the x axis and a_2 for the y axis. These two diffusion constants, along with a local rotation, were then embedded in the discrete Laplacian. This approach gives directional results, but the anisotropy function is limited to a four-pronged axis-aligned cross, and along each axis only one diffusion speed can be specified. So, for example, separate speeds cannot be defined for the positive and negative x directions.

By embedding the anisotropy directly into the reaction-diffusion equations, we can obtain a more general method. We begin by replacing the diffusion constant with a polar function

$$\frac{\partial A}{\partial t} = \nabla \cdot (a(\theta)\nabla A) + R \tag{14}$$

We define θ here as the angle of the 2D projection of the 3D normal $[\frac{\partial A}{\partial x}, \frac{\partial A}{\partial y}, -1]$. The angle becomes undefined when both derivatives are equal to zero, but this is not a problem because diffusion is also equal to zero in such regions.

In polar coordinates, we can express the Witkin-Kass scheme as:

$$a(\theta) = \begin{cases} \alpha + \frac{a_1 - \alpha}{2}(1 + \cos(4(\theta + \theta_0))) & \begin{cases} -\frac{\pi}{4} \le \theta + \theta_0 \le \frac{\pi}{4} \\ \frac{3\pi}{4} \le \theta + \theta_0 \le \frac{5\pi}{4} \end{cases} \\ \alpha + \frac{a_2 - \alpha}{2}(1 + \cos(4(\theta + \theta_0))) & \begin{cases} \frac{\pi}{4} < \theta + \theta_0 < \frac{3\pi}{4} \\ \frac{5\pi}{4} < \theta + \theta_0 < \frac{7\pi}{4} \end{cases} \end{cases} \tag{15}$$

where $\alpha = \frac{a_1 + a_2}{4}$. Polar plots of Eqn. 15 are shown in Figs. 2(a)-2(b). Intuitively, the Witkin-Kass function can be viewed as four cosine lobes of different amplitude. The '4' in Eqn. 15 specifies four total lobes, the θ_0 rotates the overall function counterclockwise, and the remaining terms of Eqn. 15 are present for normalization. A more general version of a cosine-based anisotropy function of this type is

given by Eqn. 16.

$$a(\theta) = \sum_{n=1}^{j} \frac{a_n}{2}(1+\cos(j(\theta+\theta_0)))\left\{\frac{(n-2)\pi}{j} < \theta+\theta_0 \leq \frac{n\pi}{j}\right. \quad (16)$$

Eqn. 16 describes j cosine lobes, where the influence of each lobe is limited to a $\frac{2\pi}{j}$ slice of the θ domain. Using this formula, we can now overcome the limitations of the Witkin-Kass formula and specify a separate speed a_n for each cosine lobe. In Fig. 4, we show the results of running Gray-Scott reaction diffusion with various settings of our anisotropy function. Note that the Witkin-Kass anisotropy function can only generate the upper left pattern in Fig. 4. Eqn. 16 is not a direct generalization of Eqn. 15, as two boundary values would have to be specified for each cosine lobe. This limitation could be overcome by doubling the number of cosine lobes and halving their domains of influence, but we have found that Eqn. 16 is a more intuitive and compact formula.

Nothing constrains us to solely Eqn. 16, as we can set $a(\theta)$ to any arbitrary polar function. In fact, we are not even limited to analytic forms, and could incorporate user-drawn polar functions. Further exploration of the patterns generated by various functions poses an interesting future direction.

4.2 Maintaining Unconditional Stability

With this addition of an anisotropy function, a stable integration method is needed more than ever. The stability condition for a forward Euler method is now at least $\frac{2\Delta t}{h^2} \cdot |a(\theta)| < 1$, where $|a(\theta)|$ is the maximum value of $a(\theta)$ over $0 \leq \theta \leq 2\pi$. Since the user can specify an anisotropy function with arbitrary large maxima, the timestep can easily become impractically small. However, the discretization given by Eqn. 13 assumes that the diffusion constant a is the same throughout the entire computation domain, and this is no longer true. Instead we must separate Eqn. 3.2 into its four directional components and associate a different diffusion coefficient with each direction. Ideally we would assign each direction the value of $a(\theta)$ that would exist halfway between $A^t_{i,j}$ and each of its neighbors. This produces a symmetric matrix that we can continue to solve using ICCG. We interpolate to find this value by averaging of $a(\theta)$ values between $A^t_{i,j}$ and its neighbors. So, in the anisotropic case, we use Eqn. 17 instead of Eqn. 13,

$$\begin{aligned} A^{t+1}_{i,j} = \widehat{A^t_{i,j}} + \frac{\Delta t}{h^2}\Bigg(&\frac{a(\theta)_{i-1,j}+a(\theta)_{i,j}}{2}(A^{t+1}_{i-1,j} - A^{t+1}_{i,j}) + \\ &\frac{a(\theta)_{i+1,j}+a(\theta)_{i,j}}{2}(A^{t+1}_{i+1,j} - A^{t+1}_{i,j}) + \\ &\frac{a(\theta)_{i,j-1}+a(\theta)_{i,j}}{2}(A^{t+1}_{i,j-1} - A^{t+1}_{i,j}) + \\ &\frac{a(\theta)_{i,j+1}+a(\theta)_{i,j}}{2}(A^{t+1}_{i,j+1} - A^{t+1}_{i,j})\Bigg) \end{aligned} \quad (17)$$

We note that a full discretization of $\nabla \cdot (a(\theta)\nabla A)$ would actually need to take into account a $(\nabla a(\theta))^T(\nabla A)$ term as well. However, we have found that even without this term, we get the desired anisotropy effects, so we have omitted it here for simplicity.

5. STYLIZED FLOW SIMULATION

In this section, we show how to integrate a reaction-diffusion solver with a fluid solver in order to obtain a stylized flow simulation. Formally, this is known as an *advection*-reaction-diffusion system, and an integration scheme exists that maintains the unconditional stability of the overall system.

5.1 Passive Transport

An advection operator usually takes form of $(\mathbf{u}\cdot\nabla)v$, where $\mathbf{u}$ is some vector field and v is some scalar field. The v field should be set to the chemical whose equation the operator is embedded in, and by setting $\mathbf{u}$ to the velocity field of a Stam-style solver, an elegant method of coupling a reaction-diffusion simulator to a fluid simulator is obtained. The general form of such an equation is:

$$\frac{\partial A}{\partial t} = -(\mathbf{u}\cdot\nabla)A + R_A(A,B) + \nabla\cdot(a(\theta)\nabla A) \quad (18)$$

where $\mathbf{u}$ is the external velocity field. An example of passive advection is shown in Fig. 5. Passive transport is by no means the only possible utility of $\mathbf{u}$, just its usual meaning.

5.2 An Unconditionally Stable Scheme

Advection imposes the timestep restriction of $\Delta t < \frac{\Delta x}{|\mathbf{u}|}$, where Δx is the size of a grid cell, and $|\mathbf{u}|$ is the magnitude of the largest velocity over the entire grid. In computer graphics, Stam first circumvented this restriction using a Semi-Lagrangian integration scheme [4]. In order to maintain unconditional stability in the above advection-reaction-diffusion equations, we suggest using the same scheme. Stam recommended that the advection step be run after the diffusion step, since the diffusion step smears out the current values and provides more information for the advection backtraces to 'grab'. We must therefore rewrite Eqn. 17 to treat the results as intermediate values:

$$\begin{aligned} \tilde{A}^{t+1}_{i,j} = \widehat{A^t_{i,j}} + \frac{\Delta t}{h^2}\Bigg(&\frac{a(\theta)_{i-1,j}+a(\theta)_{i,j}}{2}(\tilde{A}^{t+1}_{i-1,j} - \tilde{A}^{t+1}_{i,j}) + \\ &\frac{a(\theta)_{i+1,j}+a(\theta)_{i,j}}{2}(\tilde{A}^{t+1}_{i+1,j} - \tilde{A}^{t+1}_{i,j}) + \\ &\frac{a(\theta)_{i,j-1}+a(\theta)_{i,j}}{2}(\tilde{A}^{t+1}_{i,j-1} - \tilde{A}^{t+1}_{i,j}) + \\ &\frac{a(\theta)_{i,j+1}+a(\theta)_{i,j}}{2}(\tilde{A}^{t+1}_{i,j+1} - \tilde{A}^{t+1}_{i,j})\Bigg) \end{aligned} \quad (19)$$

where $\tilde{A}^{t+1}_{i,j}$ represents an intermediate value after reaction and diffusion, but before advection.

A linear Semi-Lagrangian advection scheme is

$$A^{t+1}_{i,j} = \tilde{A}^{t+1}_{k,l}\left\{\begin{array}{l} k = i - \frac{\Delta t}{\Delta x}\cdot u(x) \\ l = j - \frac{\Delta t}{\Delta y}\cdot u(y) \end{array}\right. \quad (20)$$

where $u(x)$ and $u(y)$ correspond to the x and y components of the vector $\mathbf{u}$. Usually k and l do not yield integers, so a bilinear interpolation is performed. If the values of k and l are outside the grid, they can be clamped to borders or wrapped periodically. For a cubic version of Eqn. 20, see [5]. The full, final integration scheme is Eqns. 6, 19, and 20.

6. A PAINTING INTERFACE

The scheme we propose is fast enough that it can run at interactive rates on non-trivial grid sizes. However, this interactivity is most useful if it is coupled with an intuitive interface. A naïve interface would merely have the user deposit chemicals in the simulation domain using a mouse or pen tablet. Creating interesting patterns can be difficult with only these basic controls.

Given the large number of tunable variables available in the reaction-diffusion simulation, it seems natural to semantically couple some of these to the user input. We propose one such coupling that ties the anisotropy orientation (θ_0 in Eqn. 16) to the direction of the user's brushstrokes. The θ_0 variable is usually kept constant over the domain, but we will vary it spatially so that it causes the reaction-diffusion patterns to form in the direction of the strokes.

When the user deposits a stroke, we rasterize its position and direction on a regular grid. This defines the θ_0 values of the grid cells that lie directly underneath a stroke. We then propagate these orientations to grid cells that do not lie underneath a stroke using fast extension velocities [14], which is a variant of fast marching methods [15, 16]. Briefly, the fast marching method constructs a distance field around an interface (in our case, the rasterized brushstrokes), and the fast extension velocity algorithm adds a step that propagates scalar values defined on the interface to cells far from the interface. In our case, our scalar value is an orientation value in the $(0, 2\pi)$ range. The overall effect is that each grid cell is constrained to form reaction-diffusion patterns according to the direction of the nearest brush stroke.

7. IMPLEMENTATION AND RESULTS

We have implemented the techniques presented here in C++ and coupled it to a tablet interface, and demonstrated their effectiveness on various applications.

7.1 Applications and Demonstrations

We applied several different anisotropy functions to Grey-Scott reaction diffusion to generate the "space cookies" [3] in Fig. 4. Note that with the exception of the upper left cookie, the Witkin-Kass anisotropy function would require a great deal of user intervention to generate similar patterns. However, using our method, the user only needs to specify the initial conditions.

Three different purely reaction-diffusion systems, Gray-Scott, FitzHugh-Nagumo [17, 18], and turbulent Barkley [19], were used to create the patterns in the Figs. 5 and 6. FitzHugh-Nagumo generates periodic heartbeats, while turbulent Barkley generates a turbulent version of FitzHugh-Nagumo, much like a defibrillating heart. The final results are suitable for use as static textures, while the temporal evolution of the system can be used as animations or flow visualizations.

In Figure 5, we simulated reaction-diffusion in the lid-driven cavity flow example described in [20]. We show the simulation with passive advection only, and then stylized with the three different kinds of reaction-diffusion. In Fig. 6 we simulate reaction-diffusion in the flow past an obstacle example described in [20]. We compare flows with advection only, reaction-diffusion only, and full advection-reaction-diffusion. All the reaction-diffusion simulations utilized Dirichlet boundary conditions. Such techniques can be useful for the generation and rapid visualization of visual effects. Please see the supplementary video clips to see animations of Figs. 5 and 6 produced by the integration schemes presented here.

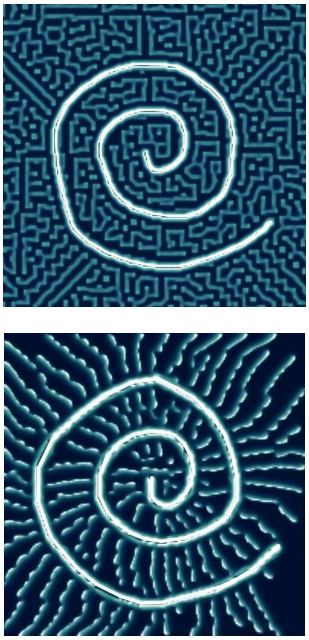


Figure 3: Patterns generated interactively using our advection-reaction-diffusion scheme with a tablet interface. In both cases the user drew the white spiral. *Top:* A pattern generated using the basic 'deposition only' method. The pattern grows outwards once and then stops. *Bottom:* Pattern generated using the brushstroke - θ_0 coupling. Waves perpetually grow out radially from the stroke, and then gradually spiral inwards toward the center.

We have implemented the painting interface we described in section 6, and integrated it with a pen tablet input. Using this interface, the user can generate complex patterns with a very small number of strokes. In the right half of Figure 3, the anisotropy function was set to a single cosine lobe that propagates in the opposite direction of the stroke. The user input a single spiral-shaped stroke, and stripes then radiate outwards perpendicular to the stroke. Once they stabilize, the stripes spiral inwards toward the center, as prescribed by the anisotropy function. The pattern is self-sustaining and continues indefinitely. If we instead use the naïve deposition-only interface (Figure 3 left), we still get an interesting pattern, but it propagates outwards once and remains static. The user stroke exists only as a boundary, and the stroke is not reflected in the overall pattern.

7.2 Performance

We have implemented our stable reaction-diffusion scheme on top of a Stam stable fluid solver [4] on the Cell processor. Somewhat surprisingly, it is possible to add our scheme with *no significant impact* on the total running time. A detailed flop and bandwidth analysis shows that this is because the fluid solver is a memory-bound process, and the Cell's

Resolution	Fluids only	Fluids + RD
128^2	223 FPS	223 FPS
256^2	78.7 FPS	78.5 FPS
512^2	21.4 FPS	21.5 FPS
1024^2	5.5 FPS	5.5 FPS

Table 1: Running time of a fluid solver on the Cell processor, and the fluid solver with reaction-diffusion added. All measurements are in frames per second. The timings are nearly identical, with slight discrepancies attributable to thread synchronization noise.

synergistic processing elements (SPEs) are idle roughly 75% of the time. By judiciously placing the reaction-diffusion computation, it is possible to exploit this idle time with no additional impact on the overall running time. We show the performance of the fluid solver and the fluid solver plus reaction-diffusion in Table 1. The slight discrepancies in between the running times can be attributed to thread synchronization noise.

While at first glance it appears that a non-linear solve per grid cell would be quite expensive, there are in fact ample unused cycles to devote to this computation. Since this running time of both schemes can be entirely hidden, it makes little sense to compare their running times directly. The implicit method is technically 'infinitely' faster than the explicit method, since it allows arbitrarily large timesteps. It is up to the user to decide when the results of the implicit method are too smeared out to be of use in their specific application.

7.3 Limitations

While our approach is fast and stable, the algorithm does have some limitations. The scheme does not guarantee that identical patterns can be obtained using large steps in place of small steps. For example, in the extreme, it is still not possible to generate leopard spots in a single timestep. Perhaps higher-order methods or a hybrid method using the cellular approach of [21] may be able to address this problem.

While we mentioned unconditionally stable second-order schemes in section 3, they do not extend to higher orders due to the second Dahlquist stability barrier [13]. To achieve an unconditionally stable scheme greater than second order, a different approach will be needed.

8. CONCLUSIONS AND FUTURE WORK

We have presented unconditionally stable schemes for reaction-diffusion, anisotropic reaction-diffusion, and advection-reaction-diffusion for pattern generation and stylized flow simulations. The PDEs we used are dimensionless, so they apply to 3D as well as 2D. In 3D, a 7-point discrete Laplacian is used in Eqn. 19, a z-coordinate lookup is added to Eqn. 20, and the polar function in Eqn. 14 becomes spherical.

We have presented one set of reaction terms, one possible advection operator in Section 5, and a cosine-based anisotropic diffusion function in Section 4. These are by no means the only possibilities; the space of functions that could be plugged into these terms is virtually infinite. This space of functions can now be explored without fear of numerical instability, and ideally classes of interesting patterns can be mapped out at interactive rates. There still remain classes of functions that our approach cannot handle. If the reaction terms contain time or space derivatives, the ODE approach we describe is no longer valid and a different approach is necessary. We are not aware of any scheme that removes this problem, so it poses an interesting direction for future work.

Finally, we have only proposed one semantic coupling between user-input brush strokes and the simulation, namely the θ_0 variable. This is by no means the only possible coupling, and further exploration is necessary to determine what coupling is the most intuitive and semantically meaningful.

Appendix A

For brevity, we have dropped the superscripts and subscripts in the following equations. Keep in mind that $\widehat{A} = \widehat{A_{i,j}^{t+1}}$, $\widehat{B} = \widehat{B_{i,j}^{t+1}}$, $A = A_{i,j}^t$, and $B = B_{i,j}^t$. In all cases, $\widehat{A}$ should be solved with a non-linear solver and substituted into $\widehat{B}$. All undocumented variables are constants.

Turing's Spot Formation Equations

Turing's spot formation reaction terms [1] are:

$$\begin{aligned} R_A &= s(AB - A - \beta) \\ R_B &= s(16 - AB) \end{aligned}$$

The implicit update rules are:

$$\begin{aligned} \widehat{A} &= A + s\Delta t\left(\frac{B + 16\Delta t}{1 + s\widehat{A}\Delta t}\widehat{A} - \widehat{A} - \beta\right). \\ \widehat{B} &= \frac{B + 16\Delta t}{1 + s\widehat{A}\Delta t} \end{aligned}$$

Meinhardt's Spot Formation Equations

Meinhardt's spot formation reaction terms [22] are:

$$\begin{aligned} R_A &= s(Ap_1 + \frac{0.01\alpha A^2}{B} + p_3) \\ R_B &= s(Bp_2 + 0.01\alpha A^2) \end{aligned}$$

The implicit update rules are:

$$\begin{aligned} \widehat{A} &= A + \Delta ts\left(\widehat{A}p_1 + \frac{(1 - \Delta tsp_2)(0.01\alpha\widehat{A}^2)}{B + \Delta ts(0.01\alpha\widehat{A}^2)} + p_3\right) \\ \widehat{B} &= \frac{B + \Delta ts(0.01\alpha\widehat{A}^2)}{1 - \Delta tsp_2} \end{aligned}$$

FitzHugh-Nagumo Model

The FitzHugh-Nagumo [17, 18] reaction terms are:

$$R_A = A - A^3 - B$$

$$R_B = \varepsilon(A - a_1 B - a_0)$$

The implicit update rules are:

$$\begin{aligned} \widehat{A} &= A + \Delta t \left(\widehat{A} - \widehat{A}^3 - \frac{B + \Delta t \varepsilon (\widehat{A} - a_0)}{1 + \Delta t \varepsilon a_1} \right) \\ \widehat{B} &= \frac{B + \Delta t \varepsilon (\widehat{A} - a_0)}{1 + \Delta t \varepsilon a_1} \end{aligned}$$

The Barkley Model

The Barkley [19] reaction terms are:

$$\begin{aligned} R_A &= \frac{1}{\varepsilon} A(1 - A) \left(A - \frac{V + b}{a} \right) \\ R_B &= A - B \end{aligned}$$

The implicit update rules are:

$$\begin{aligned} \widehat{A} &= A + \frac{\Delta t}{\varepsilon} \left(\widehat{A}(1 - \widehat{A}) \left(\widehat{A} - \frac{\frac{B + \Delta t \widehat{A}}{1 + \Delta t} + b}{a} \right) \right) \\ \widehat{B} &= \frac{B + \Delta t \widehat{A}}{1 + \Delta t} \end{aligned}$$

Barkley's turbulent flow model is the same as above except $R_B = A^3 - B$.

9. REFERENCES

[1] Alan Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society B*, 237:37–72, 1952.

[2] G. Turk. Generating textures on arbitrary surfaces using reaction-diffusion. *Proc. of SIGGRAPH*, pages 289–298, 1991.

[3] A. Witkin and M. Kass. Reaction-diffusion textures. *Proc. of SIGGRAPH*, pages pp. 299–308, 1991.

[4] Jos Stam. Stable fluids. *Proc. of SIGGRAPH*, pages 121–128, 1999.

[5] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. *Proc. of SIGGRAPH*, pages 15–22, 2001.

[6] B. Baxter, Y. Liu, and M. Lin. A viscous paint model for interactive applications. *Computer Animation and Virtual Worlds Journal*, 15:433–442, 2004.

[7] D. Ebert, F. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing and Modeling: A Procedural Approach*. AP Professional, 1998.

[8] I. Ihm, B. Kang, and D. Cha. Animation of reactive gaseous fluids through chemical kinetics. *Proc. of ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 203–212, 2004.

[9] Adam Bargteil, Tolga Goktekin, James OBrien, and John Strain. A semi-lagrangian contouring method for fluid simulation. *ACM Transactions on Graphics*, 2006.

[10] W. Hundsdorfer and J.G. Verwer. *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*. Springer Verlag, 2003.

[11] J.E. Pearson. Complex patterns in a simple system. *Science*, 261:189–192, 1993.

[12] Jonathan R Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Carnegie Mellon University, 1994.

[13] Lloyd N. Trefethen. *Finite Difference and Spectral Methods for Ordinary and Partial Differential Equations*. unpublished text, 1996. available at http://web.comlab.ox.ac.uk/oucl/work/nick.trefethen/pdetext.html.

[14] David Adalsteinsson and James Sethian. The fast construction of extension velocities in level set methods. *Journal of Computational Physics*, 148:2–22, 1999.

[15] J. N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9), 1995.

[16] James Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. Nat. Acad. Sci.*, 93, 1996.

[17] R. FitzHugh. Impulse and physiological states in models of nerve membrane. *Biophysics Journal*, 1:445–466, 1961.

[18] J. S. Nagumo, S. Arimoto, and S. Yoshizawa. An active pulse transmission line simulating nerve axon. *Proc. IRE*, 50:2061–2071, 1962.

[19] Dwight Barkley. A model for fast computer simulation of waves in excitable media. *Physica D*, 49:61–70, 1991.

[20] Michael Griebel, Thomas Dornseifer, and Tilman Neunhoeffer. *Numerical Simulation in Fluid Dynamics: A Practical Introduction*. SIAM, 1997.

[21] Kurt W. Fleischer, David H. Laidlaw, Bena L. Currin, and Alan H. Barr. Cellular texture generation. In *Proceedings of SIGGRAPH '95*, pages 239–248, 1995.

[22] Hans Meinhardt. *Models of Biological Pattern Formation*. Academic Press, 1982.

Figure 4: Space Cookies Redux: Clockwise from lower left, cookies were generated with 3,4,5, and 8 lobed anisotropy functions. Previous methods can only produce the upper left cookie.

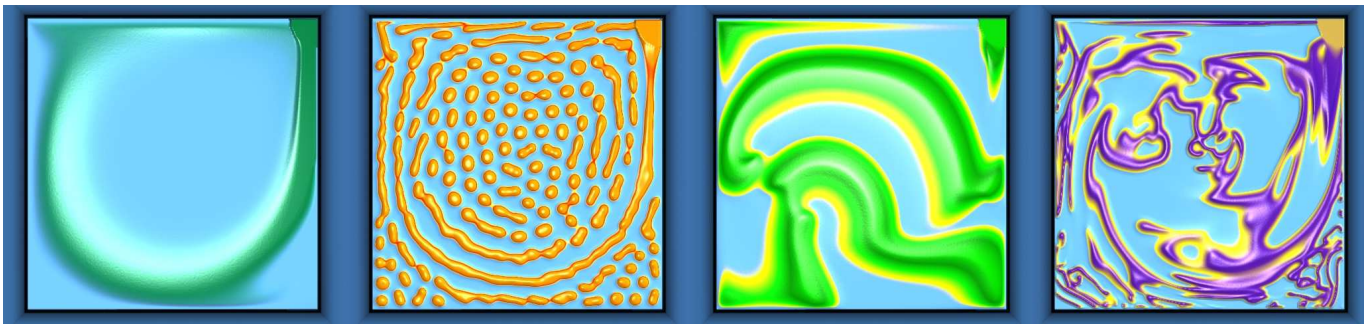


Figure 5: Stylized lid-driven cavity: Left to right, passive advection, Grey-Scott reaction-diffusion (RD), FitzHugh-Nagumo RD, Barkley RD. With passive advection, a relatively uninteresting result is obtained, even though the underlying flow is quite interesting. Coupled with RD, a much more interesting visual result is obtained.

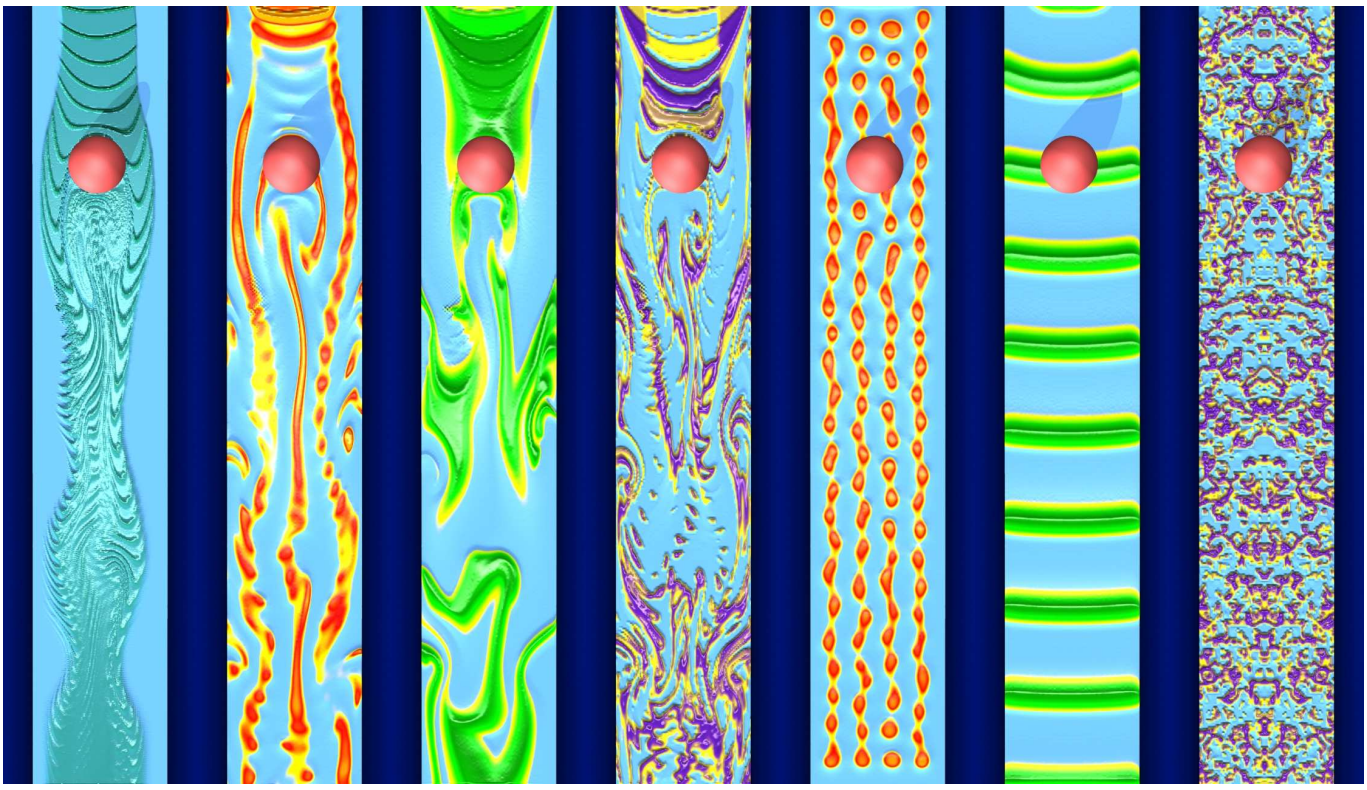


Figure 6: Stylized flow past an obstacle: Chemical flows in from the top, past the ball, and out the bottom. From left to right: passive advection, Gray-Scott advection-reaction-diffusion (ARD), FitzHugh-Nagumo ARD, Barkley ARD, Gray-Scott reaction-diffusion only (RD), FitzHugh-Nagumo RD, Barkley RD. All flows are from the same timestep in the flow simulation. The ball is not included in the RD simulation.