On what basis?

- **Topic:** basis sets, series expansions, Fourier analysis
- **Task A:**
  1. Simulate the wave using a complex Fourier transformation. What type of a boundary does it implement?
  2. Try reconstructing the solution using only a few expansion coefficients. How does this change the outcome?
  3. Try simulations with different initial shapes.
- **Task B:**
  1. Implement the method to reconstruct the function as a sine series. You can look at the cosine series for guidance.
  2. Simulate the wave using sine and cosine series expansions. What types of a boundaries do these correspond to?
  3. Try simulations with different initial shapes.
- **Template:**
  - fourier.py
- **Further reading:**
  - https://en.wikipedia.org/wiki/Fourier_series
  - https://en.wikipedia.org/wiki/Basis_(linear_algebra)
  - https://en.wikipedia.org/wiki/Basis_set_(chemistry)
  - https://en.wikipedia.org/wiki/Fast_Fourier_transform

fourier.py

`fourier.`**`add_gaussian`**`(string, a=1, center=0.5)` [source]

Initialize the wave to a gaussian initial shape.

The function adjusts the shape of the wave by adding a narrow gaussian function to the wave function. Since this operation is additive, it can be called several times to add several pulses.

Parameters:
- **string** (*array*) - discretized wave function to be modified
- **a** (*float*) – pulse height
- **center** (*float*) – fractional position of the peak, should be between 0 and 1

`fourier.`**`add_sinusoidal`**`(string, n, a=1)` [source]

Initialize the wave to a sinusoidal initial shape.

The function adjusts the shape of the wave by adding a sinus function to the wave function. The sinus function will go to zero at the edges of the system. Since this operation is additive, it can be called several times to build the wave as a superposition of sinus functions.

Parameters:
- **string** (*array*) - discretized wave function to be modified
- **n** (*int*) – number of antinodes
- **a** (*float*) – amplitude

`fourier.`**`add_triangle`**`(string, a=1, peak=0.5)` [source]

Initialize the wave to a triangular initial shape.

The function adjusts the shape of the wave by adding a triangular function to the wave function. The triangle will go to zero at the edges of the system. Since this operation is additive, it can be called several times to add several triangles.

Parameters:
- **string** (*array*) - discretized wave function to be modified
- **a** (*float*) – amplitude
- **peak** (*float*) - fractional position of the peak, should be between 0 and 1

`fourier.`**`animate`**`(wave, L)` [source]

Animates the simulation.

Parameters:
- **wave** (*list*) - list of waves as a time series
- **L** (*float*) – length of simulated region L

`fourier.`**`calculate_inverse_transformation`**`(type, coefficients, v, L, t, n_nodes)` [source]

Reconstructs wave function from expansion coefficients.

The function calls one of the Fourier analysis methods:

- `fourier_synthesis()` if type is "exp"
- `sin_synthesis()` if type is "sin"
- `cos_synthesis()` if type is "cos"

Parameters:
- **type** (*str*) – one of "exp", "sin" or "cos"
- **coefficients** (*array*) – series coefficients b_n
- **v** (*float*) – wave speed v
- **L** (*float*) – length of simulated region L
- **t** (*float*) – time t
- **n_nodes** (*int*) – number of grid points in the original wave function, N

Returns: discretized wave function u

Return type: array

`fourier.`**`calculate_transformation`**`(type, wave)` [source]

Calculates expansion coefficients.

The function calls one of the Fourier analysis methods:

- `fourier_analysis()` if type is "exp"
- `sin_analysis()` if type is "sin"
- `cos_analysis()` if type is "cos"

Parameters:
- **type** (*str*) – one of "exp", "sin" or "cos"
- **wave** (*array*) - discretized wave function

Returns: Expansion coefficients

Return type: array

`fourier.`**`cos_analysis`**`(wave)` [source]

Calculates coefficients of the discrete cosine series.

A wave function defined in $[0, L]$ can be represented as a series

$$u(x) = \sum_{n=0}^{N} a_n \cos\left(\frac{\pi n}{L} x\right),$$

if slope of the function is zero at both boundaries, $u'(0) = u'(L) = 0$.

This function calculates the coefficients a_n using numpy's fast Fourier transform functionality. This is done by taking the original function and extending it to $[0, 2L]$ so that the result is symmetric with respect to the point $x = L$. Due to symmetry, the complex Fourier transform of this function will yield real valued coefficients which are also the coefficients for the cosine series of the original function.

Parameters:
- **wave** (*array*) - discretized wave function

Returns: Expansion coefficients a_n

Return type: array

`fourier.`**`cos_synthesis`**`(a, v, L, t, n_nodes, visual=False)` [source]

Calculates the wave function from cosine series coefficients.

A wave function defined in $[0, L]$ obeying $u'(0) = u'(L) = 0$ can be represented as a time dependent cosine series

$$u(x,t) = \sum_{n=0}^{N} a_n \cos\left(\frac{\pi n}{L} x\right) \cos(\omega_n t)$$

where the angular frequency of component n is $\omega_n = \pi n v / L$. This assumes the wave was stationary at $t = 0$.

This function reconstructs a discretized approximation for the wave function at a given time from the expansion coefficients.

Note that if the original discretized wave function was known at N grid points, this function reconstruct a wave function at $N - 2$ points. This is because the original function can be almost anything but the end result must obey the boundary conditions $u'(0) = u'(L) = 0$. Therefore the reconstruction adds one grid point at both ends of the function so that it explicitly satisfies this condition.

Parameters:
- **a** (*array*) – cosine series coefficients a_n
- **v** (*float*) – wave speed v
- **L** (*float*) – length of simulated region L
- **t** (*float*) – time t
- **n_nodes** (*int*) – number of grid points in the original wave function, N
- **visual** (*bool*) – if True, the harmonic components and their superposition is plotted

Returns: discretized wave function u

Return type: array

`fourier.`**`draw`**`(frame, wave, x)` [source]

Draws the wave.

Used for animation.

Parameters:
- **frame** (*int*) – index of the frame to be drawn
- **wave** (*list*) - list of waves as a time series
- **x** (*array*) – x coordinates

`fourier.`**`fourier_analysis`**`(wave)` [source]

Calculates coefficients of the discrete complex Fourier transformation.

A wave function defined in $[0, L]$ can be represented as a Fourier series

$$u(x) = \sum_{n=0}^{N} c_n e^{i\frac{2\pi}{L}x}.$$

This function calculates the coefficients c_n using numpy's fast Fourier transform functionality.

Parameters:
- **wave** (*array*) - discretized wave function

Returns: Fourier coefficients c_n

Return type: array

`fourier.`**`fourier_synthesis`**`(c, v, L, t, n_nodes)` [source]

Calculates the wave function from Fourier coefficients.

A wave function defined in $[0, L]$ can be represented as a time dependent Fourier series

$$u(x,t) = \sum_{n=0}^{N} c_n e^{i\left(\frac{2\pi n}{L}x - \omega_n t\right)}$$

where the angular frequency of component n is $\omega_n = 2\pi n v / L$.

This function reconstructs a discretized approximation for the wave function at a given time from the Fourier coefficients.

Parameters:
- **c** (*array*) – Fourier coefficients c_n
- **v** (*float*) – wave speed v
- **L** (*float*) – length of simulated region L
- **t** (*float*) – time t
- **n_nodes** (*int*) – number of grid points in the original wave function

Returns: discretized wave function u

Return type: array

`fourier.`**`main`**`()` [source]

Main function. Simulates a 1D wave using harmonic basis functions.

`fourier.`**`print_progress`**`(step, total)` [source]

Prints a progress bar.

Parameters:
- **step** (*int*) – progress counter
- **total** (*int*) – counter at completion

`fourier.`**`run_simulation`**`(type, initial_wave, v, L, time, dt, cutoff, plot_coefficients=False)` [source]

Calculates the wave function at different time steps.

The wave function is solved at equally spaced time steps by first calculating the harmonic series expansion coefficients of the wave function at time $t = 0$. Once the series expansion is known, the wave function can be calculated at any time by calculating the phases of all the harmonic components at the given time and reconstructing the wave function from the exansion coefficients and the time-dependent harmonic functions.

It is also possible to calculate approximate solutions by only including some of the first expansion coefficients in the calculation. This saves both calculation time and memory.

Parameters:
- **type** (*str*) – one of "exp", "sin" or "cos"
- **initial_wave** (*array*) - discretized wave function at the beginning
- **v** (*float*) – wave speed v / L
- **L** (*float*) – length of simulated region L
- **time** (*float*) – total simulation time
- **dt** (*float*) – time between recorded wave functions
- **cutoff** (*int*) – the number of expansion coefficients to include
- **plot_coefficients** (*bool*) - If True, expansion coefficients are plotted. All coefficients are shown as black dots. The ones included in the simulation are shown with a white center.

Returns: time evolution of the wave function

Return type: list

`fourier.`**`sin_analysis`**`(wave)` [source]

Calculates coefficients of the discrete sine series.

A wave function defined in $[0, L]$ can be represented as a series

$$u(x) = \sum_{n=0}^{N} b_n \sin\left(\frac{\pi n}{L} x\right),$$

if the function is zero at both boundaries, $u(0) = u(L) = 0$.

This function calculates the coefficients a_n using numpy's fast Fourier transform functionality. This is done by taking the original function and extending it to $[0, 2L]$ so that the result is antisymmetric with respect to the point $x = L$. Due to symmetry, the complex Fourier transform of this function will yield imaginary valued coefficients the absolute values of which are also the coefficients for the sine series of the original function.

Parameters:
- **wave** (*array*) - discretized wave function

Returns: Expansion coefficients b_n

Return type: array

`fourier.`**`sin_synthesis`**`(b, v, L, t, n_nodes, visual=False)` [source]

Calculates the wave function from sine series coefficients.

A wave function defined in $[0, L]$ obeying $u(0) = u(L) = 0$ can be represented as a time dependent sine series

$$u(x,t) = \sum_{n=0}^{N} b_n \sin\left(\frac{\pi n}{L} x\right) \cos(\omega_n t)$$

where the angular frequency of component n is $\omega_n = \pi n v / L$. This assumes the wave was stationary at $t = 0$.

This function reconstructs a discretized approximation for the wave function at a given time from the expansion coefficients.

Note that if the original discretized wave function was known at N grid points, this function reconstruct a wave function at $N + 2$ points. This is because the original function can be almost anything but the end result must obey the boundary conditions $u(0) = u(L) = 0$. Therefore the reconstruction adds one grid point at both ends of the function so that it explicitly satisfies this condition.

Note

This function is incomplete!

Parameters:
- **b** (*array*) – sine series coefficients b_n
- **v** (*float*) – wave speed v
- **L** (*float*) – length of simulated region L
- **t** (*float*) – time t
- **n_nodes** (*int*) – number of grid points in the original wave function, N
- **visual** (*bool*) – if True, the harmonic components and their superposition is plotted

Returns: discretized wave function u

Return type: array

Table of Contents
On what basis?
fourier.py
Previous topic
Good vibes
Next topic
Elementary
This Page
Show Source
Quick search
Go