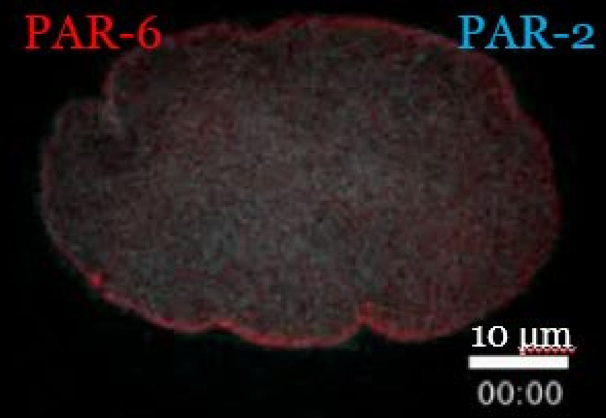
Modeling Reactive and Diffusive Biological Systems

QLS Breakfast Seminar
25 September 2024

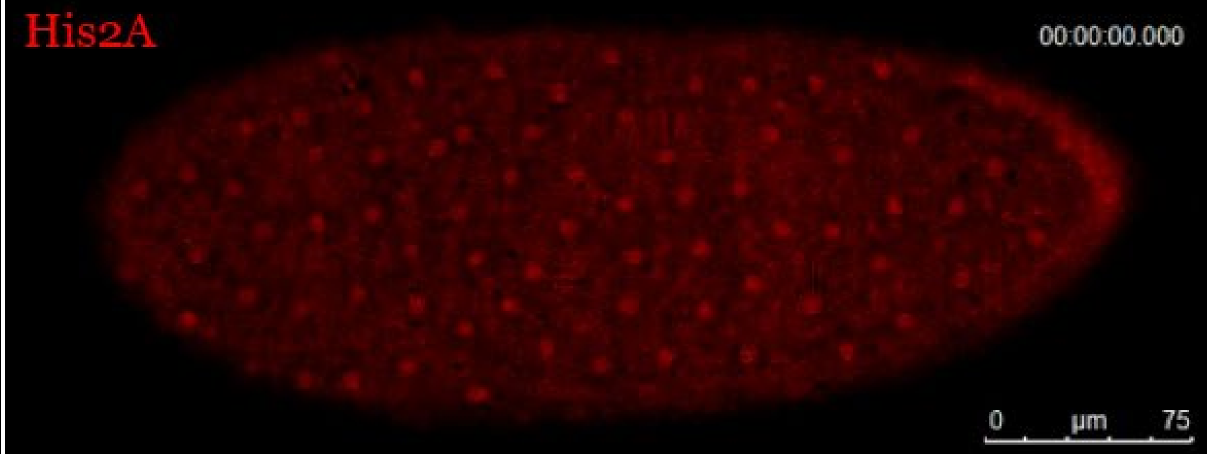
Rocky Diegmiller

cell polarization



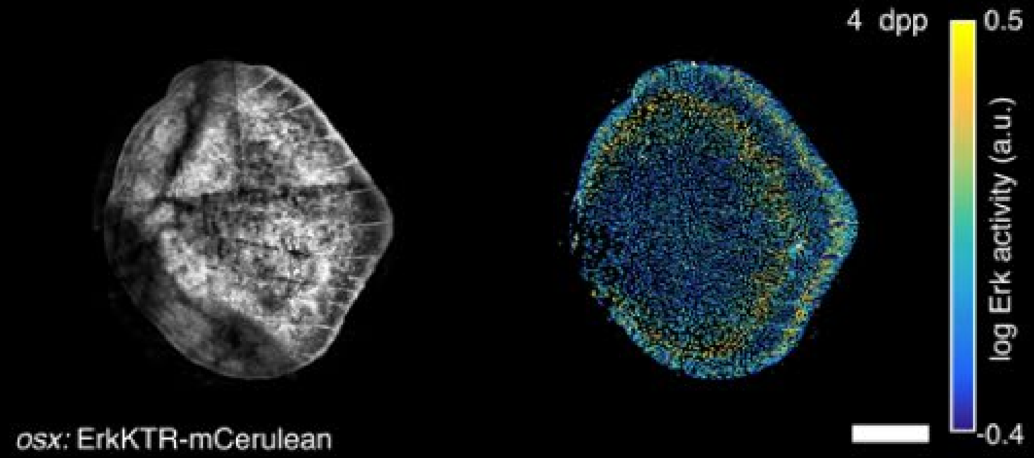
C. elegans embryo

waves of coordinated cell behavior



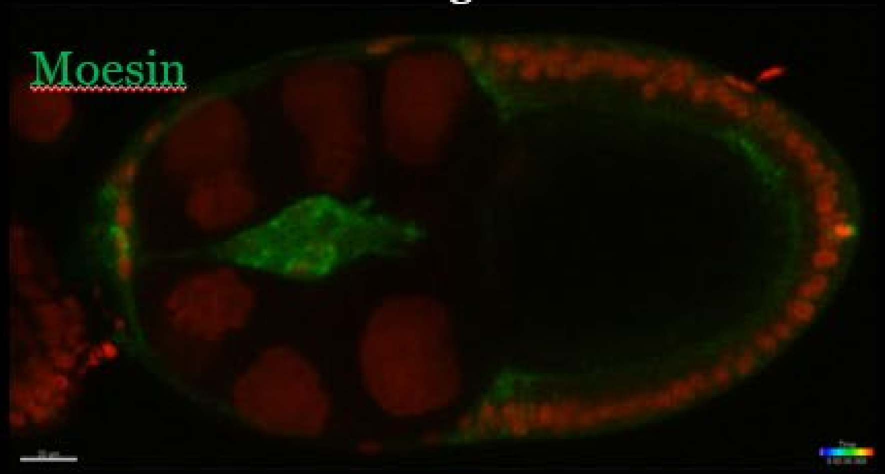
D. melanogaster embryo

appendage regeneration



D. rerio scale

cell migration



D. melanogaster egg chamber

Polarization of PAR Proteins by Advective Triggering of a Pattern-Forming System

Nathan W. Goehring,[1] Philipp Khuc Trong,[2,1*] Justin S. Bois,[2,1†] Debanjan Chowdhury,[2‡] Ernesto M. Nicola,[2§] Anthony A. Hyman,[1] Stephan W. Grill[2,1||]

$$\boxed{\partial_t} A = D_A \boxed{\partial_x^2} A - \boxed{\partial_x}(vA) + R_A$$

$$\boxed{\partial_t} P = D_P \boxed{\partial_x^2} P - \boxed{\partial_x}(vP) + R_P$$

Science, 2011

Waves of Cdk1 Activity in S Phase Synchronize the Cell Cycle in *Drosophila* Embryos

Victoria E. Deneke,[1] Anna Melbinger,[2] Massimo Vergassola,[2] and Stefano Di Talia[1,3,*]
[1]Department of Cell Biology, Duke University Medical Center, Durham, NC 27710, USA
[2]Department of Physics, University of California San Diego, La Jolla, CA 92093, USA
[3]Lead Contact
*Correspondence: stefano.ditalia@duke.edu
http://dx.doi.org/10.1016/j.devcel.2016.07.023

$$\boxed{\frac{\partial f}{\partial t}} = D_{Chk1} \boxed{\frac{\partial^2 f}{\partial x^2}} - \frac{a^\sigma}{K^\sigma_{Chk1} + a^\sigma} r_0 f + \xi_f(x,t)$$

$$\boxed{\frac{\partial a}{\partial t}} = D_{Cdk} \boxed{\frac{\partial^2 a}{\partial x^2}} + \alpha + r_+(a,f)(c(x,t) - a) - r_-(a,f) + \xi_c(x,t) + \xi_r(x,t)$$

$$\boxed{\frac{\partial c}{\partial t}} = D_{Cdk} \boxed{\frac{\partial^2 c}{\partial x^2}} + \alpha + \xi_c(x,t)$$

Dev Cell, 2016

Control of osteoblast regeneration by a train of Erk activity waves

https://doi.org/10.1038/s41586-020-03085-8
Received: 8 October 2019

Alessandro De Simone[1,2], Maya N. Evanitsky[1,2], Luke Hayden[1,2], Ben D. Cox[1,2,6], Julia Wang[1,2], Valerie A. Tornini[1,2,7], Jianhong Ou[1], Anna Chao[1,2], Kenneth D. Poss[1,2,3,4] & Stefano Di Talia[1,2,5]

$$\frac{dE}{dt} = \frac{\alpha_1 A^2}{\beta_1^2 + A^2}(1 - E) - E(\gamma_1 I + \gamma_e)$$

$$\boxed{\frac{\partial A}{\partial t}} = \alpha_2 + \alpha_3 E - \gamma_2 A + D \boxed{\nabla^2} A$$

$$\frac{dI}{dt} = \gamma_3(\alpha_4 E - I)$$

osx: ErkKTR-m

log Erk

-0.4

Nature, 2021

Modeling and analysis of collective cell migration in an in vivo three-dimensional environment

[a]Molecular, Cellular and Developmental Biology, University of California Santa Barbara, CA 93106; [b]Department of Biological Chemistry, Johns Hopkins School of Medicine, Baltimore, MD 21205; [c]Department of Biological Sciences, Indian Institute of Science Education and Research Kolkata, West Bengal 741252, India; and [d]Department of Chemical Physics, The Weizmann Institute of Science, Rehovot 76100, Israel

$$\dot{\rho}_a = c\rho_n - \Gamma\rho_a,$$

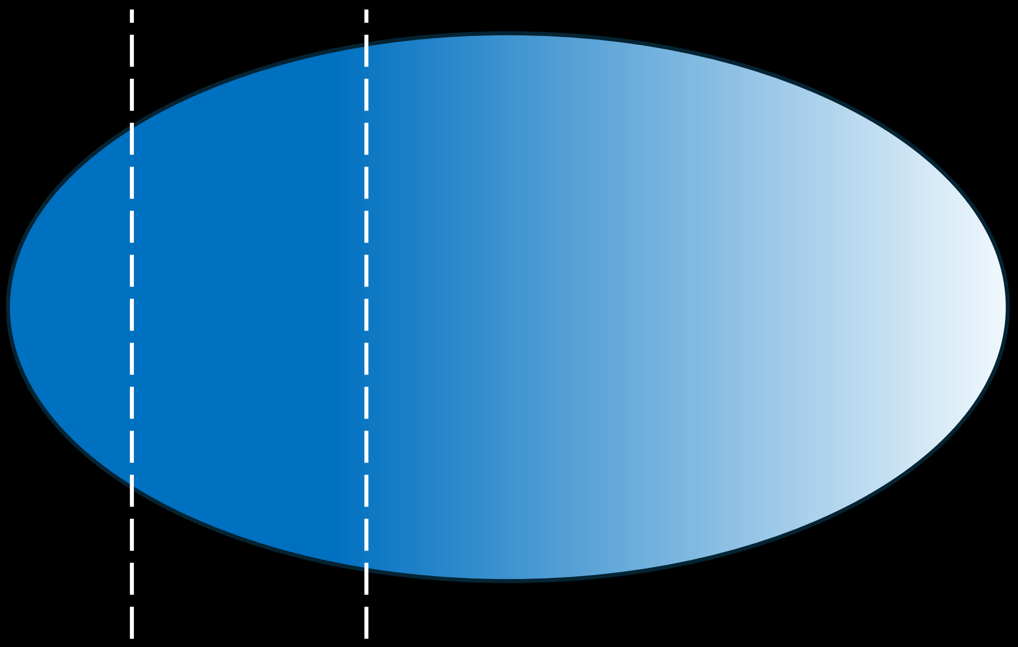
$$\dot{\rho}_n = -c\rho_n - \Gamma\rho_n + \alpha,$$

D. melanogaster egg chamber

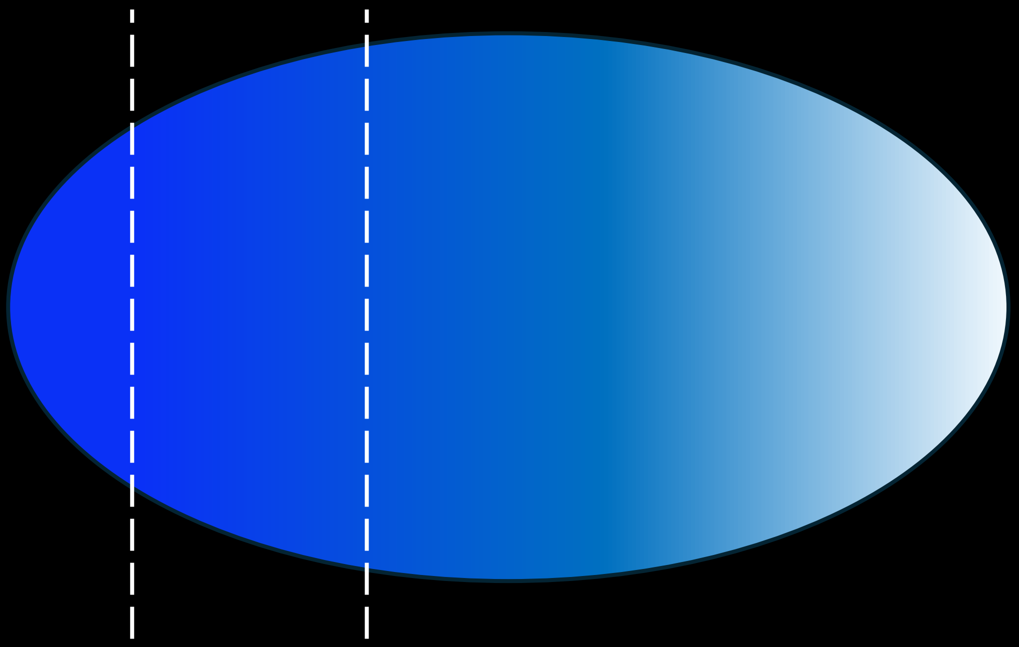
PNAS, 2016

What about systems that change in space AND time!?

Modeling rates of change in space and time



$C(x,t)$ $C(x+\Delta x,t)$



$C(x,t+\Delta t)$ $C(x+\Delta x,t+\Delta t)$

$$\frac{\partial C(x,t)}{\partial x} = \lim_{\Delta x \to 0} \frac{C(x+\Delta x,t) - C(x,t)}{\Delta x}$$

$$\frac{\partial C(x,t)}{\partial t} = \lim_{\Delta t \to 0} \frac{C(x,t+\Delta t) - C(x,t)}{\Delta t}$$

Modeling rates of change in space and time

- Previously, we discussed how we can use MATLAB to numerically solve systems of ODEs and ODEs of arbitrary order

- Now we are faced with the task of solving partial differential equations, where more than one independent variable exists within the system

- It may not come as a shock, but explicit solutions to PDEs are rare; in addition, numerical schemes and programs need to be more careful to remain accurate (and stable)

Finite differences for approximating derivatives

- Flashback to calculus – Taylor series expansion of a function about a point:

$$f(y) = f(y_0) + f'(y_0)(y - y_0) + \frac{1}{2}f''(y_0)(y - y_0)^2 + \frac{1}{6}f'''(y_0)(y - y_0)^3 + \cdots$$

- Now, pick $y_0 = x$ and $y = x + \Delta x$ (and truncate the series):

$$f(x + \Delta x) \approx f(x) + f'(x)(\Delta x) + \frac{1}{2}f''(x)(\Delta x)^2 \qquad (*)$$

- Alternatively, we could pick $y = x - \Delta x$ to get:

$$f(x - \Delta x) \approx f(x) - f'(x)(\Delta x) + \frac{1}{2}f''(x)(\Delta x)^2 \qquad (**)$$

Finite differences for approximating derivatives

- Truncating $(*)$ before the $f''(x)$ term yields a first order approx. for $f'(x)$:

$$f(x+\Delta x) \approx f(x) + f'(x)(\Delta x) \qquad \Rightarrow \boxed{f'(x) \approx \frac{f(x+\Delta x) - f(x)}{(\Delta x)}}$$

- Subtracting $(**)$ from $(*)$ yields a second order approx. for $f'(x)$:

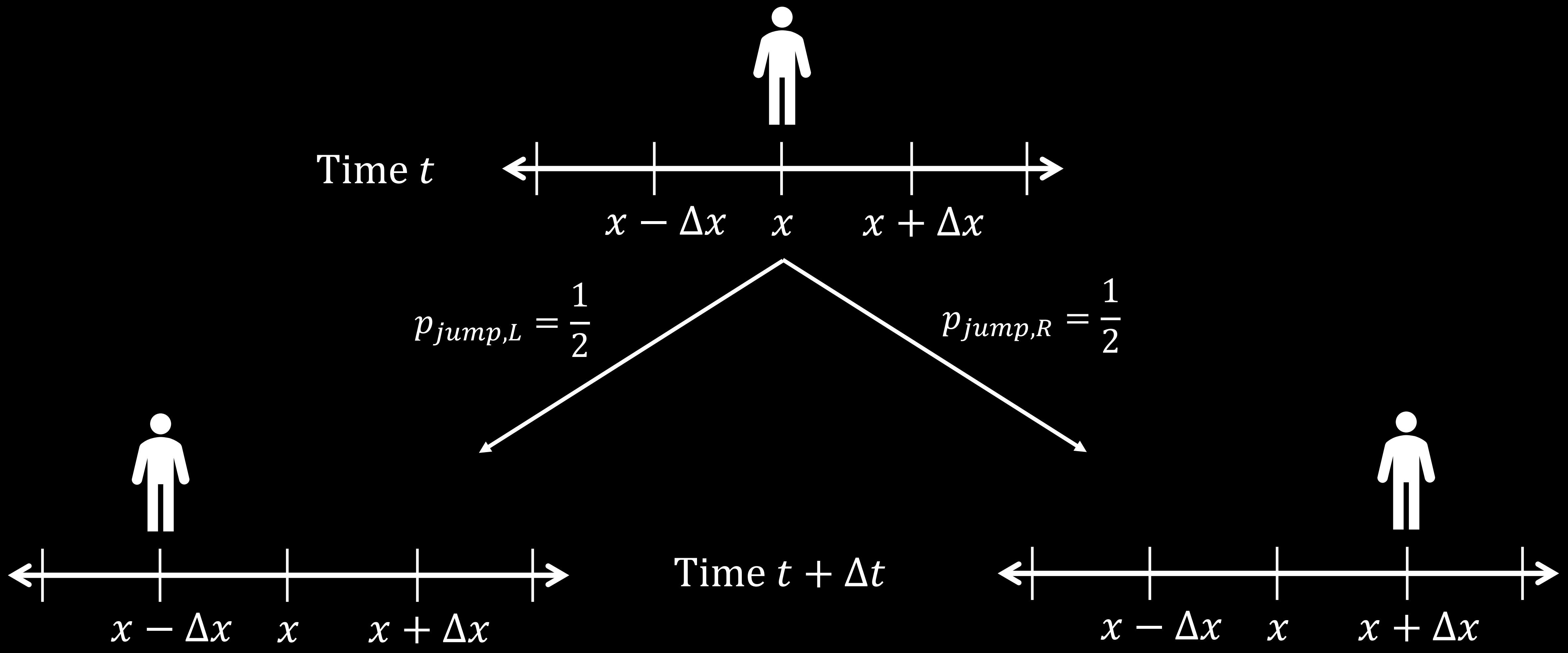
$$f(x+\Delta x) - f(x-\Delta x) \approx 2f'(x)(\Delta x) \quad \Rightarrow \boxed{f'(x) \approx \frac{f(x+\Delta x) - f(x-\Delta x)}{2(\Delta x)}}$$

- Similarly, adding $(**)$ and $(*)$ yields a second order approx. for $f''(x)$:

$$f(x+\Delta x) + f(x-\Delta x) \approx 2f(x) + f''(x)(\Delta x)^2$$

$$\Rightarrow \boxed{f''(x) \approx \frac{f(x+\Delta x) - 2f(x) + f(x-\Delta x)}{(\Delta x)^2}}$$

Random walks and the (1D) diffusion equation



Random walks and the (1D) diffusion equation

- Consider a person at position x and time $t + \Delta t$
- To be here, they must have jumped from the left $(x - \Delta x)$ or right $(x + \Delta x)$ at time t
- Writing this in probabilistic terms (jumping occurs with equal prob.), we have:

$$P(x, t + \Delta t) \;= \frac{1}{2} P(x + \Delta x, t) + \frac{1}{2} P(x - \Delta x, t)$$

- Subtract $P(x, t)$ from both sides and multiply the LHS by $\frac{\Delta t}{\Delta t}$ and RHS by $\frac{(\Delta x)^2}{(\Delta x)^2}$

$$\Rightarrow \Delta t \frac{P(x, t + \Delta t) - P(x, t)}{\Delta t} = \frac{(\Delta x)^2}{2} \left(\frac{P(x + \Delta x, t) - 2P(x, t) + P(x - \Delta x, t)}{(\Delta x)^2} \right)$$

Random walks and the (1D) diffusion equation

- But the fractions on each side are just approx. of derivatives (for small $\Delta t, \Delta x$):

$$\Delta t \frac{P(x, t+\Delta t) - P(x,t)}{\Delta t} = \frac{(\Delta x)^2}{2}\left(\frac{P(x+\Delta x, t) - 2P(x,t) + P(x-\Delta x, t)}{(\Delta x)^2}\right)$$

$$\Rightarrow \frac{\partial P}{\partial t} = \frac{(\Delta x)^2}{2\Delta t}\left(\frac{\partial^2 P}{\partial x^2}\right)$$

- This prefactor can be redefined to give us a PDE describing diffusion:

$$D \equiv \frac{(\Delta x)^2}{2\Delta t} \;[=]\; \frac{(\mathrm{LENGTH})^2}{\mathrm{TIME}} \qquad \Rightarrow \boxed{\frac{\partial P}{\partial t} = D\left(\frac{\partial^2 P}{\partial x^2}\right)}$$

(NOTE: in some contexts, D is called α and the PDE is called the heat equation)

Reaction-Advection-Diffusion Equation in 1D

- Let the concentration of a chemical species at a point in time and space be $u(x, t)$
- In addition to diffusion, this species could undergo some sort of reaction $R(u)$, or it may also be affected by some external signal or induced flow (advection) at rate c
- These two new behaviors can be integrated into our PDE to give us the more generalized (1D) Reaction-Advection-Diffusion Equation:

$$\boxed{\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + c\frac{\partial u}{\partial x} + R(u)}$$

Numerically solving (1D) PDEs in MATLAB

- Consider the 1D diffusion equation in the interval $x \in [0,1]$:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}, \qquad \left.\frac{\partial u}{\partial x}\right|_{x=0} = \left.\frac{\partial u}{\partial x}\right|_{x=1} = 0, \quad u(x,0) = \frac{10}{\sqrt{2\pi}}\exp\left(-\frac{(x-0.5)^2}{0.02}\right)$$

- These boundary conditions at the endpoints are called no-flux BCs, as they essentially act as barriers to stop material from escaping the domain

- The initial condition here is just a Gaussian profile centered in the middle of the domain that will let us visualize diffusion at work over time

Numerically solving (1D) PDEs in MATLAB

- Time for some (not-so-helpful) notation from the MathWorks website:

$$\underbrace{c\left(x,t,u,\frac{\partial u}{\partial x}\right)}_{\text{coupling}}\frac{\partial u}{\partial t} = x^{-n}\frac{\partial}{\partial x}\left(x^n \underbrace{f\left(x,t,u\,\frac{\partial u}{\partial x}\right)}_{\text{flux}}\right) + \underbrace{s\left(x,t,u,\frac{\partial u}{\partial x}\right)}_{\text{source}}$$

$$a \le x \le b, \qquad t_i \le t \le t_f$$

- We can match c, f, s to our PDE in the following way:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} \quad \Rightarrow \quad c = 1, f = D\frac{\partial u}{\partial x}, s = 0$$

$(n = 0)$ as well

$$0 \le x \le 1, \qquad 0 \le t \le T$$

Numerically solving (1D) PDEs in MATLAB

- Boundary conditions are also written in an opaque way:

$$p(x,t,u) + q(x,t) f\left(x,t,u,\frac{\partial u}{\partial x}\right) = 0$$

- We know our system has no-flux conditions at either end: $\left.\frac{\partial u}{\partial x}\right|_{x=0} = \left.\frac{\partial u}{\partial x}\right|_{x=1} = 0$
- Plugging in for f at either endpoint lets us solve for p, q:

$$p(0,t,u) + q(0,t)\left(D\frac{\partial u}{\partial x}\right) = 0$$

$$p(1,t,u) + q(1,t)\left(D\frac{\partial u}{\partial x}\right) = 0$$

$$\Rightarrow$$

$$p(0,t,u) = 0 \qquad q(0,t) = 1$$

$$p(1,t,u) = 0 \qquad q(1,t) = 1$$

Coding it all up in MATLAB

- MATLAB has a cookie-cutter way of specifying all these conditions within three separate functions that will be run nested inside the larger pdepe function:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}, \qquad \left.\frac{\partial u}{\partial x}\right|_{x=0} = \left.\frac{\partial u}{\partial x}\right|_{x=1} = 0, \quad u(x,0) = \frac{10}{\sqrt{2\pi}}\exp\left(-\frac{(x-0.5)^2}{0.02}\right)$$

```
function [c,f,s] = pdefun(x,t,u,dudx)
c = 1;
f = D*dudx;
s = 0;
end
```

```
function u0 = pdeic(x)
u0 = 10/sqrt(2*pi).*exp(-(x-0.5).^2/0.02);
end
```

```
function [pl,ql,pr,qr] = pdebc(xl,ul,xr,ur,t)
pl = 0;
ql = 1;
pr = 0;
qr = 1;
end
```

Coding it all up in MATLAB

- The only thing left to do is specify our points to solve on in space and time:

```
x = linspace(0,1,100);
t = linspace(0,2,100);
```

- Then, we can run our full function with ICs, BCs, and space and time frame:

This value tells the system what type of geometric symmetry to employ (what coordinate system are you running over):

```
sol = pdepe(0,@pdefun,@pdeic,@pdcbc,x,t);
```

PDE function(s)

initial conditions

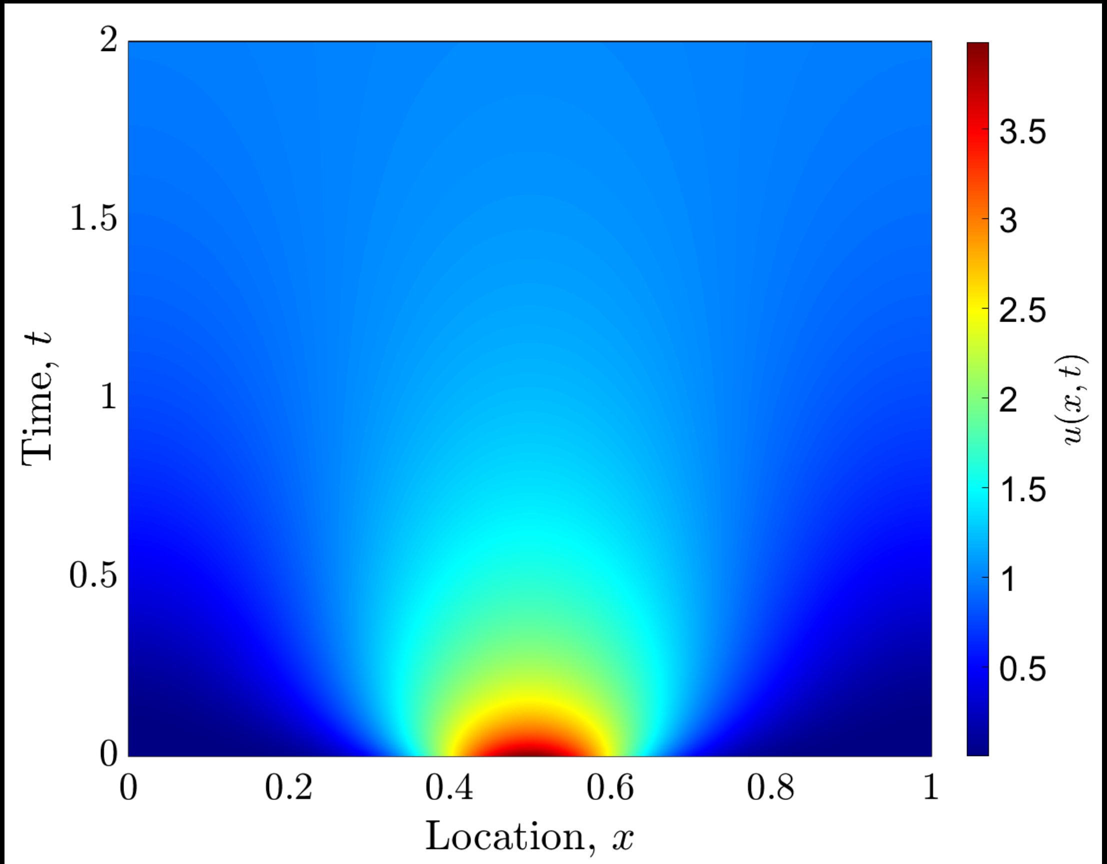
boundary conditions

space variable (discretized)

time variable (discretized)

```
0 if rectangular
1 if cylindrical
2 if spherical
```

Coding it all up in MATLAB



Alternative: turn PDEs into systems of ODEs

- For this approach, we will approximate space derivatives like we did earlier:

$$\frac{\partial u(x,t)}{\partial x} \approx \frac{u(x+\Delta x,t)\ - u(x-\Delta x,t)}{2\Delta t}$$

$$\frac{\partial^2 u(x,t)}{\partial x^2} \approx \frac{u(x+\Delta x,t) - 2u(x,t) + u(x-\Delta x,t)}{(\Delta x)^2}$$

- We need to now discretize space by splitting the interval $[0,1]$ into $N+1$ points equally spaced Δx apart, which lets us write

$$\Delta x = \frac{1-0}{N} = \frac{1}{N}, \qquad x_i = \frac{i}{N}, \text{for } i = 0,1,2,\ldots N$$

Alternative: turn PDEs into systems of ODEs

- If we only discretize in space (but not time), we can rewrite the PDE as:

$$\frac{\partial u(x,t)}{\partial t} \approx D\,\frac{u(x+\Delta x,t) - 2u(x,t) + u(x-\Delta x,t)}{(\Delta x)^2}$$

- Plugging in the space discretization gives us:

$$\Delta x = \frac{1-0}{N} = \frac{1}{N}, \qquad x_i = \frac{i}{N}, \text{for } i = 0,1,2,\ldots N$$

$$\Rightarrow \frac{du(x_i,t)}{dt} \approx D\,\frac{u(x_{i+1},t) - 2u(x_i,t) + u(x_{i-1},t)}{(\Delta x)^2}$$

Alternative: turn PDEs into systems of ODEs

- Using a shorthand notation for the space variable, we can write:

$$\frac{du_i}{dt} \approx D\frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2} = \frac{D}{(\Delta x)^2}u_{i+1} - \frac{2D}{(\Delta x)^2}u_i + \frac{D}{(\Delta x)^2}u_{i-1}$$

- Thus, we have a system of ODEs as follows:

$$\frac{du_0}{dt} = \frac{D}{(\Delta x)^2}u_1 - \frac{2D}{(\Delta x)^2}u_0 + \frac{D}{(\Delta x)^2}u_{-1}$$

$$\frac{du_1}{dt} = \frac{D}{(\Delta x)^2}u_2 - \frac{2D}{(\Delta x)^2}u_1 + \frac{D}{(\Delta x)^2}u_0$$

$$\vdots \qquad\qquad \vdots$$

$$\frac{du_{N-1}}{dt} = \frac{D}{(\Delta x)^2}u_N - \frac{2D}{(\Delta x)^2}u_{N-1} + \frac{D}{(\Delta x)^2}u_{N-2}$$

$$\frac{du_N}{dt} = \frac{D}{(\Delta x)^2}u_{N+1} - \frac{2D}{(\Delta x)^2}u_N + \frac{D}{(\Delta x)^2}u_{N-1}$$

What is going on with these points?

Alternative: turn PDEs into systems of ODEs

- To take care of these terms, we need to consider our boundary conditions:

$$\left.\frac{\partial u}{\partial x}\right|_{x=0} = \left.\frac{\partial u}{\partial x}\right|_{x=1} = 0, \qquad \Rightarrow \frac{\partial u_0}{\partial x} = \frac{\partial u_N}{\partial x} = 0$$

- Let's use our second order accurate definition of the derivative we found earlier:

$$f'(x) \approx \frac{f(x+\Delta x) - f(x-\Delta x)}{2(\Delta x)}$$

$$\frac{\partial u_0}{\partial x} \approx \frac{u_1 - u_{-1}}{2(\Delta x)} = 0 \qquad \Rightarrow \boxed{u_{-1} = u_1}$$

$$\frac{\partial u_N}{\partial x} \approx \frac{u_{N+1} - u_{N-1}}{2(\Delta x)} = 0 \qquad \Rightarrow \boxed{u_{N+1} = u_{N-1}}$$

Alternative: turn PDEs into systems of ODEs

- We can now substitute for our weird problem points u_{-1}, u_{N+1}:

$$
\begin{aligned}
\frac{du_0}{dt} &= \frac{2D}{(\Delta x)^2} u_1 - \frac{2D}{(\Delta x)^2} u_0 \\
\frac{du_1}{dt} &= \frac{D}{(\Delta x)^2} u_2 - \frac{2D}{(\Delta x)^2} u_1 + \frac{D}{(\Delta x)^2} u_0 \\
&\vdots \qquad\qquad\qquad \vdots \\
\frac{du_{N-1}}{dt} &= \frac{D}{(\Delta x)^2} u_N - \frac{2D}{(\Delta x)^2} u_{N-1} + \frac{D}{(\Delta x)^2} u_{N-2} \\
\frac{du_N}{dt} &= -\frac{2D}{(\Delta x)^2} u_N + \frac{2D}{(\Delta x)^2} u_{N-1}
\end{aligned}
$$

- Because each of these equations specifies the sliding value of u over time for each point in space, this approach is called the Method of Lines (MOL)

Method of Lines (MOL)

- All equations are linear in u_i, so we can write them in matrix form:

$$\frac{d}{dt}\begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_N \end{pmatrix} = \underbrace{\begin{pmatrix} -\frac{2D}{(\Delta x)^2} & \frac{2D}{(\Delta x)^2} & 0 & & 0 \\ \frac{D}{(\Delta x)^2} & -\frac{2D}{(\Delta x)^2} & \frac{D}{(\Delta x)^2} & \cdots & 0 \\ 0 & \frac{D}{(\Delta x)^2} & -\frac{2D}{(\Delta x)^2} & & 0 \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots \quad \frac{2D}{(\Delta x)^2} & -\frac{2D}{(\Delta x)^2} \end{pmatrix}}_{(N+1) \text{ x } (N+1) \text{ matrix}} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ u_N \end{pmatrix}$$

Method of Lines (MOL)

```
%Initialize values
n = 51;
x = linspace(0,1,n);
delx = (x(end)-x(1))/(n-1);
diff = 0.05;
```

```
%Creating update matrix (recursive formulas)
A = zeros(n,n);

for i = 1:n
    for j = 1:n
        if i == j
            A(i,j) = -2*diff/(delx^2);
        elseif i == j+1
            A(i,j) = diff/(delx^2);
        elseif i == j-1
            A(i,j) = diff/(delx^2);
        end
    end
end

A(1,2) = 2*diff/(delx^2);
A(end,end-1) = 2*diff/(delx^2);
```

Method of Lines (MOL)

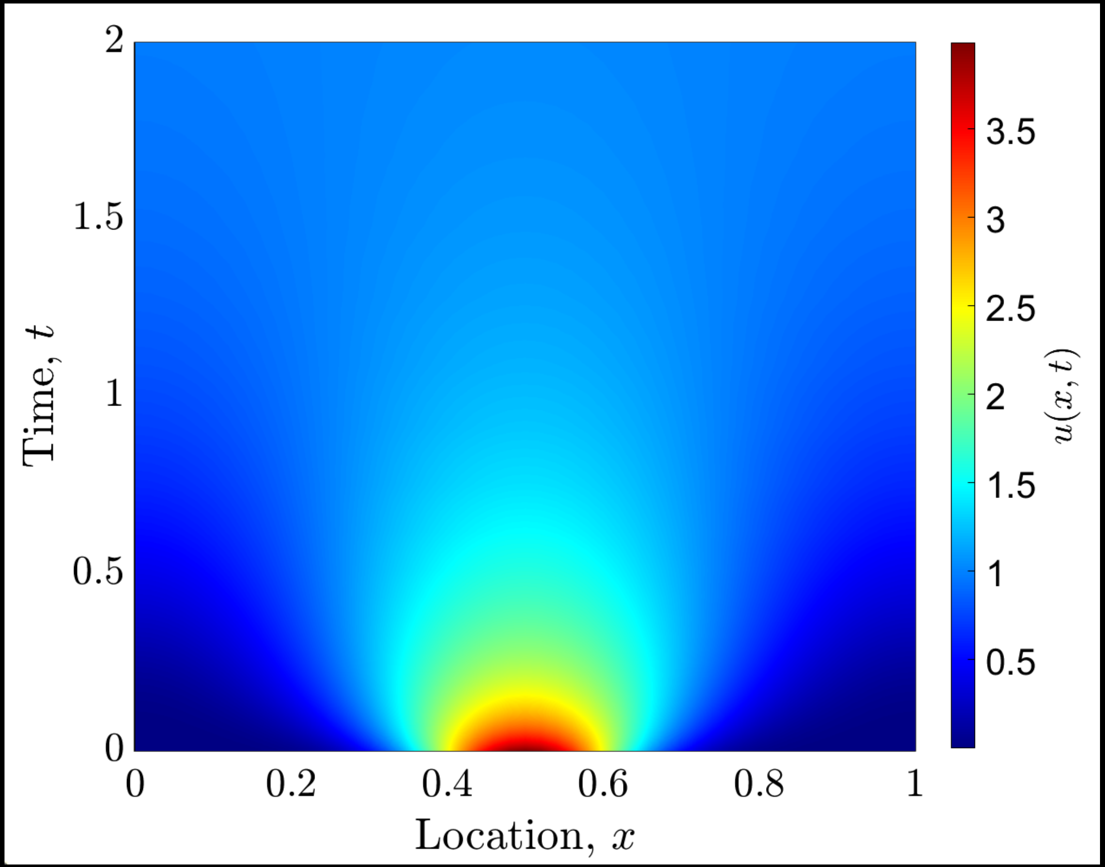
$N = 50$

$\Delta x = 0.02$

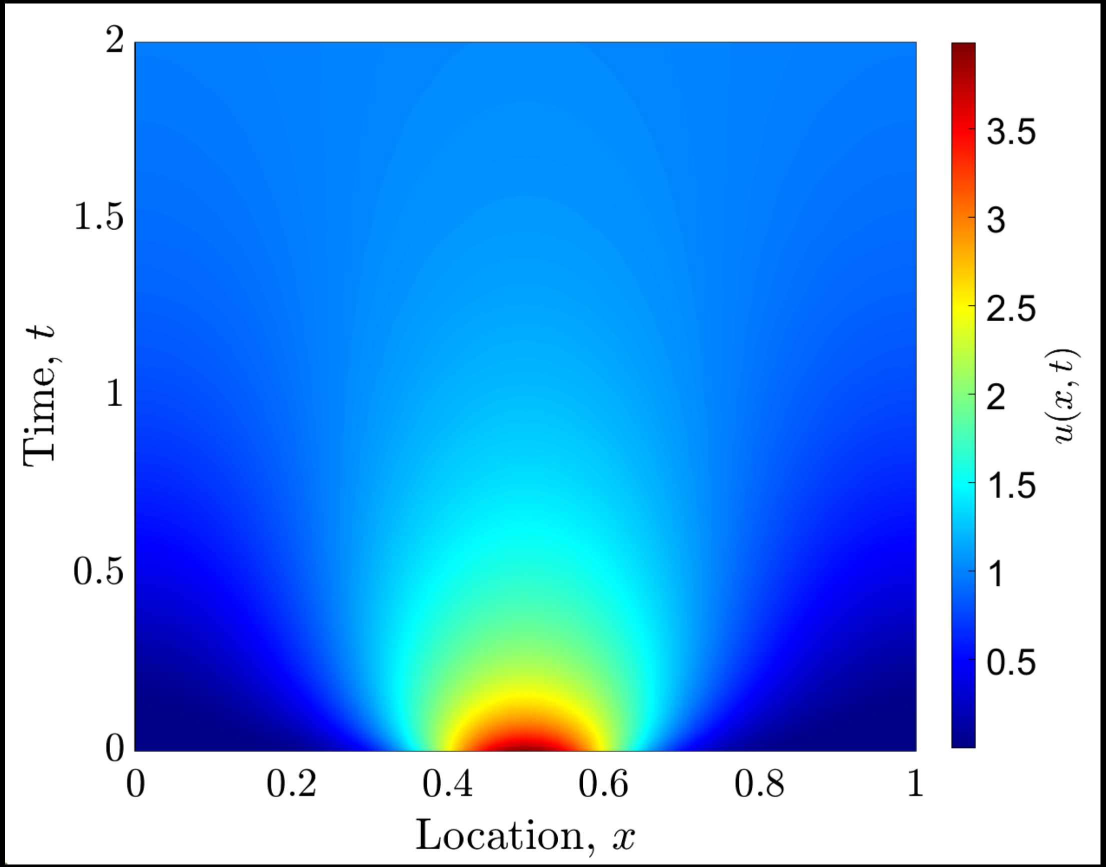
$D = 0.05$

```
u0 = 10/sqrt(2*pi).*exp(-(x-0.5).^2/0.02);

u = zeros(n,1);
[t,u] = ode23s(@(t,u) A*u,[0 10],u0);

figure;
for i = 1:size(t,1)
         plot(x,u(:,i),'LineWidth',2)
         xlabel('$x$', 'interpreter','latex’)
         ylabel('$u(x,t)$', 'interpreter','latex’)
         title(strcat('$t =
',num2str(round(t(i),3)),'$'),'interpreter','latex’)
         axis([0 1 0 5])
         ax=gca;
         ax.FontSize = 20;
         drawnow
end
```

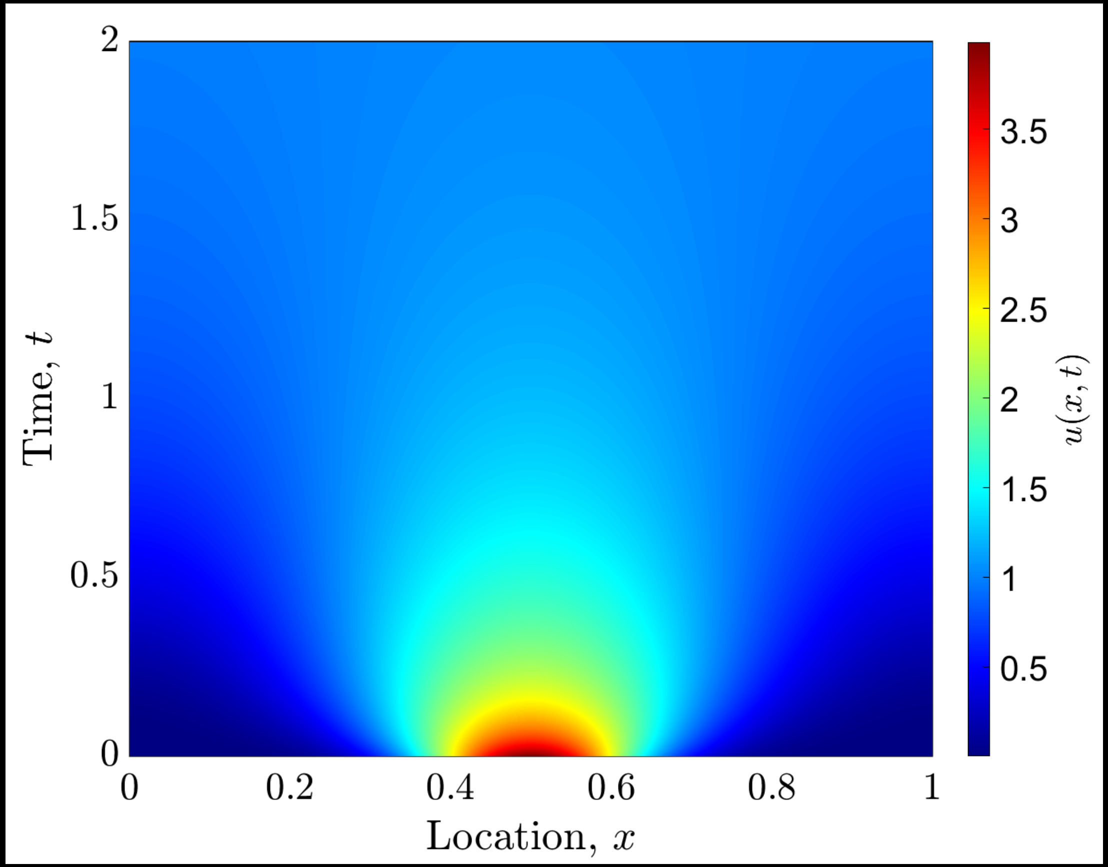
Method of Lines (MOL)



Comparison with MATLAB built-in PDE solver



Method of Lines (MOL)



MATLAB pdepe

Summary

- As compared to ODEs, solving PDEs numerically can be much more computationally expensive and mathematically painful

- Making some clever approximations for derivatives can enable numerical approaches to remain accurate up to arbitrary order (at the expense of computational cost)

- The Method of Lines is an alternative approach that effectively turns a PDE into a system of ODEs that can be numerically solved using `ode23s`