Physics 331
Introduction to Numerical Techniques in Physics

Instructor: Joaquín Drut
Last time:

- Polynomial interpolation: basics; Lagrange interpolation.
Today:

- Quick review. Formal properties.
- Polynomial interpolation: Splines!
- Practice
Polynomial interpolation: basics.

Given \( n \) points \((x,y)\), they uniquely determine a polynomial of degree \( m = n-1 \).

Why use a polynomial? What are the advantages?

Any disadvantages you can think of?

Advice:

- Always keep in mind the oscillating behavior of polynomials.

- Resist their nice properties... unless you know for sure that the underlying function is a polynomial!

- Try to avoid high-order (use your own judgement to decide).

- Last but not least: remember that polynomials can be numerically unstable beasts!
Some formal properties of polynomials

- Polynomials of degree less or equal to $n > 0$ form an $n+1$ dimensional vector space.

  You can add and subtract, multiply by a constant, etc.

  A possible basis is the set of monomials:

  $$\{1, x, x^2, ..., x^n\}$$

  The components of the vectors are the coefficients $\{a_n\}$, such that

  $$P(x) = a_0 + a_1x + a_2x^2 + ... + a_nx^n$$
Wait… a “vector space”? What is that?

- A set of objects we call **vectors**, e.g. n-tuples \((a_1, a_2, a_3, \ldots, a_n)\), where the entries are real numbers.

- A set of objects we call **scalars**, e.g. the set of real numbers.

**Such that…**

The scalars form a “**field**”, which means that addition and multiplication are defined in the usual way (associativity, commutativity, distributive property), those operations are “closed” in the set, and inverses and null elements “0” and “1” exist and are in the set as well.
Wait… a “vector space”? What is that?

• A set of objects we call **vectors**, e.g. n-tuples \((a_1, a_2, a_3, \ldots, a_n)\), where the entries are real numbers.

• A set of objects we call **scalars**, e.g. the set of real numbers.

• **Such that**…

  The scalars form a “**field**”, which means that addition and multiplication are defined in the usual way (associativity, commutativity, distributive property), those operations are “closed” in the set, and inverses and null elements “0” and “1” exist and are in the set as well.

A set of operations between vectors and scalars exists:

• **Vector-vector** addition “+” is associative and commutative, and there is a null vector “0”.

• **Scalar-vector** multiplication is closed in the set and obeys associative, commutative, and distributive properties

**Note:** No **vector-vector** multiplication of any kind is defined at this stage!
Wait... a “vector space”? What is that?

- Examples?
Wait... a “vector space”? What is that?

- Examples
  - Vectors on the plane with reals (or rationals) as scalars.
  - Complex numbers (as vectors) with reals (or rationals) as scalars.
  - n-tuples with real entries and reals (or rationals) as scalars.
  - Polynomials with real coefficients and reals (or rationals) as scalars.
Wait... what happened to the “dot” product between vectors?

- The scalar (or “dot”, or “inner”) product
  
  An operation involving two vectors \( \mathbf{v}, \mathbf{w} \) mapped onto a scalar.

  We denote that resulting scalar by

  \[
  (\mathbf{v}, \mathbf{w}) \quad \text{or sometimes} \quad \mathbf{v} \cdot \mathbf{w}
  \]

  This mapping is such that:

  \[
  (\mathbf{v}, \mathbf{w}) = (\mathbf{w}, \mathbf{v})^* \quad \text{“Conjugate property”}
  \]

  \[
  (x \mathbf{v}, \mathbf{w}) = x (\mathbf{v}, \mathbf{w}) \quad \text{where } x \text{ is any scalar} \quad \text{“Linearity”}
  \]

  \[
  (\mathbf{v} + \mathbf{u}, \mathbf{w}) = (\mathbf{v}, \mathbf{w}) + (\mathbf{u}, \mathbf{w})
  \]

  \[
  (\mathbf{v}, \mathbf{v}) > 0 \quad \text{“Positive definiteness”}
  \]

  \[
  (\mathbf{v}, \mathbf{v}) = 0 \quad \text{implies } \mathbf{v} = \mathbf{0}
  \]

A vector space armed with an inner product is called an “inner product space”.
Some formal properties of polynomials

- To extend the interpretation of polynomials as vectors, one defines spaces of functions, which are infinite-dimensional vector spaces (specifically, Hilbert spaces).

Imagine interpreting a function as a \textbf{continuous} vector…

\[\mathbf{v} \rightarrow [\mathbf{v}]_i = v_i\quad \text{vector and its \textbf{discrete coordinates}}\]

\[f \rightarrow f(x) = f(x)\quad \text{function and its \textbf{continuous coordinates}}\]

Instead of
\[\mathbf{v} \cdot \mathbf{w} = \sum_i v_i w_i\]

\[\text{we’ll have}\]
\[f \cdot g = \int dx f(x)g(x)\]
Some formal properties of polynomials

- Polynomials can only be square integrable if we define them on a bounded subset of the real line, e.g. [-1,1]... Why?

Then, the dot-product definition

\[ f \cdot g = \int dx \, f(x)g(x) \]

makes sense.

- A more general definition of the dot product for such function spaces is possible:

\[ f \cdot g = \int dx \, W(x)f(x)g(x) \]

where we have used a positive definite weight function \( W(x) \).

If we orthonormalize a set of polynomials according to the above dot product, we obtain different sets of well-known polynomials (next slide).

There is a well-known algorithm for orthonormalization: “Gram-Schmidt”. We will return to this algorithm in a few lectures.
Some formal properties of polynomials

_Gauss-Legendre:_

\[ W(x) = 1 \quad -1 < x < 1 \]

\[(j + 1)P_{j+1} = (2j + 1)xP_j - jP_{j-1} \]

_Gauss-Chebyshev:_

\[ W(x) = (1 - x^2)^{-1/2} \quad -1 < x < 1 \]

\[ T_{j+1} = 2xT_j - T_{j-1} \]

_Gauss-Laguerre:_

\[ W(x) = x^\alpha e^{-x} \quad 0 < x < \infty \]

\[(j + 1)L_{j+1}^\alpha = (-x + 2j + \alpha + 1)L_j^\alpha - (j + \alpha)L_{j-1}^\alpha \]

_Gauss-Hermite:_

\[ W(x) = e^{-x^2} \quad -\infty < x < \infty \]

\[ H_{j+1} = 2xH_j - 2jH_{j-1} \]
Polynomial interpolation: “easiest” way.

Given \( n \) points \((x,y)\), they *uniquely* determine a polynomial of degree \( m = n-1 \).

\[
f(x) = a_m x^m + a_{m-1} x^{m-1} + ... + a_0
\]

How do we find the \( n = m+1 \) coefficients based on our \( n \) datapoints?

\[
(x_1, y_1) \quad \quad a_m x_1^m + a_{m-1} x_1^{m-1} + ... + a_0 = y_1
\]

\[
(x_2, y_2) \quad \quad a_m x_2^m + a_{m-1} x_2^{m-1} + ... + a_0 = y_2
\]

\[
(x_3, y_3) \quad \quad : \quad : \quad : \quad : \quad : \quad : \quad : \quad : \quad : \quad :
\]

\[
(x_n, y_n) \quad \quad a_m x_n^m + a_{m-1} x_n^{m-1} + ... + a_0 = y_n
\]

A “nice” linear set of equations for the coefficients, but generally a bad idea due to ill-conditioning: the determinant is usually close to zero.
Polynomial interpolation: Lagrange version.

Given n points \((x,y)\), they uniquely determine a polynomial of degree \(m = n - 1\).

\[
f(x) = a_m x^m + a_{m-1} x^{m-1} + \ldots + a_0
\]

How do we find the \(n = m+1\) coefficients based on our \(n\) datapoints?

\((x_1, y_1)\) \(n = m+1 = 2\) case, by definition:

\[
f(x) = y = a_2(x - x_1) + a_1(x - x_2)
\]

\((x_2, y_2)\)

\((x_3, y_3)\) \(n = m+1 = 3\) case, by definition:

\[
f(x) = y = a_1(x - x_2)(x - x_3) + a_2(x - x_1)(x - x_3) + a_3(x - x_1)(x - x_2)
\]

What about in general?
Polynomial interpolation: Lagrange version.

General expression

\[ f(x) = \sum_{i=1}^{n} y_i L_i(x) \]

“Lagrange functions”

\[ L_i(x) = \prod_{j=1; j \neq i}^{n} \frac{(x - x_j)}{(x_i - x_j)} \]

Keep in mind:

These polynomials are used by evaluating the Lagrange functions at the point of interest \( x \), every time a new point is needed!

This is much more costly than computing coefficients once and for all and using them later for any new point we may want!

However, it can be expected to be numerically more stable!

No matrix inversion needed!
Polynomial interpolation

Polynomials have great continuity and differentiability properties.

However, if we have many datapoints (and you don’t need that many to see this), a polynomial interpolation will quickly develop unwanted oscillations.

**The final interpolating form will depend strongly on the support points!**

What can we do about this?
Splines!
Polynomial interpolation: linear splines.

For each pair of points, use the linear Lagrange form:

\[(x_1, y_1)\]
\[(x_2, y_2)\]
\[(x_3, y_3)\]
\[\vdots\]
\[(x_n, y_n)\]
Polynomial interpolation: linear splines.

For each pair of points, use the linear Lagrange form:

\[ f(x) = y = a_2(x - x_1) + a_1(x - x_2) \]

E.g.

\[ f(x) = y = y_2 \frac{(x - x_1)}{(x_2 - x_1)} + y_1 \frac{(x - x_2)}{(x_1 - x_2)} \]

...for points in the first interval.
Polynomial interpolation: linear splines.

For each pair of points, use the linear Lagrange form:

\[ f(x) = y = a_2(x - x_1) + a_1(x - x_2) \]

E.g.

\[ f(x) = y = y_2 \frac{(x - x_1)}{(x_2 - x_1)} + y_1 \frac{(x - x_2)}{(x_1 - x_2)} \]

...for points in the first interval.

In general,

\[ f_i(x) = y_{i+1} \frac{(x - x_i)}{(x_{i+1} - x_i)} + y_i \frac{(x - x_{i+1})}{(x_i - x_{i+1})} \]

valid in \([x_i, x_{i+1}]\)
Polynomial interpolation: linear splines.

For each pair of points, use the linear Lagrange form:

\( f(x) = y = a_2(x - x_1) + a_1(x - x_2) \)

\( = y_2 \frac{(x - x_1)}{(x_2 - x_1)} + y_1 \frac{(x - x_2)}{(x_1 - x_2)} \)

E.g.

\( f(x) = y = y_2 \frac{(x - x_1)}{(x_2 - x_1)} + y_1 \frac{(x - x_2)}{(x_1 - x_2)} \)

...for points in the first interval.

In general,

\( f_i(x) = y_{i+1} \frac{(x - x_i)}{(x_{i+1} - x_i)} + y_i \frac{(x - x_{i+1})}{(x_i - x_{i+1})} \)

valid in \([x_i, x_{i+1}]\)
Polynomial interpolation: quadratic splines.

In each interval, we propose a quadratic form:

$$f_i(x) = a_i x^2 + b_i x + c_i$$
Polynomial interpolation: quadratic splines.

\[(x_1, y_1)\] In each interval, we propose a quadratic form:

\[f_i(x) = a_i x^2 + b_i x + c_i\]

\[(x_2, y_2)\]

\[(x_3, y_3)\] ...but then we have:

\[n \text{ datapoints which determine } 3 \text{ coefficients at each of the } n-1 \text{ intervals, i.e. } 3(n-1) \text{ coefficients total}\]

\[(x_n, y_n)\]

How do we proceed?
Polynomial interpolation: quadratic splines.

\[(x_1, y_1)\]  \( (x_2, y_2)\)  \( (x_3, y_3)\)  \( \ldots \)   \( (x_n, y_n)\)

In each interval, we propose a quadratic form:

\[f_i(x) = a_i x^2 + b_i x + c_i\]

...but then we have:

\( n \) datapoints which determine \( 3 \) coefficients at each of the \( n-1 \) intervals, i.e. \( 3(n-1) \) coefficients total

How do we proceed?

- Demand that \( f_i(x_i) = y_i \) and \( f_i(x_{i+1}) = y_{i+1} \)
  i.e. the function should pass through the data

  That’s 2 conditions per interval, i.e. \( 2(n-1) \) conditions. We have \( n-1 \) left!
Polynomial interpolation: quadratic splines.

\((x_1, y_1)\) In each interval, we propose a quadratic form:
\((x_2, y_2)\) \(f_i(x) = a_i x^2 + b_i x + c_i\)
\((x_3, y_3)\) ...but then we have:
\(\ldots\)
\(n, n\) datapoints \(\ldots\) which determine \(3\) coefficients at each of the \(n-1\) intervals, i.e. \(3(n-1)\) coefficients total

How do we proceed?

- Demand that \(f_i(x_i) = y_i\) and \(f_i(x_{i+1}) = y_{i+1}\)
  i.e. the function should pass through the data

  That’s 2 conditions per interval, i.e. \(2(n-1)\) conditions. We have \(n-1\) left!

- At the interior points (i.e. all except \(i=1\) and \(i=n\)), match the first derivative from both sides:

  \[f_i'(x_{i+1}) = f_{i+1}'(x_{i+1}) \implies 2a_i x_{i+1} + b_i = 2a_{i+1} x_{i+1} + b_{i+1}\]

  That’s 1 condition per interior point, i.e. \((n-2)\) conditions. We have 1 left!
Polynomial interpolation: quadratic splines.

- Use a linear interpolation for the first interval, instead of a quadratic form.

That gives a total of $3(n-1)$ conditions for our $3(n-1)$ coefficients.

Not only is the resulting interpolation continuous, its derivative is continuous too!

But maybe we want more than that...
Polynomial interpolation: cubic splines.

\((x_1, y_1)\) \(\quad\) In each interval, we propose a cubic form:
\[(x_2, y_2)\] \[f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i\]
\[(x_3, y_3)\]
\[\vdots\]
\[(x_n, y_n)\]

4 coefficients at each of the \(n-1\) intervals, i.e. 4(n-1) coefficients total

How do we proceed?
Polynomial interpolation: cubic splines.

In each interval, we propose a cubic form:

\[ f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i \]

\[(x_1, y_1)\] \quad \text{4 coefficients at each of the n-1 intervals, i.e. 4(n-1) coefficients total}
\[(x_2, y_2)\]
\[(x_3, y_3)\]
\[\vdots\]
\[(x_n, y_n)\]

How do we proceed?

- Match the function: \(2(n-1)\) conditions
- At the \textit{interior points}, match the \textbf{first} derivative from both sides: \((n-2)\) conditions
- At the \textit{interior points}, match the \textbf{second} derivative from both sides: \((n-2)\) conditions

That’s a total of \(4n-6\) conditions; we need 2 more!

Take the second derivative to vanish at the first and last points.