Recap: Lecture 3

- Solved the *diffusion equation* for the heat spike problem using the *forward time centered space* (FTCS) method.
  - $T_i^{n+1} = T_i^n + f\left(T_{i-1}^n - 2T_i^n + T_{i+1}^n\right)$, where $f = \kappa\tau/h^2$
  - Works pretty well for diffusion!
- Represented the scheme in matrix form.
  - Multiple time updates become powers of the update matrix, $\mathbf{A}$: $\mathbf{T}^n = \mathbf{A}^{n-1}\mathbf{T}^1$.
- Analyzing the eigenvalues of $\mathbf{A}$ allowed us to understand the behavior of algorithms (including their *stability*) (computer lab):
  - $|\lambda_{\mathbf{max}}|$ tells us about stability, and the sign of eigenvalues allows us to reason about oscillatory behavior.

Assignment is live

- Available here: https://canvas.sydney.edu.au/courses/55271/assignments/530617
- Due Week 13, but some of the questions require a bit of thought, so I would recommend getting started as soon as you can.
- Questions 1,2: Two-body problem.
  - (Which interested students could extend to three if interested!)
- Question 3: Two-dimensional diffusion.
- Please submit via Gradescope, where you will *need to annotate your answers to the corresponding questions that they answer*

***Computational Physics*: Lecture 5 🌊**

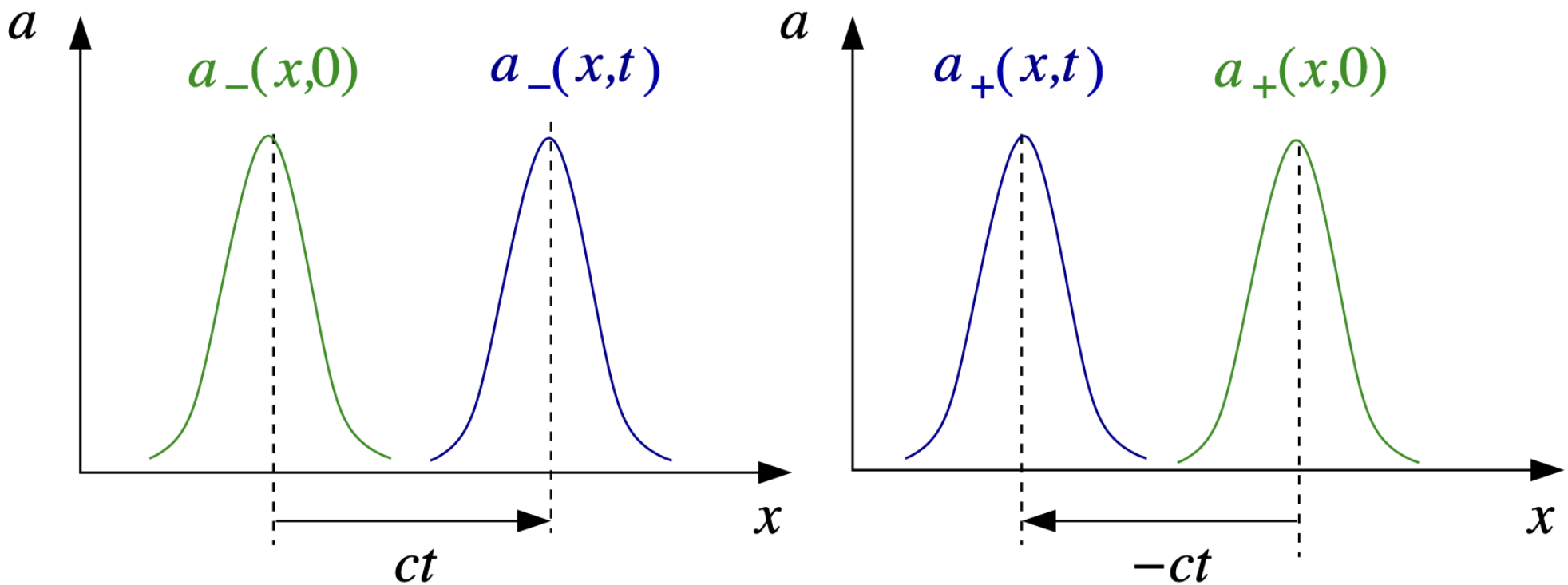
- The *wave* and *advection* equations 🌊
  - Describe propagation of an amplitude profile.
- The Forward Time Centered Space (FTCS) method for advection
  - *Unstable for all time steps* τ! 😳
- *von Neumann analysis* for assessing algorithm stability.
- *Lax method* for advection. 👨‍👨‍👦
  - Stabilizes by adding diffusion.
  - Courant-Friedrichs-Lewy (CFL) condition for stability.
  - Can yield *worse* performance with low τ 😲

The *wave equation*

- $\frac{\partial^2 a}{\partial t^2} = c^2 \frac{\partial^2 a}{\partial x^2}$.
  - Position x, time t, constant c.
  - Describes the *propagation of a disturbance* in x at speed c.
- Initial Condition: $f(x) = a_{\pm}(x, 0)$.
- There are two basic analytic solutions: $a_{\pm}(x, t) = f(x \pm ct)$
  - for some arbitrary function, f.
- 📖 **Exercise** (*easy*): Show that these two solutions satisfy the PDE.

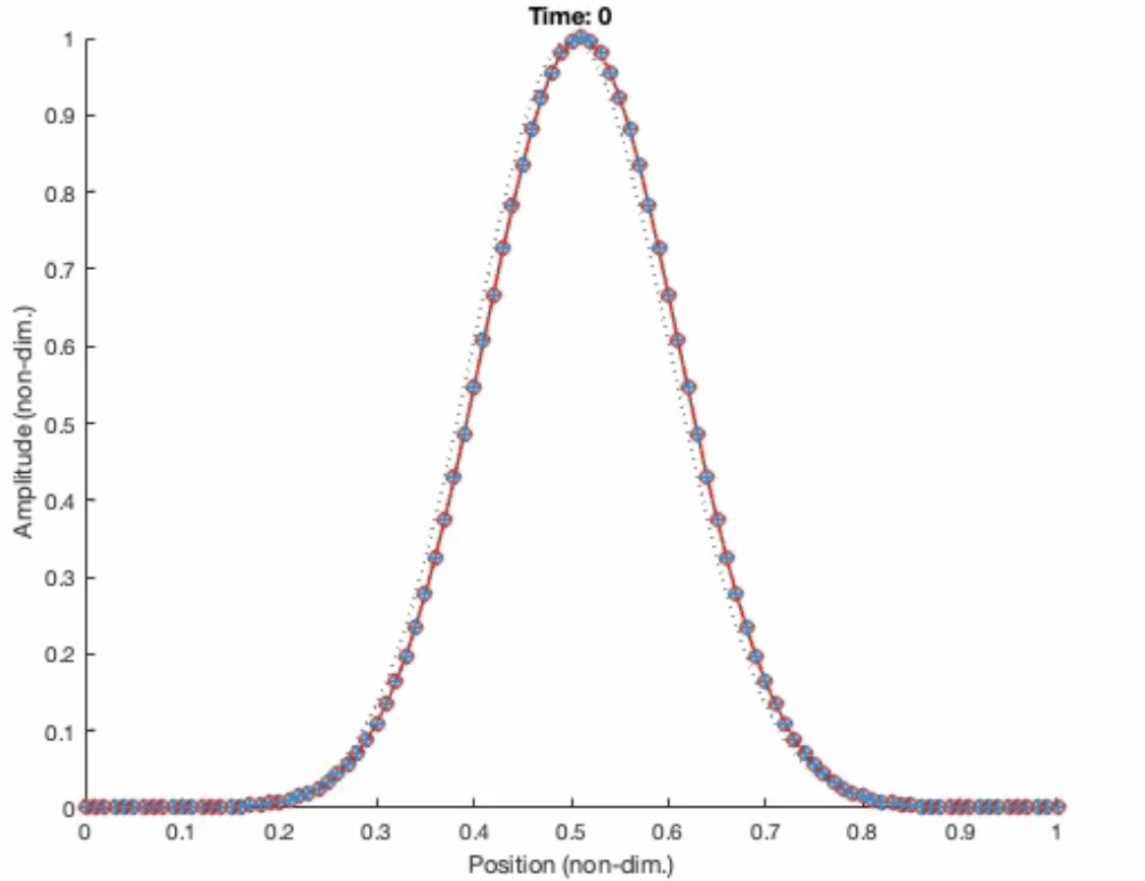
Over time, the initial profile *shifts* to the left/right $(+/-)$ in x:

$a_{\pm}(x,t) = f(x \pm ct)$.



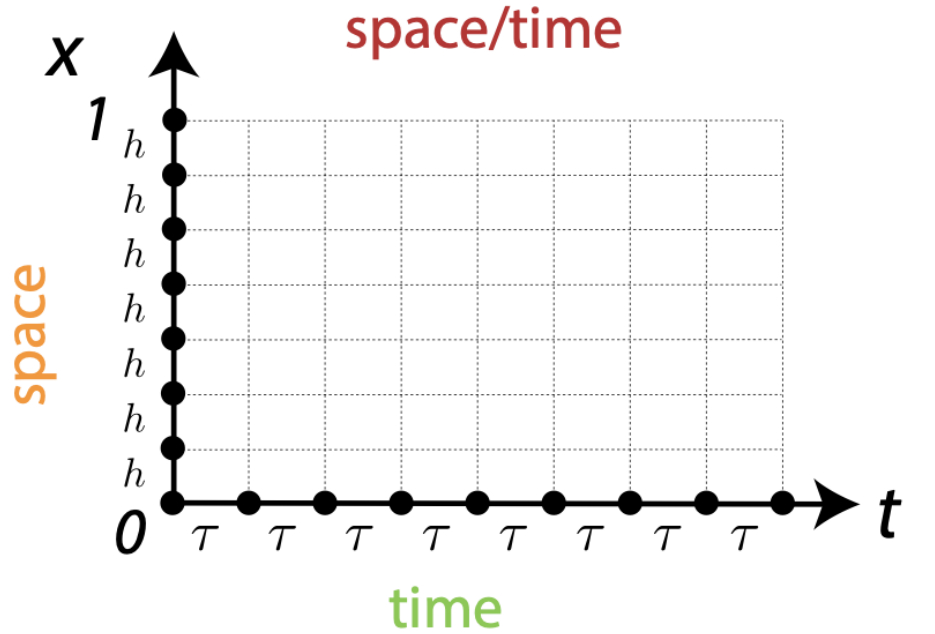
The Advection Equations

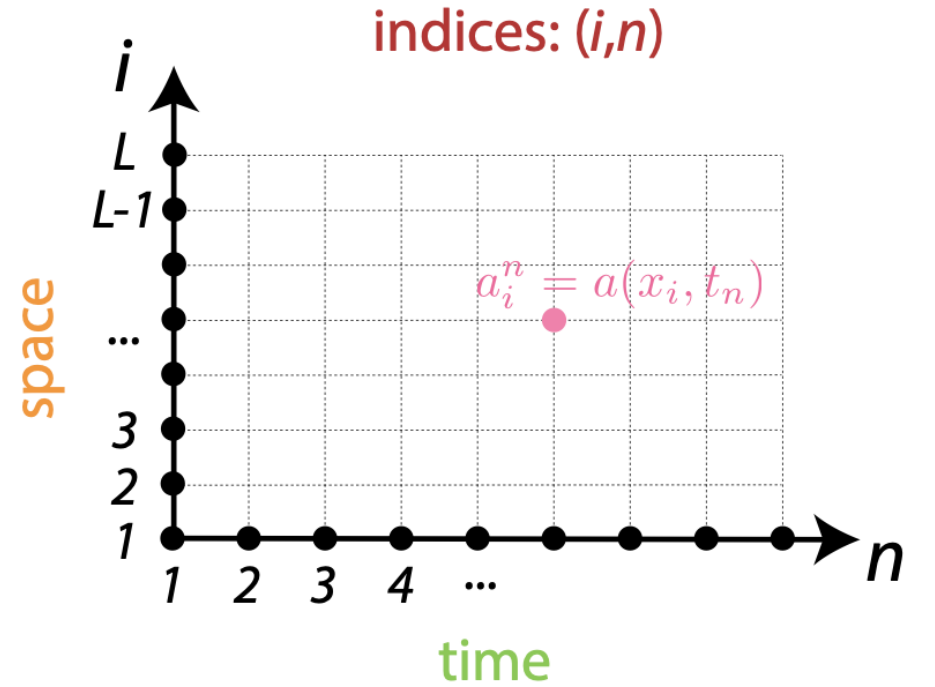
- The wave equation can be factorized as $\left(\frac{\partial}{\partial t} - c\frac{\partial}{\partial x}\right)\left(\frac{\partial}{\partial t} + c\frac{\partial}{\partial x}\right) a = 0.$
- These two operators define the *linear advection equations* (wave speed $c > 0$):
  - $\frac{\partial a_+}{\partial t} = c\frac{\partial a_+}{\partial x}$ with solution: $a_+ = f(x + ct)$.
  - $\frac{\partial a_-}{\partial t} = -c\frac{\partial a_-}{\partial x}$ with solution: $a_- = f(x - ct)$.
- *Simpler* since each equation describes propagation in one direction
  - We'll focus on $\frac{\partial a}{\partial t} = -c\frac{\partial a}{\partial x}$ 😋
  - ❓ Is this a left-moving or right-moving wave?
- *FYI*: We could easily 'remove' c by rescaling time/space with the choice $t_s = L_s/c$, but let's keep it 😋



***Reminder*: Discretizing space and time**

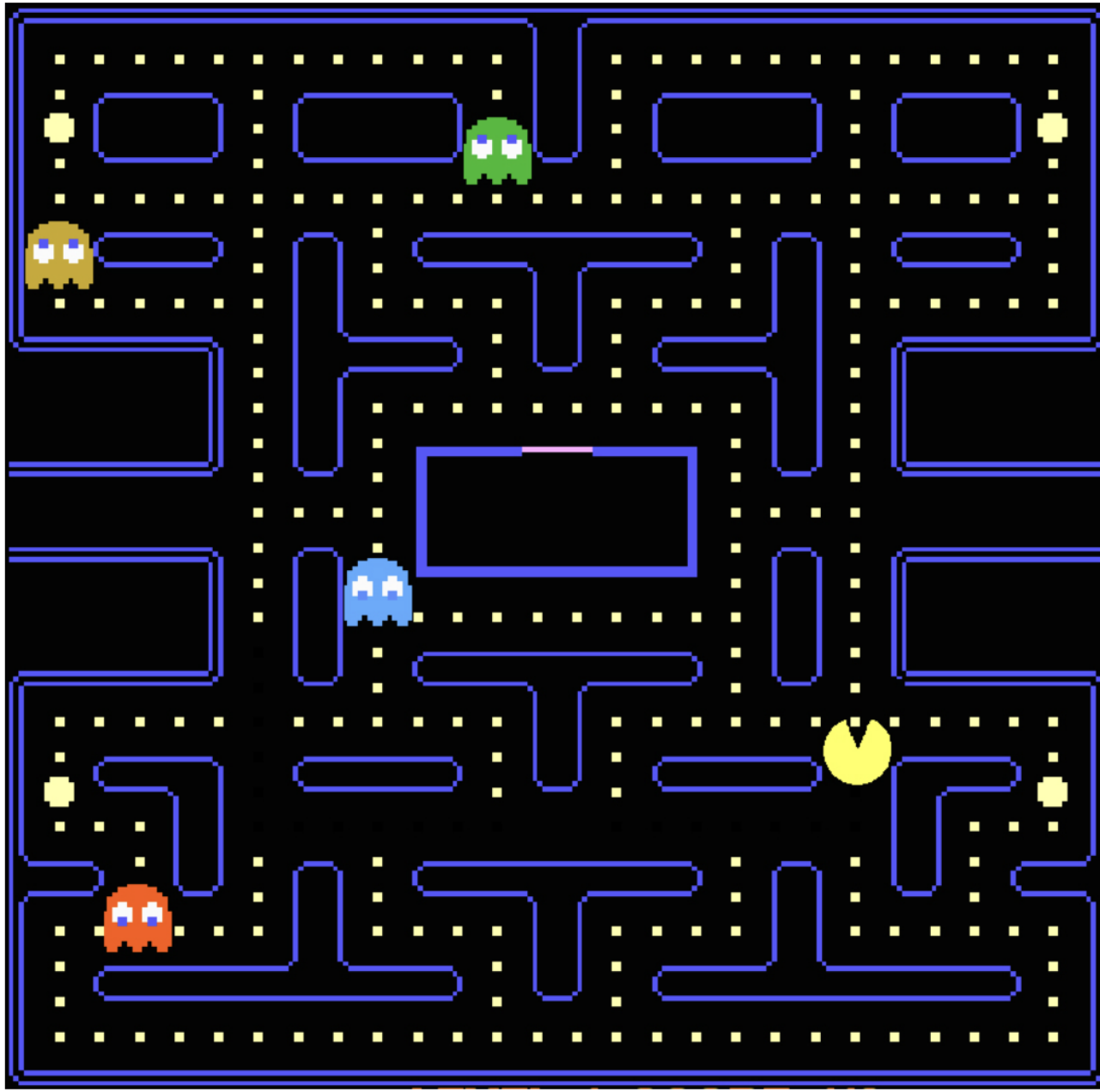
- *Reminder* (from last week): We solve on a *discrete grid* in space and time:
  - $x_i = (i-1)h$ with $h = (L-1)^{-1}$ for $i = 1, 2, \ldots, L$.
  - $t_n = (n-1)\tau$ with $n = 1, 2, \ldots$ (so $t_1 = 0$).
  - Using the notation: $a_i^n = a(x_i, t_n)$.





Periodic Boundary Conditions

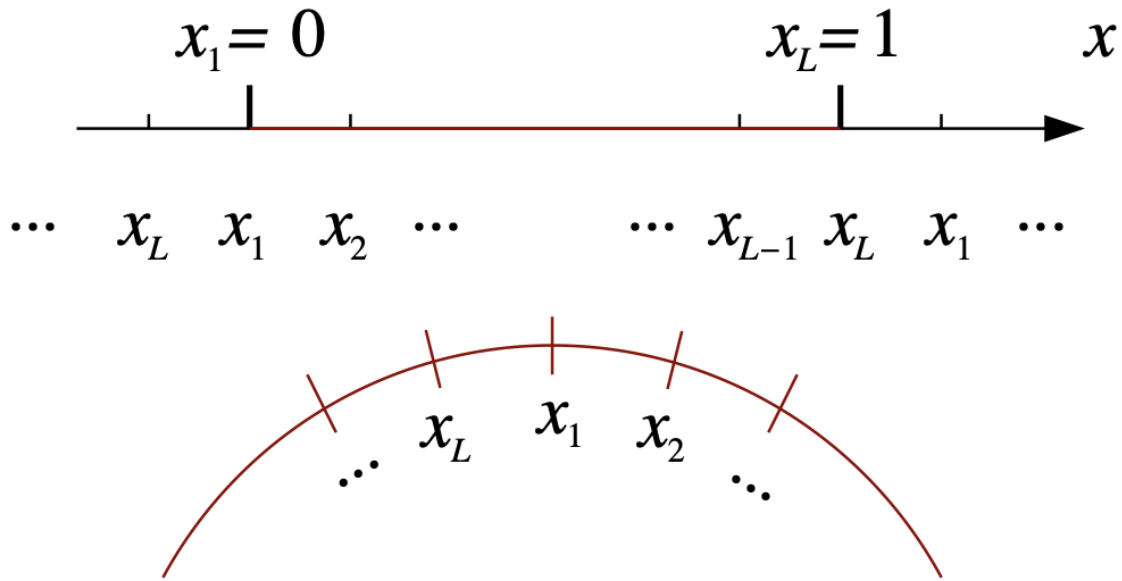
- (Remember *Dirichlet*?: specify the value at the boundary and solve inside).
- Another choice is *periodic boundary conditions*
- We assume we're on a circular strip of paper, such that walking off the right end takes us to the left end.
  - Just like walking out the right/left tunnels in Pacman.



Periodic Boundary Conditions

- Assume one step to the right of $x_L = 1$ coincides with $x_1 = 0$.
- Assume one step to the left of $x_1 = 0$ coincides with $x_L = 1$.
- Mathematically, this corresponds to the *two conditions* (for all t_n):

  1. $a^n_{L+1} = a^n_1$, and
  2. $a^n_0 = a^n_L$.

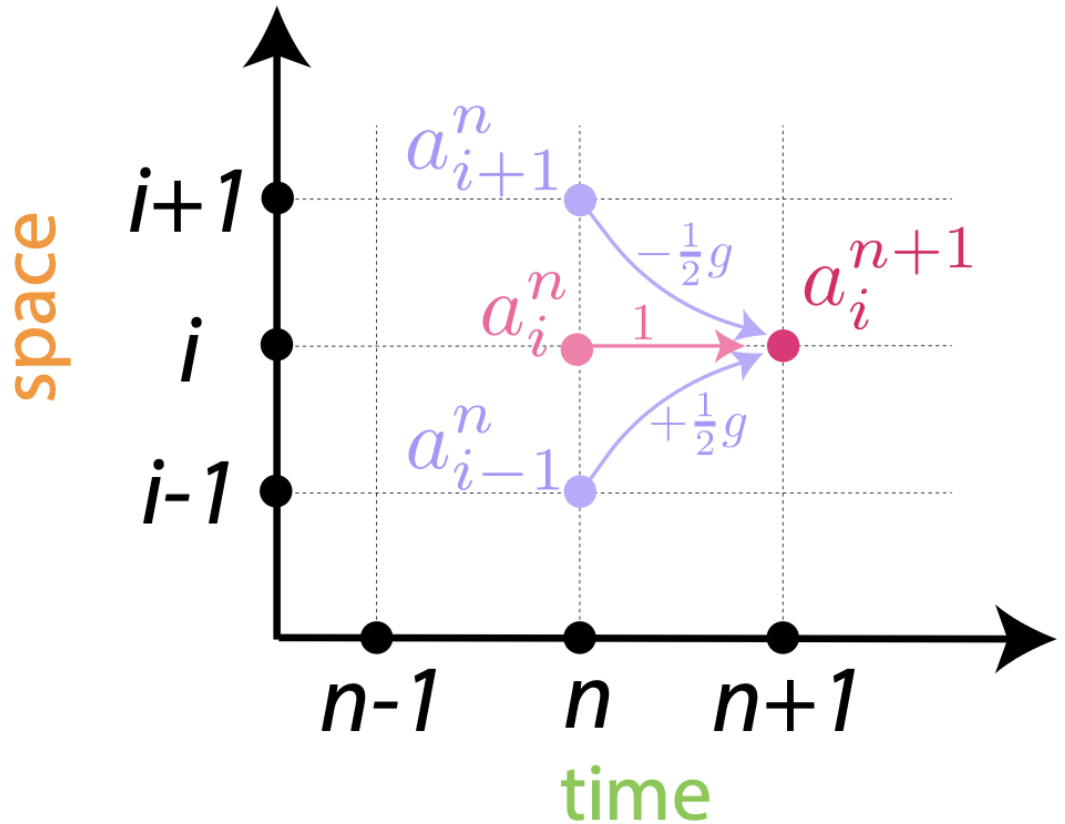


The Forward Time Centered Space (FTCS) Algorithm for Advection, $\frac{\partial a}{\partial t} = -c\frac{\partial a}{\partial x}$

- *Forward difference* approximation for $\frac{\partial a}{\partial t}$ ('step forward in time'):
  - $\frac{a_i^{n+1} - a_i^n}{\tau} + O(\tau)$.
- *Centered difference* approximation for $\frac{\partial a}{\partial x}$ ('a step forward and backward in space'):
  - $\frac{a_{i+1}^n - a_{i-1}^n}{2h} + O(h^2)$.
- Yields: $\boxed{a_i^{n+1} = a_i^n - \frac{1}{2}g\left(a_{i+1}^n - a_{i-1}^n\right)}$.
  - Constants all wrapped up in $g = \frac{c\tau}{h}$.

The FTCS method for advection

$$a_i^{n+1} = a_i^n - \frac{1}{2} g \left(a_{i+1}^n - a_{i-1}^n \right)$$



All the updates $a_i^{n+1} = a_i^n - \frac{1}{2} g \left(a_{i+1}^n - a_{i-1}^n \right)$

$$
\begin{aligned}
i = 1 : \quad & a_1^{n+1} = a_1^n - \tfrac{1}{2} g \left(a_2^n \boxed{- a_L^n} \right) \\
i = 2 : \quad & a_2^{n+1} = a_2^n - \tfrac{1}{2} g \left(a_3^n - a_1^n \right) \\
\vdots = \vdots \quad & \vdots \quad = \quad \vdots \\
i = L-1 : \quad & a_{L-1}^{n+1} = a_{L-1}^n - \tfrac{1}{2} g \left(a_L^n - a_{L-2}^n \right) \\
i = L : \quad & a_L^{n+1} = a_L^n - \tfrac{1}{2} g \left(\boxed{a_1^n} - a_{L-1}^n \right) .
\end{aligned}
$$

- *Periodic BCs*: incorporated in the boxed terms 👀
- Our a_i^{n+1} are linear combinations of our a_i^n
  - So we can write the updates using *matrix multiplication*.

Matrix version of $a_i^{n+1} = a_i^n - \frac{1}{2} g \left(a_{i+1}^n - a_{i-1}^n \right)$

- We can write our updates as $\mathbf{a}^{n+1} = \mathbf{M}\mathbf{a}^n$ with $\mathbf{a}^n = (a_1^n, a_2^n, \ldots, a_L^n)^T$ for
  - $\mathbf{M} = \mathbf{I} - \frac{1}{2} g \mathbf{D}$ with $\mathbf{D} =$

$$\begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 & \boxed{-1} \\ -1 & 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & & & & & & & \vdots \\ 0 & 0 & 0 & 0 & \cdots & -1 & 0 & 1 \\ \boxed{1} & 0 & 0 & 0 & \cdots & 0 & -1 & 0 \end{pmatrix}$$

- *Periodic BCs*: come from the boxed elements 👀
- *In code*, `advection.m`: Once we have constructed `M`, a step forward in time is simply:
  - `amp = M*amp;` 😋

FCTS solution for a Gaussian Initial Value Problem

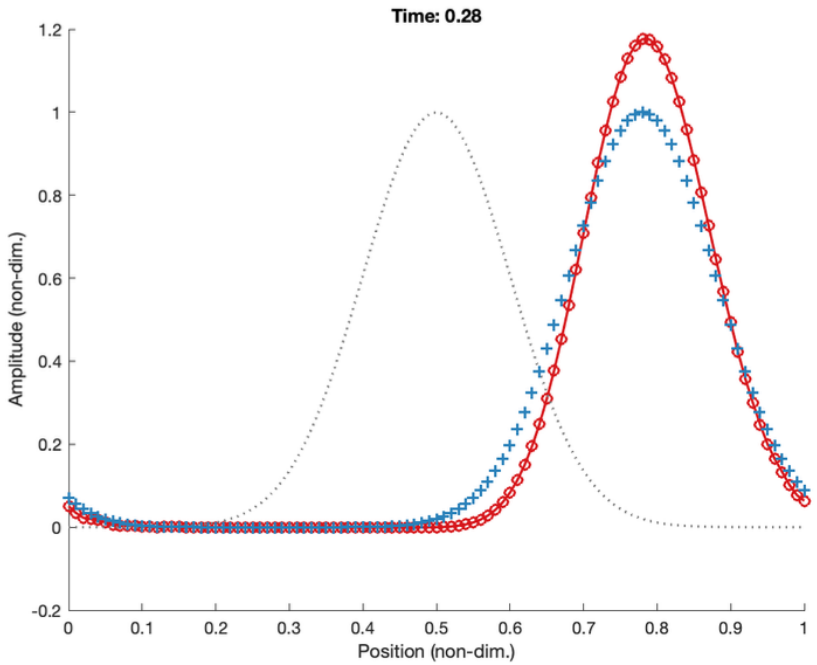
- *Initial profile:* A cute little Gaussian 👶 centered at $x = \frac{1}{2}$:
  - $a(x,0) = \exp\left[-\frac{1}{2\sigma^2}\left(x - \frac{1}{2}\right)^2\right].$
- The analytic solution (with periodic BCs) is:
  - $a(x,t) = \exp\left\{-\left[(x - ct \mod 1 + h) - \frac{1}{2}\right]^2 / 2\sigma^2\right\}.$
  - Translation of the pulse in x by ct (with periodic BCs via the mod).

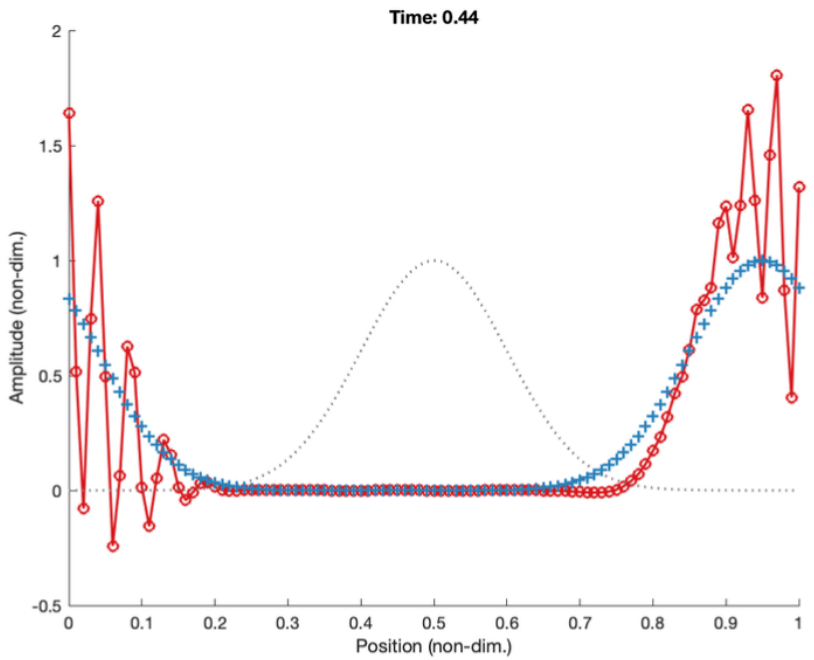
FTCS for advection: advection.ipynb with `whatMethod = 'ftcs'`

- *Remember* (last week): For 1D *diffusion*, FTCS is stable for $\tau < h^2/2\kappa$.
- Let's investigate using advection.ipynb setting **`whatMethod = 'ftcs'`**.
  - Implements FTCS for the Gaussian pulse IVP.
  - Uses the `construct_update_matrix` function to generate the FTCS update matrix for advection (later we will switch to a different method).
  - Defaults are $\tau = h = 0.01$ and $c = 1$ (so $g = \dfrac{c\tau}{h} = 1$).
  - Total number of time steps, $N_{\text{steps}} = (1 + h)/c\tau$.
  - Total distance of propagation: $N_{\text{steps}} \times c\tau = 1 + h$: the size of the domain, so the pulse should *return to its starting position* 🙏
  - Animates the numerical and analytic solutions.
  - Also displays $a(x, t)$ as a surface over the (x, t) plane.

FTCS is unstable for all choices of τ 😟

- The amplitude grows at each time step.
- Doesn't improve with smaller τ: just *delays the inevitable*.
- Matrix stability: spectral radius $\rho(\mathbf{M}) > 1$ *for all* τ! 😲





von Neumann Analysis

- *von Neumann analysis* is more useful in practice than (cumbersome) Matrix stability analysis 😫
  - von Neumann (1903–1957): 20th Century Hungarian mathematician, physicist, computer scientist, engineer, ...
- Involves *looking for complex wave solutions* 👀 (we saw these driving the advection instability above):
  - $a(x,t) = A(t)\mathrm{e}^{ikx}$
    - *wavenumber* k (spatial wavelength $\lambda = 2\pi/k$).
    - complex time-dependent *amplitude* $A(t)$.
  - (🔥 these don't come from nowhere: a Fourier basis expansion)
- ⭐ If $|A(t)|$ grows with time, for any k, *the method is unstable* ⭐

von Neumann Analysis

- We use the discrete version of von Neumann's complex wave solutions, $a(x,t) = A(t)\mathrm{e}^{ikx}$, as:
  - $a_j^n = a(x_j, t_n) = A^n e^{ikjh}$
  - **NB**: here *we use j for the spatial index* to avoid confusion with $i = \sqrt{-1}$.
- Then $a_j^{n+1} = A^{n+1} e^{ikjh} = \xi A^n e^{ikjh}$, where we have defined a complex *amplification factor*, ξ, as how the amplitude, A, of some complex wave solution changes in a step $n \to n+1$:
  - $\xi = \dfrac{A^{n+1}}{A^n}$.
  - This is important because an unstable method grows in amplitude over sucessive iterations...
- ⭐ *If $|\xi| > 1$ for any k in a time step the method is unstable*
  - NB: $|\xi|^2 = \mathrm{Re}(\xi)^2 + \mathrm{Im}(\xi)^2$.
- Our strategy in *von Neumann stability analysis* is to insert our trial oscillatory solution into the numerical scheme and see how ξ depends on τ and h! 😄

von Neumann to FTCS for advection 🔨

$a_j^{n+1} = a_j^n - \frac{1}{2}g\left(a_{j+1}^n - a_{j-1}^n\right)$, with $g = \frac{c\tau}{h}$

- ❓ Consider a trial solution, $a_j^n = A^n e^{ikjh}$ and compute $\xi = \dfrac{A^{n+1}}{A^n}$.
  - 👀 $\xi A^n e^{ikjh} = A^n e^{ikjh} - \ldots$
    - $\ldots \frac{1}{2}g\left[A^n e^{ik(j+1)h} - A^n e^{ik(j-1)h}\right]$
  - i.e., $\xi = 1 - \frac{1}{2}g\left(\mathrm{e}^{ikh} - \mathrm{e}^{-ikh}\right)$
  - So $\xi = 1 - ig\sin(kh)$
    - using $\sin(kh) = \frac{1}{2i}\left(e^{ikh} - e^{-ikh}\right)$

- ❓ Determine $|\xi|$ and assess stability.
  - 👀 $|\xi| = \sqrt{1 + g^2\sin^2(kh)}$
  - 👀 Since $g^2 \geq 0$ and $\sin^2(kh) \geq 0$, $\Rightarrow |\xi| > 1$ for some k *for all* τ.
  - 👀 So FTCS is *unconditionally unstable* (for all values of τ): it is a 'useless' numerical method 😢

Fixing FTCS for advection

- We need to 'fix' the inherent instability.
- Peter Lax 👨‍👨‍👦 introduced a *simple fix* for this inherent instability 🔨
- ❓ Which do you think it was?:

1. $a_j^{n+1} = a_j^n - \frac{1}{2}g\left(\boxed{2a_{j+1}^n} - a_{j-1}^n\right)$
2. $a_j^{n+1} = \boxed{\frac{1}{2}\left(a_{j-1}^n + a_{j+1}^n\right)} - \frac{1}{2}g\left(a_{j+1}^n - a_{j-1}^n\right)$
3. $a_j^{n+1} = \boxed{\frac{1}{3}\left(a_{j-1}^n + a_j^n + a_{j+1}^n\right)} - \frac{1}{2}g\left(a_{j+1}^n - a_{j-1}^n\right)$

The Lax Method

- Lax's idea was to introduce some 'artificial diffusion' (averaging) to counteract the FTCS instability.
- Replace a_i^n term on RHS in FTCS by a *spatial average*.
- *The Lax method for advection*:
  - $$a_j^{n+1} = \boxed{\frac{1}{2}\left(a_{j-1}^n + a_{j+1}^n\right)} - \tfrac{1}{2}g\left(a_{j+1}^n - a_{j-1}^n\right)$$
  - $g = c\tau/h$
- Compare to *FTCS*:

$$a_j^{n+1} = \boxed{a_j^n} - \tfrac{1}{2}g\left(a_{j+1}^n - a_{j-1}^n\right).$$

Lax method for advection: von Neumann analysis

- We can compute amplification factor $\xi = \cos(kh) - ig\sin(kh)$
  - So $|\xi|^2 = \cos^2(kh) + g^2\sin^2(kh) = 1 + (g^2 - 1)\sin^2(kh)$
- Hence $|\xi|^2 \leq 1$ for $g^2 \leq 1$, i.e., stable for $g \leq 1$.
- $g \leq 1$ corresponds to $\tau \leq \tau_{\max} = \dfrac{h}{c}$.
  - This is the *Courant-Friedrichs-Lewy (CFL) condition*.
- Physically, $\tau_{\max} = h/c$ corresponds to the propagation of one spatial step per time step.

Lax method for advection: Matrix version

- $\mathbf{a}^{n+1} = \left(\mathrm{A} - \frac{1}{2} g \mathrm{D}\right) \mathbf{a}^{n}$,
  - where D is defined as before but we just slightly tinker with A to get our averaging in:

$$\mathrm{A} = \frac{1}{2}|D| = \frac{1}{2}\begin{pmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & \boxed{1} \\ 1 & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & & & & & & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 \\ \boxed{1} & 0 & 0 & 0 & \cdots & 1 & 0 \end{pmatrix}.$$

(Periodic boundary conditions are the boxed terms).

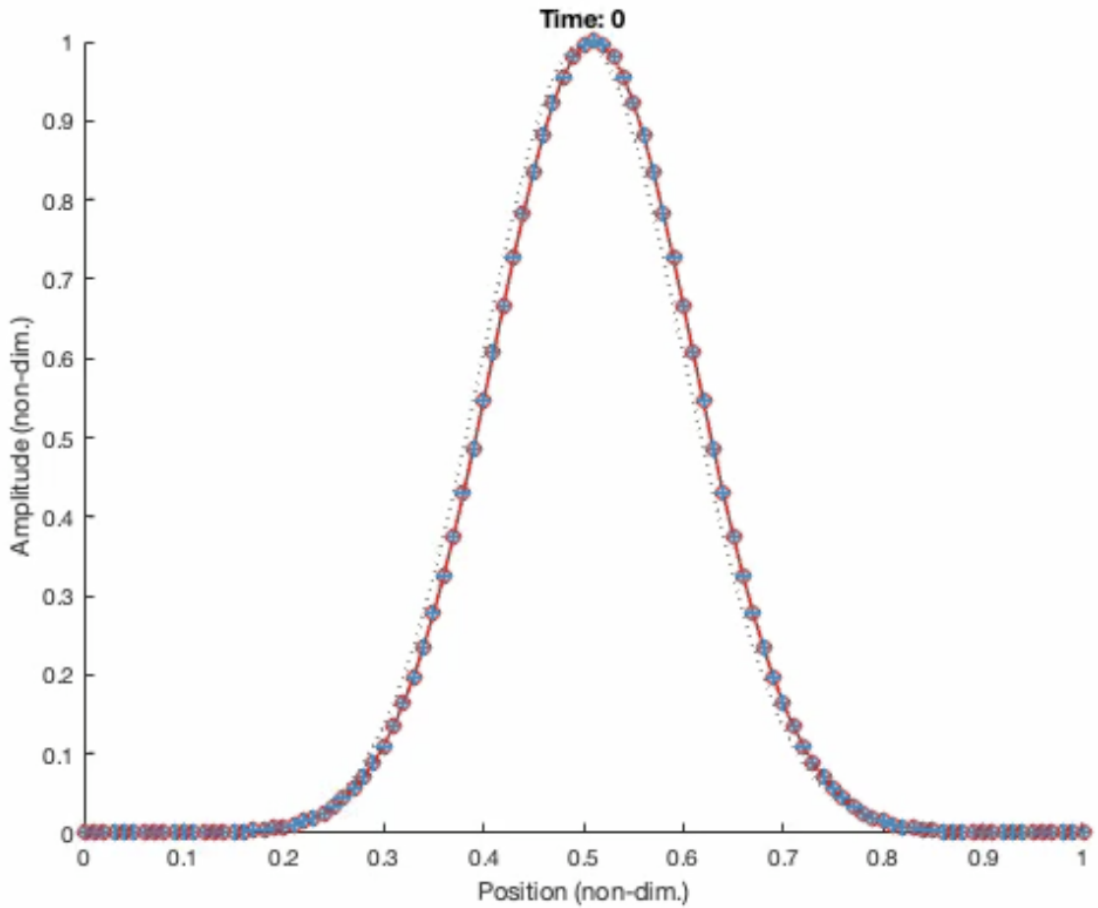
Lax method for advection advection.ipynb with `whatMethod = 'lax'`

- The *only change* is in how the update matrix is constructed (the call to

```
# Diagonal matrix to implement the centred spatial first-derivative

D = np.zeros((L, L))

for i in range(L-1):
    D[i, i+1] = 1     # Super-diagonal
    D[i+1, i] = -1    # Sub-diagonal

# Additional elements for periodic boundary conditions
D[0, L-1] = -1
D[L-1, 0] = 1

# Lax averaging matrix
A = 0.5*np.abs(D)

# Update matrix
M = A - 0.5*g*D
```

Lax Method

- 😢 For $\tau > \tau_{\max}$: *unstable*.
- 😂 For $\tau = \tau_{\max} = \frac{h}{c}$: *exact* propagation!
- For $\tau < \tau_{\max}$: pulse spreads (*diffuses*) as it propagates.
- Smaller time steps τ don't always yield more accurate solutions 😲



Good luck out there! 🦤

- This course is a *very brief introductory sketch* into some of the principles of using computers to simulate the solution of physical equations.
- In the next few years when you've flown from this here nest 🦤, you might get your dirty hands on some gnarly equations. *Then what do you do?*

1. **The same principles apply**
   - Similar methods can often be adapted to different problems.
2. **Look to the literature for guidance (textbooks and refereed journal articles)**
   - This course focused on *finite differencing* methods, but there are many other powerful approaches (e.g., finite elements, spectral methods, ...)
3. **Always try the simplest thing first**
   - Solve a simpler problem/a limited case first, then build up if required.
4. **Always test any numerical method on known analytic solutions to understand/validate its behavior**.

Assignment

- Good luck with your assignment!

Computational Physics Cheat Sheet

Numerical errors

Floating point versus truncation errors

Global versus local error $E_g \approx O(\tau^{k-1})$

Constructing algorithms

1st derivative: forward difference approximation:

$$f'(t) = \frac{f(t+\tau) - f(t)}{\tau} + O(\tau)$$

1st derivative: centered difference approximation:

$$f'(t) = \frac{f(t+\tau) - f(t-\tau)}{2\tau} + O(\tau^2)$$

2nd derivative: centered difference approximation:

$$f''(t) = \frac{f(t+\tau) - 2f(t) + f(t-\tau)}{\tau^2} + O(\tau^2)$$

1D Diffusion $\frac{\partial T}{\partial t} = \kappa \frac{\partial^2 T}{\partial x^2}$

FTCS: $T_i^{n+1} = T_i^n + f\left(T_{i-1}^n - 2T_i^n + T_{i+1}^n\right) \qquad f = \frac{\kappa\tau}{h^2}$

Matrix form: $\mathbf{T}^{n+1} = (\mathrm{I} + \mathrm{D})\mathbf{T}^n = \mathrm{A}\mathbf{T}^n$ Stable: $\tau \leq \frac{h^2}{2\kappa}$

Advection $\frac{\partial a}{\partial t} = -c \frac{\partial a}{\partial x}$

FTCS: $a_i^{n+1} = a_i^n - \frac{1}{2} g\left(a_{i+1}^n - a_{i-1}^n\right) \qquad g = \frac{c\tau}{h}$

Matrix form: $\mathbf{a}^{n+1} = (\mathrm{I} - \frac{1}{2} g \mathrm{D})\mathbf{a}^n$ Stability: (unstable)

Lax: $a_j^{n+1} = \frac{1}{2}\left(a_{j-1}^n + a_{j+1}^n\right) - \frac{1}{2} g\left(a_{j+1}^n - a_{j-1}^n\right)$

Stability: $g \leq 1$

Non-dimensionlization

Introduce new dependent and independent variables by rescaling each by chosen constants.
Do not rescale constants/parameters.

$$\bar{\mathbf{r}} = \frac{\mathbf{r}}{L_s}, \quad \bar{t} = \frac{t}{t_s}, \quad \bar{\mathbf{v}} = \frac{\mathbf{v}}{L_s/t_s}.$$

Algorithms for ODEs

Dynamics: $\frac{d\mathbf{r}}{dt} = \mathbf{v}, \quad \frac{d\mathbf{v}}{dt} = \mathbf{a}(\mathbf{r}, \mathbf{v}, t).$

Euler:
$$\mathbf{r}_{n+1} = \mathbf{r}_n + \tau \mathbf{v}_n,$$
$$\mathbf{v}_{n+1} = \mathbf{v}_n + \tau \mathbf{a}_n.$$

General form: $\frac{d\mathbf{x}}{dt} = \mathbf{f}\left[\mathbf{x}(t), t\right]$

Euler (RK1): $\mathbf{x}(t+\tau) = \mathbf{x}(t) + \tau \mathbf{f}^{(1)}$

RK4: $\mathbf{x}(t+\tau) = \mathbf{x}(t) + \frac{1}{6}\tau\left[\mathbf{f}^{(1)} + 2\mathbf{f}^{(2)} + 2\mathbf{f}^{(3)} + \mathbf{f}^{(4)}\right]$

Stability analysis

Matrix power formulation: spectral radius $\rho(\mathrm{A}) \leq 1$

von Neumann: trial solution: $a(x,t) = A(t)\mathrm{e}^{ikx}$

Amplification factor: $\xi = \frac{A^{n+1}}{A^n}$ Stability: $|\xi| \leq 1$