PHY604 Lecture 14

October 9, 2025

Today’s lecture: Curve fitting and PDEs

- Curve fitting

- Partial differential equations

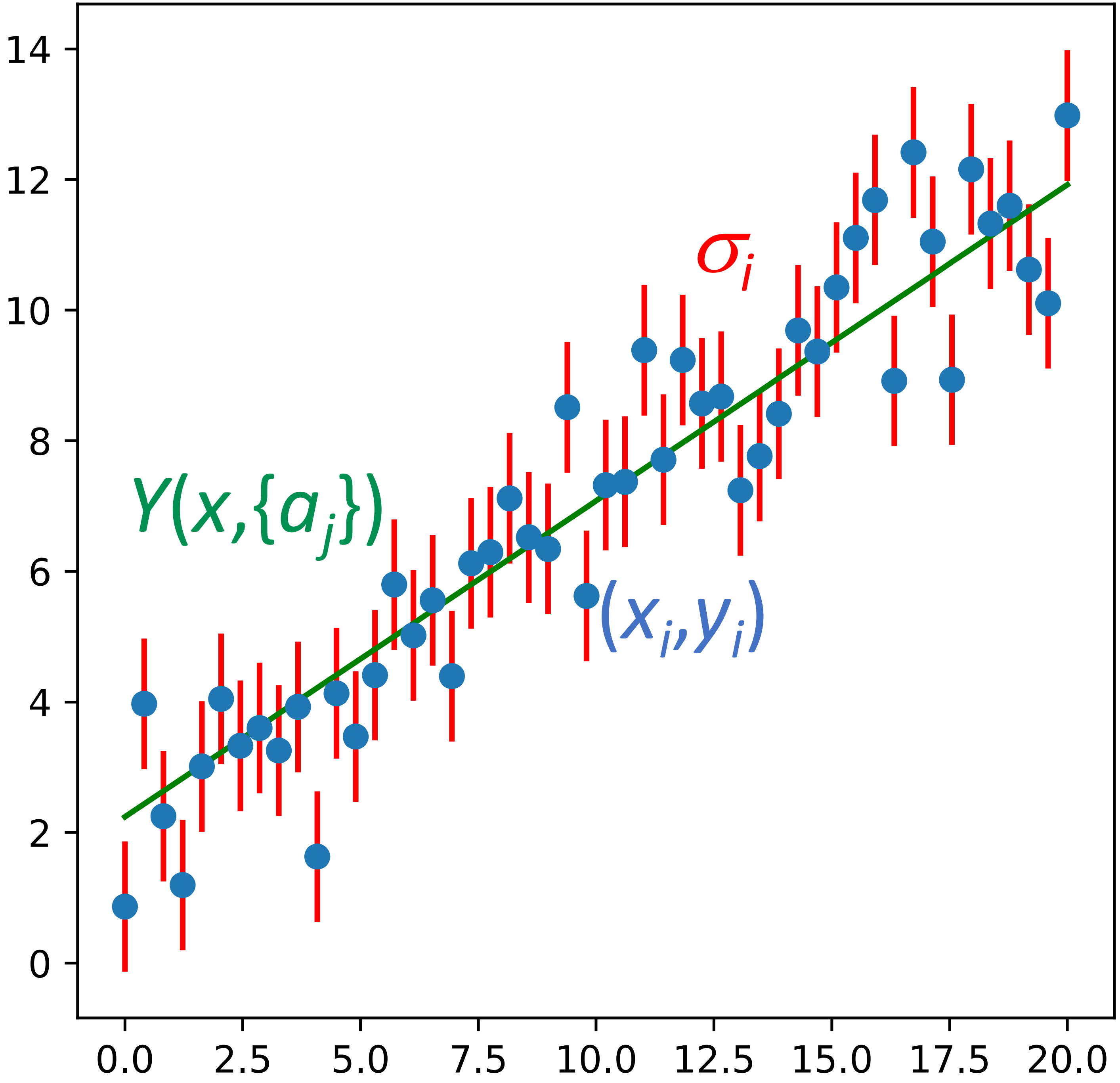
Fitting data

- We have discussed interpolation, now we'll talk about fitting
  - *Interpolation* seeks to fill in missing information in some small region of the whole dataset
  - *Fitting* a function to the data seeks to produce a model (guided by physical intuition) so you can learn more about the global behavior of your data

- Goal is to understand data by finding a simple function that best represents the data
  - Previous discussion on linear algebra and root finding comes into play

- We will follow Garcia (Sec. 5.1)
  - Big topic, we'll just look at the basics

Notation



General theory of fitting

- We have a dataset of N points (x_i, y_i)
- Would like to "fit" this dataset to a function $Y(x,\{a_j\})$
  - $\{a_j\}$ is a set of M adjustable parameters
  - Find the value of these parameters that minimizes the distance between data points and curve: $$\Delta_i = Y(x_i, \{a_j\}) - y_i$$
- Curve-fitting criteria: Minimize the sum of the squares

$$D(\{a_j\}) = \sum_{i=0}^{N-1} \Delta_i^2 = \sum_{i=0}^{N-1} [Y(x_i, \{a_j\}) - y_i]^2$$

- "Least squares fit"
  - Not the only way, but the most common

General theory of fitting

- Often data points have estimated error bars/confidence intervals σ_i
- Modify fit criterion to give less weight to points with the most error

$$\chi^2(\{a_j\}) = \sum_{i=0}^{N-1} \left(\frac{\Delta_i}{\sigma_i}\right)^2 = \sum_{i=0}^{N-1} \frac{[Y(x_i, \{a_j\}) - y_i]^2}{\sigma_i^2}$$

- χ^2 most used fitting function
  - Errors have a Gaussian distribution

- We will not discuss “validation” of curve fitted to data
  - i.e., probability that the data is described by a given curve

Linear regression

- Now that we have criteria for a good fit, we need to find $\{a_i\}$

- First consider the simplest example: fitting data with a straight line

$$Y(x_i, \{a_0, a_1\}) = a_0 + a_1 x$$

- Such that χ^2 is minimized:

$$\chi^2(a_0, a_1) = \sum_{i=0}^{N-1} \frac{[a_0 + a_1 x_i - y_i]^2}{\sigma_i^2}$$

Linear regression: Finding coefficients

- Minimize χ^2 with respect to coefficients:

$$\frac{\partial \chi^2}{\partial a_0} = 2 \sum_{i=0}^{N-1} \frac{a_0 + a_1 x_i - y_i}{\sigma_i^2} = 0, \qquad \frac{\partial \chi^2}{\partial a_0} = 2 \sum_{i=0}^{N-1} x_i \frac{a_0 + a_1 x_i - y_i}{\sigma_i^2} = 0$$

- We can write as:

$$a_0 S + a_1 \Sigma_x - \Sigma_y = 0, \qquad a_0 \Sigma_x + a_1 \Sigma_{x^2} - \Sigma_{xy} = 0$$

- Where coefficients are known:

$$S \equiv \sum_{i=0}^{N-1} \frac{1}{\sigma_i^2}, \quad \Sigma_x \equiv \sum_{i=0}^{N-1} \frac{x_i}{\sigma_i^2}, \quad \Sigma_y \equiv \sum_{i=0}^{N-1} \frac{y_i}{\sigma_i^2}, \quad \Sigma_{x^2} \equiv \sum_{i=0}^{N-1} \frac{x_i^2}{\sigma_i^2}, \quad \Sigma_{xy} \equiv \sum_{i=0}^{N-1} \frac{x_i y_i}{\sigma_i^2}$$

Linear regression: Finding coefficients

- Solving for a_0 and a_1:

$$a_0 = \frac{\Sigma_y \Sigma_{x^2} - \Sigma_x \Sigma_{xy}}{S\Sigma_{x^2} - (\Sigma_x)^2}, \qquad a_1 = \frac{S\Sigma_{xy} - \Sigma_y \Sigma_x}{S\Sigma_{x^2} - (\Sigma_x)^2}$$

- Note that if σ_i is constant, it will cancel out
- Now let's define an error bar for the curve-fitting parameter a_j

$$\sigma_{a_j}^2 = \sum_{i=0}^{N-1} \left(\frac{\partial a_j}{\partial y_i} \right)^2 \sigma_i^2$$

  - See: https://en.wikipedia.org/wiki/Propagation_of_uncertainty

Both independent of y_i

- For our linear case (after some algebra):

$$\sigma_{a_0} = \sqrt{\frac{\Sigma_{x^2}}{S\Sigma_{x^2} - (\Sigma_x)^2}}, \qquad \sigma_{a_1} = \sqrt{\frac{S}{S\Sigma_{x^2} - (\Sigma_x)^2}}$$

Linear regression: Errors in coefficients

- If error bars are constant:

$$\sigma_{a_0} = \frac{\sigma_0}{\sqrt{N}}\sqrt{\frac{\langle x^2\rangle}{\langle x^2\rangle - \langle x\rangle^2}}, \qquad \sigma_{a_1} = \frac{\sigma_0}{\sqrt{N}}\sqrt{\frac{1}{\langle x^2\rangle - \langle x\rangle^2}}$$

variance

- Where:

$$\langle x\rangle = \frac{1}{N}\sum_{i=0}^{N-1} x_i, \quad \langle x^2\rangle = \frac{1}{N}\sum_{i=0}^{N-1} x_i^2$$

- If data does not have error bars, we can estimate σ_0 from the sample variance (https://en.wikipedia.org/wiki/Variance)

Sample std deviation

$$\sigma_0 \simeq s^2 = \frac{1}{N-2}\sum_{i=0}^{N-1}\left[y_i - (a_0 + a_1 x_i)\right]^2$$

N-2 since already extracted a0 and a1 from data

Nonlinear regression (with two variables)

- We have been discussing fitting a linear function, but many nonlinear curve-fitting problems can be transformed into linear problems

- Examples: $Z(x, \{\alpha, \beta\}) = \alpha e^{\beta x}$

- Rewrite with: $\ln Z = Y, \quad \ln \alpha = a_0, \quad \beta = a_1$

- Result: $Y = a_0 + a_1 x$

General least squares fit

- No analytic solution to general least squares problem, but can solve numerically
- Generalize to functions of the form:

$$Y(x_i, \{a_j\}) = a_0 Y_0(x) + a_1 Y_1(x) + \cdots + a_{M-1} Y_{M-1}(x) = \sum_{j=0}^{M-1} a_j Y_j(x)$$

- Now minimize χ^2: $$\frac{\partial \chi^2}{\partial \{a_j\}} = \frac{\partial}{\partial \{a_j\}} \sum_{i=0}^{N-1} \frac{1}{\sigma_i^2} \left[\sum_{k=0}^{M-1} a_k Y_k(x_i) - y_i \right]^2 = 0$$

$$= \sum_{i=0}^{N-1} \frac{Y_j(x_i)}{\sigma_i^2} \left[\sum_{k=0}^{M-1} a_k Y_k(x_i) - y_i \right] = 0$$

General least-squares fit

- From previous slide, we have:

$$\sum_{i=0}^{N-1} \sum_{k=0}^{M-1} \frac{Y_j(x_i)Y_k(x_i)}{\sigma_i^2} a_k = \sum_{i=0}^{N-1} \frac{Y_j(x_i)y_i}{\sigma_i^2}$$

- Set of *j* equations known as normal equations of the least-squares problem (*Y*'s may be nonlinear, but linear in *a*'s)
- Define design matrix with elements $A_{ij} = Y_j(x_i)/\sigma_i$:

$$\mathbf{A} = \begin{bmatrix} \frac{Y_0(x_0)}{\sigma_0} & \frac{Y_1(x_0)}{\sigma_0} & \cdots \\ \frac{Y_0(x_1)}{\sigma_1} & \frac{Y_1(x_1)}{\sigma_1} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

- Only depends on independent variables (not y_i)

General least-squares fit

- With design matrix, we can rewrite: $$\sum_{i=0}^{N-1}\sum_{k=0}^{M-1}\frac{Y_j(x_i)Y_k(x_i)}{\sigma_i^2}a_k=\sum_{i=0}^{N-1}\frac{Y_j(x_i)y_i}{\sigma_i^2}$$

- As: $$\sum_{i=0}^{N-1}\sum_{k=0}^{M-1}A_{ij}A_{ik}a_k=\sum_{i=0}^{N-1}A_{ij}\frac{y_i}{\sigma_i}\implies(\mathbf{A}^{\mathrm{T}}\mathbf{A})\mathbf{a}=\mathbf{A}^{\mathrm{T}}\mathbf{b}$$
  - Where $b_i=y_i/\sigma_i$

- Thus: $\mathbf{a}=(\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}$

- Or, we can solve for **a** via Gaussian elimination

Goodness of fit

- Usually, we have *N* >> *M*, the number of data points is much greater than the number of fitting variables
- Given the error bars, how likely is it that the curve actually describes the data?
- Rule of thumb: If the fit is good, on average the difference should be approximately equal to the error bars

$$|y_i - Y(x_i)| \simeq \sigma_i$$

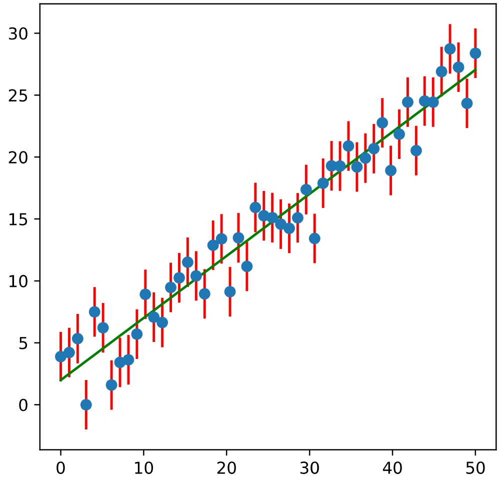
- Plugging in gives χ^2 equal to N. Since we know we can have a perfect fit for M=N, we postulate:

$$\chi^2 \simeq N - M$$

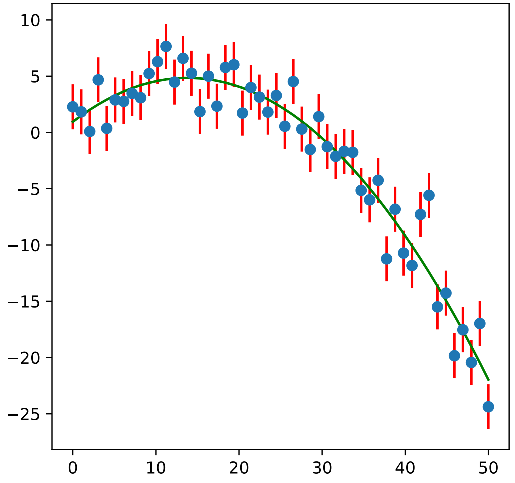
- If $\chi^2 \gg N - M$, probably not an appropriate function (or too small error bars
- If $\chi^2 \ll N - M$, fit is too good, error bars may be too large

Least squares fitting example:

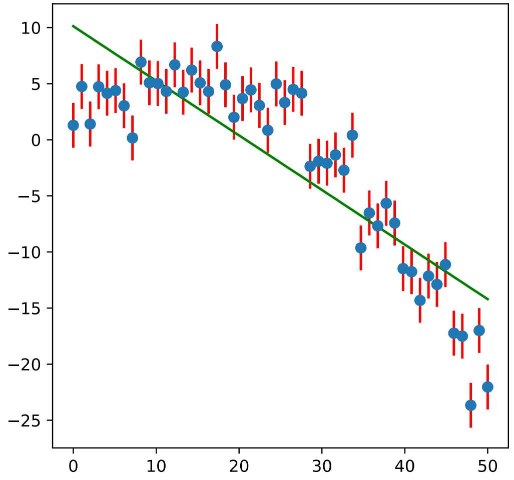
Linear regression, linear function



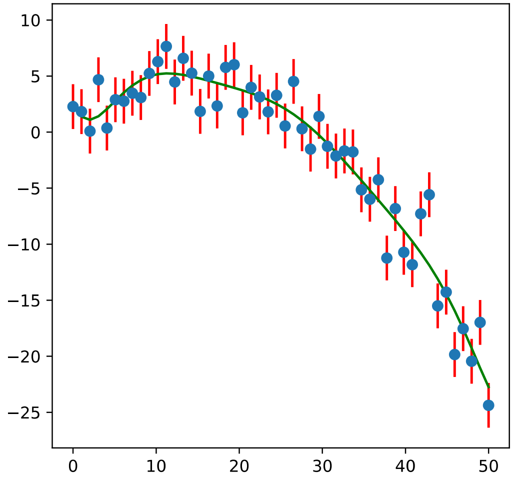
Polynomial regression (order 2), quadratic function



Linear regression, quadratic function



Polynomial regression (order 10), quadratic function



Comments on general least squares

- In the example, we used polynomials as our functions, but can use linear combinations of any functions we would like

- We choose functions strategically to get the best least squares fit
  - Often choosing orthogonal basis functions in the range of the fit will produce better fits

- The matrix $\mathbf{A}^T\mathbf{A}$ is notoriously ill conditioned especially for increased number of basis functions
  - Gaussian substitution will have problems solving (numpy solve uses singular-value decomposition)

- Procedure can be generalized if we also have errors in *x*

Nonlinear least-squares fitting

- Even in the polynomial case, we were using linear combinations of functions
- We can also directly fit a function whose parameters enter nonlinearly
- Consider the function: $f(a_0, a_1) = a_0 e^{a_1 x}$
- Want to minimize: $Q \equiv \sum_{i=1}^{N} (y_i - a_0 e^{a_1 x_i})^2$
- Take derivatives:

$$f_0 = \frac{\partial Q}{\partial a_0} = \sum_{i=1}^{N} e^{a_1 x_i} (a_0 e^{a_1 x_i} - y_i) = 0,$$

$$f_1 = \frac{\partial Q}{\partial a_1} = \sum_{i=1}^{N} x_i e^{a_1 x_i} (a_0 e^{a_1 x_i} - y_i) = 0$$

Nonlinear least-squares fitting

- Produces a nonlinear system—we can use the multivariate root-finding techniques we learned earlier:
  - Compute the Jacobian
  - Take an initial guess for unknown coefficients
  - Use Newton-Raphson techniques to compute the correction:

$$\mathbf{a}_1 = \mathbf{a}_0 - \mathbf{J}^{-1}\mathbf{f}$$

  - Iterate

- Can be very difficult to converge, and highly dependent on the initial guess

Fitting packages

- Fitting is a very sensitive procedure—especially for nonlinear cases
- Lots of minimization packages exist that offer robust fitting procedures

- MINUIT2: the standard package in high-energy physics (Python version: PyMinuit and Iminuit)
- MINPACK: Fortran library for solving least squares problems—this is what is used under the hood for the built in SciPy least squares routine
    - http://www.netlib.org/minpack/
- SciPy optimize: https://docs.scipy.org/doc/scipy/reference/optimize.html

Today's lecture:
Curve fitting and PDEs

- Curve fitting
- Partial differential equations

Partial differential equations (Garcia Chs. 6-9)

- Previously, we studied ordinary differential equations
- Much of physics is involved in solving partial differential equations
  - Schrodinger equation in Quantum mechanics
  - Maxwell's equations in electricity and magnetism
  - Wave equation in optics and acoustics
- For ODEs we developed general methods to solve a variety of problems, e.g., 4th order Runge-Kutta
- For PDEs, we first classify the type of equation, that will tell us what method to use

Examples of PDE types

- Parabolic equations
  - E.g., Time-dependent Schrodinger equation, 1D diffusion equation
  - Consider the Fourier equation with temperature *T* and thermal diffusion coefficient κ:

$$\frac{\partial T(x,t)}{\partial t} = \kappa \frac{\partial^2 T(x,t)}{\partial x^2}$$

- Hyperbolic equations
  - E.g., 1D wave equation with amplitude *A* and speed *c*:

$$\frac{\partial^2 A(x,t)}{\partial t^2} = c^2 \frac{\partial^2 A(x,t)}{\partial x^2}$$

- Elliptic equations
  - E.g., Poisson equation:

$$\frac{\partial^2 \Phi}{\partial x^2} + \frac{\partial^2 \Phi}{\partial y^2} = -\frac{1}{\epsilon_0}\rho(x,y)$$

General classification of PDEs

- Consider a general PDE of two independent variables:

$$a\frac{\partial^2 A}{\partial x^2} + b\frac{\partial^2 A}{\partial x \partial y} + c\frac{\partial^2 A}{\partial y^2} + d\frac{\partial A}{\partial x} + e\frac{\partial A}{\partial y} + fA(x,y) + g = 0$$

- Hyperbolic if: $b^2 - 4ac > 0$

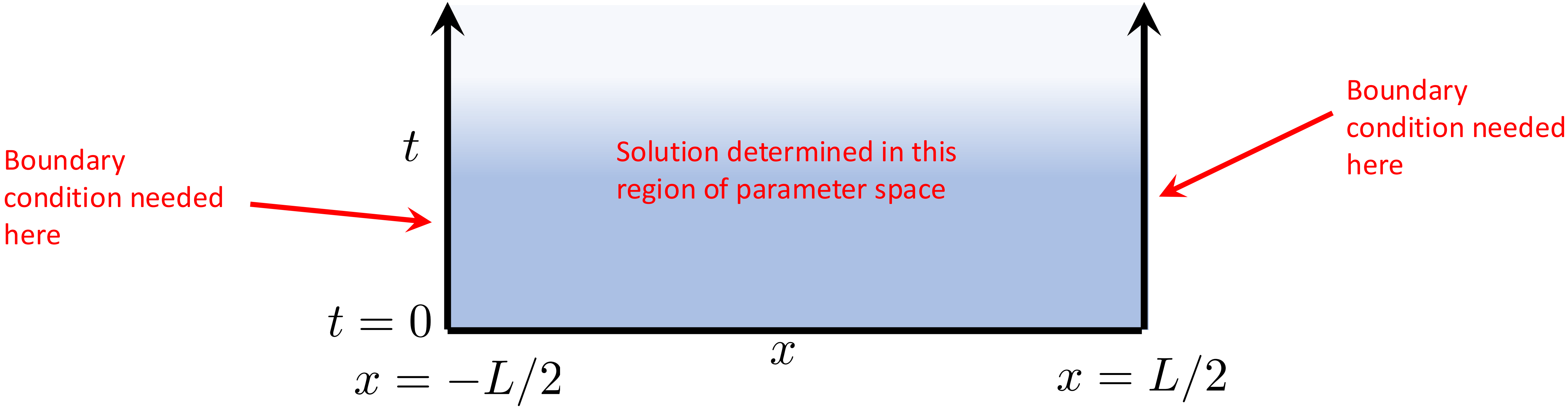
- Parabolic if: $b^2 - 4ac = 0$

- Elliptic if: $b^2 - 4ac < 0$

- Most problems involve hybrid systems, including multiple types

Initial value problems

- Diffusion and wave equations usually solved as initial value problems
  - Diffusion: Given initial temperature distribution, find temperature distribution at a later time
  - Wave: Start with initial amplitude and velocity of wave pulse and find the wave pulse at a later time
- Need to specify initial conditions as well as boundary conditions



Types of boundary conditions

- Dirichlet boundary conditions: Specify the solution on boundary
  - E.g., fix the temperature at the boundaries:

$$T(x=-L/2,t)=T_a, \quad T(x=L/2,t)=T_b$$

- Neumann boundary conditions: Specify the derivative on the boundary
  - E.g., "insulated" boundaries

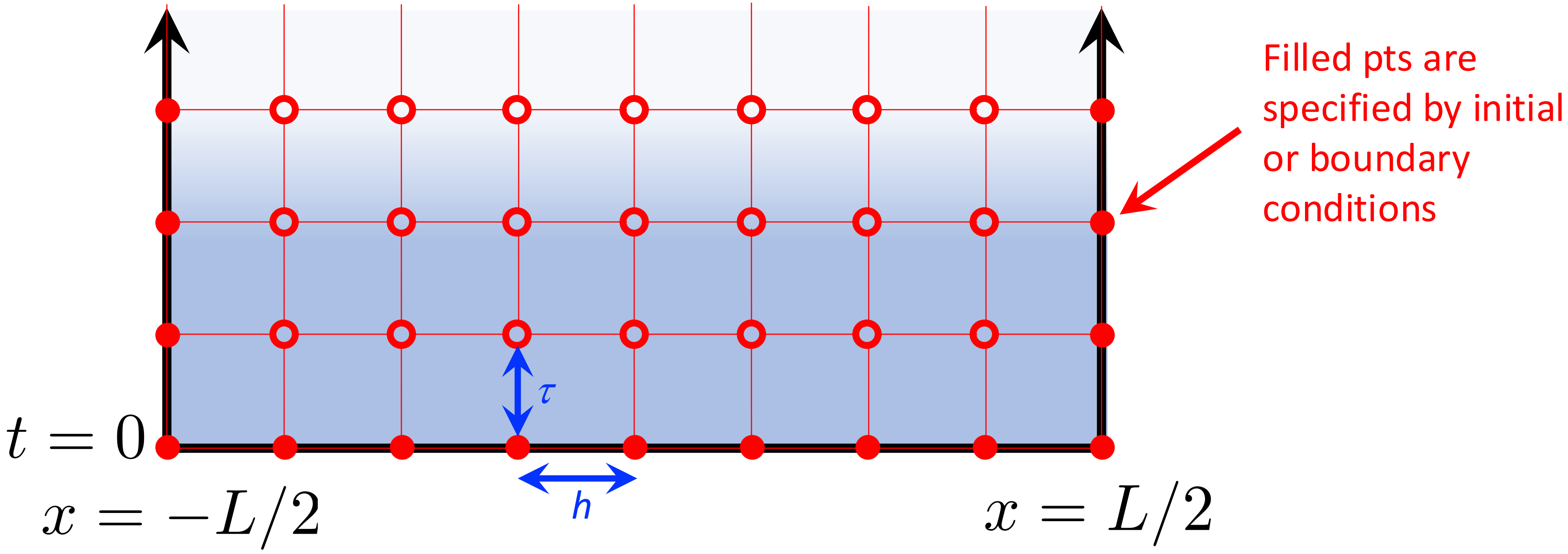
$$-\kappa \left.\frac{dT}{dx}\right|_{x=-L/2} = F_a = 0, \quad -\kappa \left.\frac{dT}{dx}\right|_{x=L/2} = F_b = 0$$

- Periodic boundary conditions: Equate the functions at both ends

$$T(x=-L/2,t)=T(x=L/2,t), \quad \left.\frac{dT}{dx}\right|_{x=-L/2} = \left.\frac{dT}{dx}\right|_{x=L/2}$$

Marching methods for initial value problems

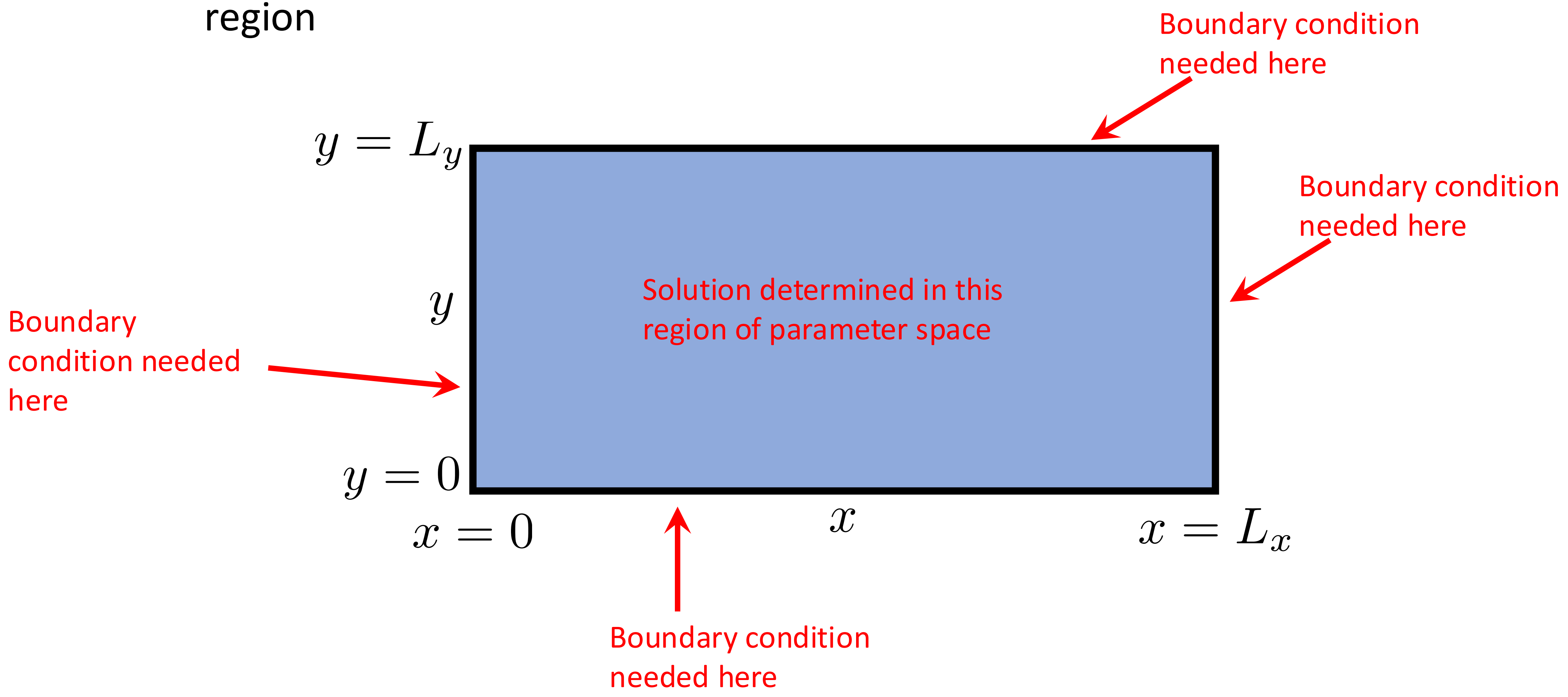
- We first must discretize in time and space:



- Start from the initial condition, move forward in time one timestep at a time

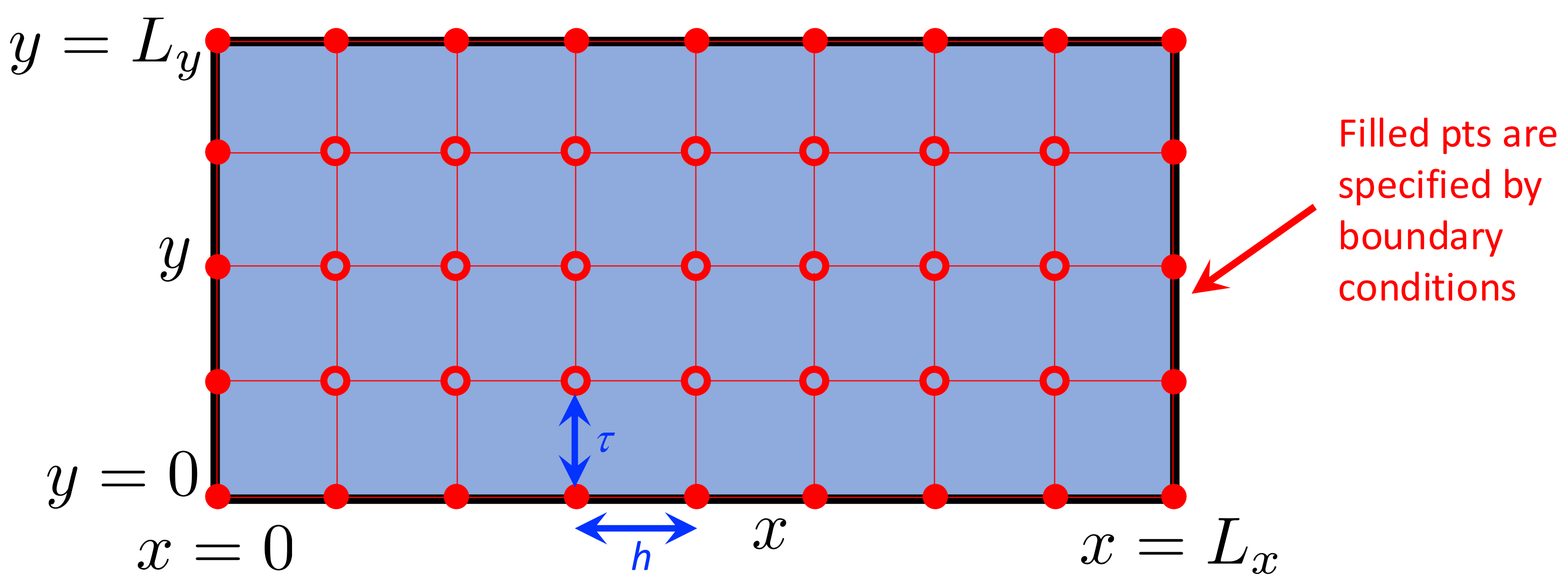
Boundary value problems

- All boundary values are specified at the outset
  - E.g., Laplace's equation in electrostatics, potential fixed on for sides of spatial region



Jury methods for boundary value problems

- Discretize in space:



- Potential in interior is influenced by all the boundary points, reconciles all the constraints imposed by boundaries

Diffusion equation with FTCS

- Diffusion equation: $$\frac{\partial T(x,t)}{\partial t} = \kappa \frac{\partial^2 T(x,t)}{\partial x^2}$$
- Discretize in space and time: $T_i^n = T(x_i, t_n)$
  - $x_i = i\,h - L/2$ and $t_n = n\,\tau$
  - Take spatial boundary points as $i = 0$ and $i = N$-1, so h = $L/(N$-1)
- Discretize time derivative with forward difference:
$$\frac{\partial T(x,t)}{\partial t} \to \frac{T_i^{n+1} - T_i^n}{\tau}$$
- Discretize spatial derivative using central difference:
$$\frac{\partial^2 T(x,t)}{\partial x^2} \to \frac{T_{i+1}^n + T_{i-1}^n - 2T_i^n}{h^2}$$

Diffusion equation with FTCS

- Now the discretized PDE is:

$$\frac{T_i^{n+1} - T_i^n}{\tau} = \kappa \frac{T_{i+1}^n + T_{i-1}^n - 2T_i^n}{h^2}$$

- And temperature at future time is:

$$T_i^{n+1} = T_i^n + \frac{\kappa\tau}{h^2}(T_{i+1}^n + T_{i-1}^n - 2T_i^n)$$

- Explicit: Everything that depends on previous timestep *n* is on RHS

- Discretization is reminiscent of Euler's method for ODEs

Numerical stability of FTCS method

- The numerical stability of the solution depends on the timestep
- Consider initial conditions of a delta-function peak in *T* located at *N*/2
  - Discrete approximation, $T(x=N/2, t_0) = 1/h$
- Can show from analytical solutions to this problem that the delta function will spread into a Gaussian with width:

$$\sigma(t) = \sqrt{2\kappa t}$$

- Thus, if t_σ is the time it takes for σ to increase by one grid spacing:

$$t_\sigma = \frac{h^2}{2\kappa}$$

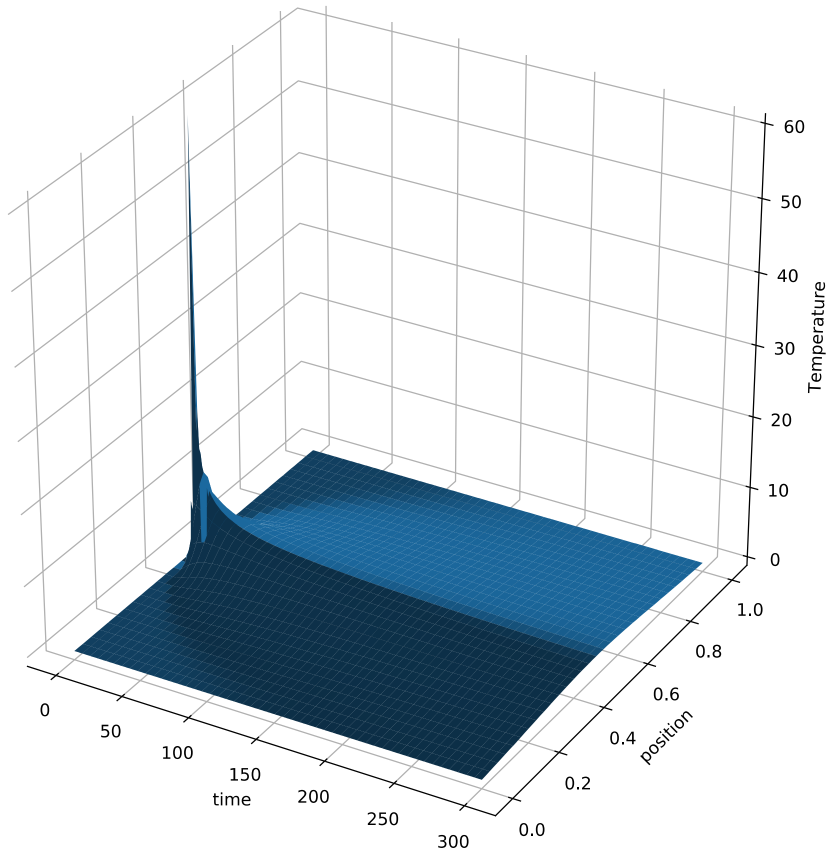
- Then:

$$T_i^{n+1} = T_i^n + \frac{\tau}{2t_\sigma}(T_{i+1}^n + T_{i-1}^n - 2T_i^n)$$

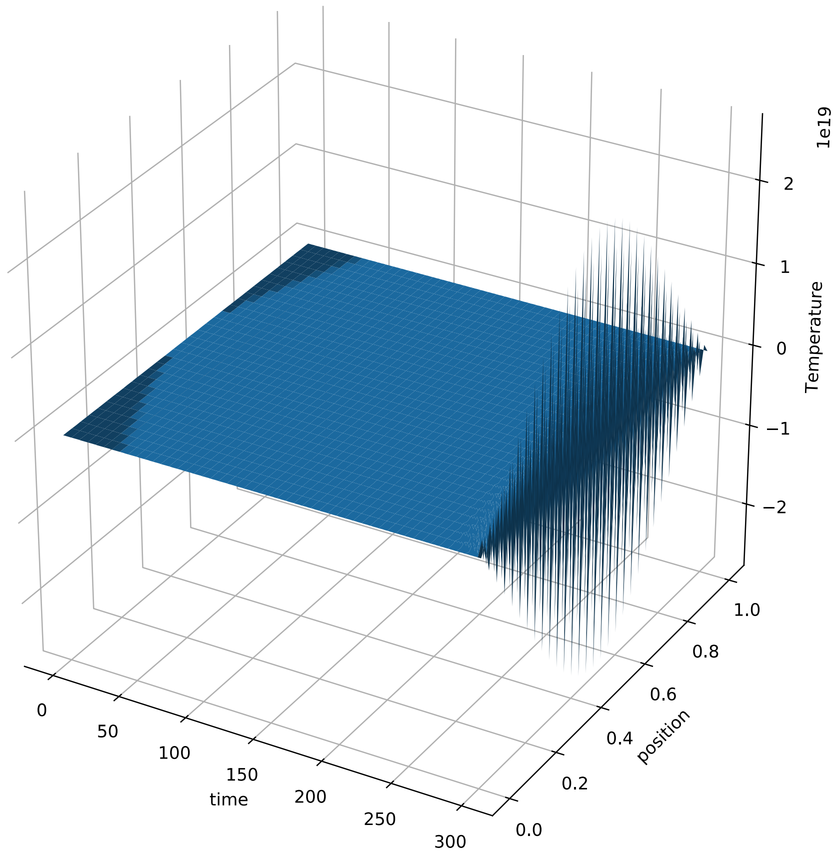
- Should not use a timestep much larger than t_σ

FCTS method on diffusion equation

Numerically stable: τ = 1e-4



Numerically stable: τ = 1.5e-4



After class tasks

- Homework 3 posted due Oct. 22

- Readings
  - Linear regression:
    - Wikipedia page on varience
    - Wikipedia page on propagation of errors
    - Garcia Sec. 5.1
  - PDEs
    - Garcia Chapters 6 and 7