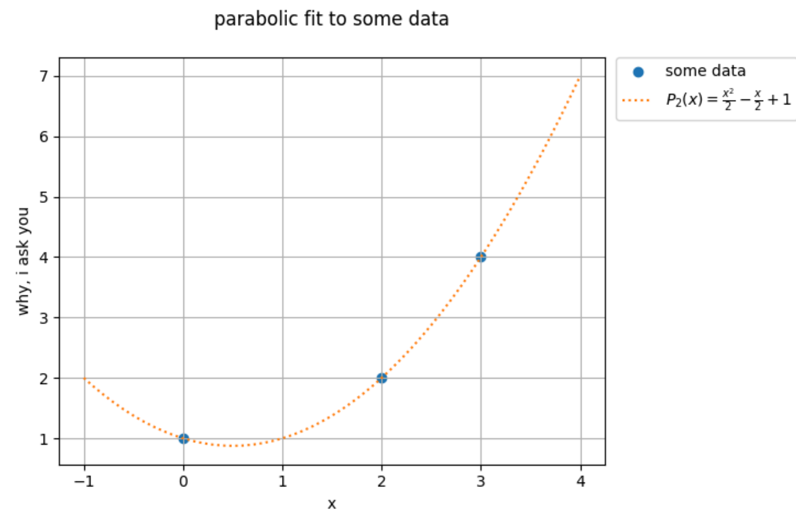
03.01 interpolation: interpolating functions

efficient ways of representing data are fundamental to science and engineering wrt understanding and application. at its most fundamental, approximating data with a polynomial is an act of compression - ie, function $\hat{y}(x)$ takes the place of experimental measurements $\{x_i, y_i\}$ sufficiently to facilitate design.

code, visual

```
import matplotlib.pyplot as plt
import numpy as np

def lazy_spacer(xs,col=0,margin=1,h=0.1):
  """
  xs     : array
  col    : column to space
  margin : how much to extend min,max of column interval
  h      : stepsize per column unit
```



usw

definition 01 function $y = P(x)$ **interpolates** data points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ if $p(x_i) = y_i$ for each $1 \leq i \leq n$.

P must be a function. ie, that each x corresponds to one y. this restricts interpolation of set $\{(x_i, y_i)\}$ in that each x_i must be distinct for a function to pass through them; there is no such restriction on y_i.

wrt interpolation methods, obviously (= first) consider polynomials. q: does a polynomial fit always exist? a: if x_i are distinct then some polynomial $y = P(x)$ runs through them.

also, interpolation is the reverse of evaluation. ie, polynomial evaluation calculates y_i given x_i; polynomial interpolation computes a polynomial from points (x_i, y_i).

1 lagrange interpolation

for an interpolating polynomial, given n data points $(x_1, y_1), \ldots, (x_n, y_n)$, lagrange is a method of explicit formula of degree $d = n - 1$. eg, for three points $(x_1, y_1), (x_2, y_2), (x_3, y_3)$,

$$P_2(x) = y_1 \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} + y_2 \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)} + y_3 \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)}.$$

is the **lagrange interpolating polynomial** for these points. note that

$$\begin{aligned} x_1 \mapsto x &: P_2(x_1) = y_1 + 0 + 0 = y_1 \\ x_2 \mapsto x &: P_2(x_2) = 0 + y_2 + 0 = y_2 \\ x_3 \mapsto x &: P_2(x_3) = 0 + 0 + y_3 = y_3 \end{aligned}$$

example 01

find interpolating polynomial for data points $(0,1),(2,2),(3,4)$.

$$
\begin{aligned}
P_2(x) &= 1\cdot\frac{(x-2)(x-3)}{(0-2)(0-3)} + 2\cdot\frac{(x-0)(x-3)}{(2-0)(0-3)} + 4\cdot\frac{(x-0)(x-2)}{(3-0)(3-2)} \\
&= \frac{1}{6}(x^2-5x+6) + 2(-\frac{1}{2})(x^2-3x) + 4(\frac{1}{3})(x^2-2x) \\
&= \frac{1}{2}x^2 - \frac{1}{2}x + 1. \\
&\Rightarrow \quad P_2(0)=1, P_2(2)=2, P_2(3)=4. \quad \checkmark
\end{aligned}
$$

usw

for n points $(x_1,y_1),\ldots,(x_n,y_n)$, for each k between 1 and n, define degree $d=n-1$ polynomial

$$
L_k(x) = \frac{(x-x_1)\ldots(x-x_{k-1})(x-x_{k+1})\ldots(x-x_n)}{(x_k-x_1)\ldots(x_k-x_{k-1})(x_k-x_{k+1})\ldots(x_k-x_n)}.
$$

as seen for $n=3$, generally $L_k(x_k)=1, L_k(x_{j\neq k})=0$. then

$$
P_{d=n-1}(x_k) = y_1L_1(x_k)+\cdots+y_nL_n(x_k) = 0+\cdots+0+y_kL_k(x_k)+0+\cdots+0 = y_k,
$$

a polynomial of degree at most $n-1$ that passes through any set of n points with distinct x_is -- and it is the only one. she said, as if picking a fight.

theorem 02 main theorem of polynomial interpolation

let $(x_1,y_1),\ldots,(x_n,y_n)$ be n points in the plane with distinct x_i. then there exists one and only one polynomial P of degree $n-1$ or less that satisfies $P(x_i)=y_i, i=1,\ldots,n$.

proof

the existence is proved by the explicit formula for lagrange interpolation. to show there is only one, assume that there are two, $P(x)$ and $Q(x)$ that have degree at most $n-1$ and that both interpolate all n points. ie, $P(x_1)=Q(x_1)=y_1, P(x_2)=Q(x_2)=y_2,\ldots,P(x_n)=Q(x_n)=y_n$. define polynomial $H(x)=P(x)-Q(x)$. H is also of degree of at most $n-1$ and note that $0=H(x_1)=H(x_2)=\ldots=H(x_n)$. ie, H has n distinct zeros. the fundamental theorem of algebra states a degree d polynomial can have at most d zeros unless it is the identically zero polynomial. therefore, H is the identically zero polynomial and $P(x)\equiv Q(x)$. therefore there is a unique $P(x)$ of degree $d\leq n-1$ interpolating the n points (x_i,y_i). ■

example 02

find polynomial of degree three or less that interpolates $(0,2),(1,1),(2,0),(3,-1)$.

lagrange:

$$
\begin{aligned}
P(x) = {} & 2\frac{(x-1)(x-2)(x-3)}{(0-1)(0-2)(0-3)} + 1\frac{(x-0)(x-2)(x-3)}{(1-0)(1-2)(1-3)} \\
& + 0\frac{(x-0)(x-1)(x-3)}{(2-0)(2-1)(2-3)} - 1\frac{(x-0)(x-1)(x-2)}{(3-0)(3-1)(3-2)} \\
= {} & -\frac{1}{3}(x^3-6x^2+11x-6) + \frac{1}{2}(x^3-5x^2+6x) - \frac{1}{6}(x^3-3x^2+2x) \\
= {} & -x+2.
\end{aligned}
$$

code

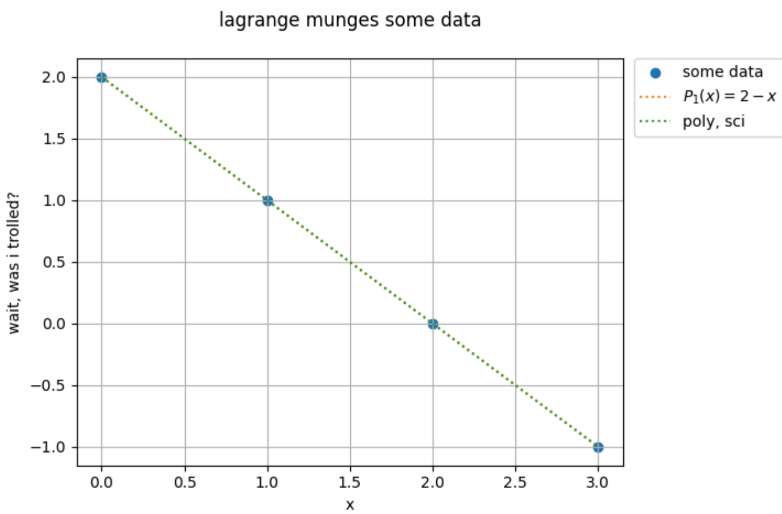
```
import matplotlib.pyplot as plt
import numpy as np
from numpy.polynomial.polynomial import Polynomial
import scipy as sp

#https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.lagrange.html

def main():
  # known
  input = np.array([[0,2],[1,1],[2,0],[3,-1]])
```

```
as object:
          3
2.776e-17 x - 1 x + 2

as array of coeffs:
[ 2.00000000e+00 -1.00000000e+00  0.00000000e+00  2.77555756e-17]
```



2 newtons divided difference

newtons divided difference gives a simple way to achieve the interpolating polynomial. given n data points, its result will be a polynomial of degree $d = n - 1$. bc of theorem 02, it is the same polynomial achieved by lagrange.

definition 03

denote by $f[x_1 \dots x_n]$ the coefficient of the x^{n-1} term in the unique polynomial that interpolates $(x_1, (f(x_1)), \dots, (x_n, f(x_n))$.

continuing example 01,

$$\begin{matrix} f(0) = 1 \\ f(2) = 2 \\ f(3) = 4 \end{matrix} \quad \xrightarrow{\text{by uniqueness}} \quad \frac{1}{2} = f[0\ 3\ 2] = f[3\ 0\ 2] = \dots \quad \text{usw}$$

newtons divided difference formula

$$\left. \begin{aligned} y = P(x) = N(x) = f[x_1] \\ &+ f[x_1\ x_2] \cdot (x - x_1) \\ &+ f[x_1\ x_2\ x_3] \cdot (x - x_1)(x - x_2) \\ &+ f[x_1\ x_2\ x_3\ x_4] \cdot (x - x_1)(x - x_2)(x - x_3) \\ &+ \cdots \\ &+ f[x_1\ \dots\ x_n] \cdot (x - x_1)\ \cdots\ (x - x_{n-1}) \end{aligned} \right\} \text{unique interpolationg polynomial}$$

$$\left. \begin{aligned} f[x_k] &= f(x_k) = y_k \\ f[x_k\ x_{k+1}] &= \frac{f[x_{k+1}] - f[x_k]}{x_{k+1} - x_k} \\ f[x_k\ x_{k+1}\ x_{k+2}] &= \frac{f[x_{k+1}\ x_{k+2}] - f[x_k\ x_{k+1}]}{x_{k+2} - x_k} \\ f[x_k\ x_{k+1}\ x_{k+2}\ x_{k+3}] &= \frac{f[x_{k+1}\ x_{k+2}\ x_{k+3}] - f[x_k\ x_{k+1}\ x_{k+2}]}{x_{k+3} - x_k} \\ &\dots \end{aligned} \right\} \text{its coefficients}$$

algorithm **newtons divided differences**

given $x = [x_1, \dots, x_n], y = [y_1, \dots, y_n]$.

```
for j = 1,...,n
  f[x(j)] = y(j)
end

for i = 2,...,n
  for j = 1,...,n+1-i
    f[x(j),...,x(j+i-1)] = (f[x(j+1) ... x(j+i-1)] - f[x(j) ... x(j+i-2)])/(x(j+i-1)-x(j))
  end
end
```

$$\Rightarrow P(x) = \sum_{i=1}^{n} f[x_1 \ \dots \ x_i](x-x_1)\dots(x-x_{i-1})$$

$$\begin{array}{lll}
x_1 \mapsto f[x_1] & & \\
 & f[x_1 \ x_2] & \\
x_2 \mapsto f[x_2] & & f[x_1 \ x_2 \ x_3] \\
 & f[x_2 \ x_3] & \\
x_3 \mapsto f[x_3] & &
\end{array}$$

where top edge terms of triangle are the coefficients, surprise.

example 03

continuing example 01,

$$\begin{array}{lll}
0 \mapsto f(0) = f[x_1] = 1 & & \\
 & f[x_1 \ x_2] = \dfrac{2-1}{2-0} = \dfrac{1}{2} & \\
2 \mapsto f(2) = f[x_2] = 2 & & f[x_1 \ x_2 \ x_3] = \dfrac{2-\frac{1}{2}}{3-0} = \dfrac{1}{2}. \\
 & f[x_2 \ x_3] = \dfrac{4-2}{3-2} = 2 & \\
3 \mapsto f(3) = f[x_3] = 4 & &
\end{array}$$

$P(x) = 1 + \frac{1}{2}\cdot(x-0) + \frac{1}{2}\cdot(x-0)(x-2) = \frac{1}{2}x^2 - \frac{1}{2}x + 1. \quad \checkmark$

example 04

add fourth data point $(1,0)$ to previous example.

$$\begin{array}{llll}
0 \mapsto 1 & & & \\
 & \dfrac{2-1}{2-0} = \dfrac{1}{2} & & \\
2 \mapsto 2 & & \dfrac{2-\frac{1}{2}}{3-0} = \dfrac{1}{2} & \\
 & \dfrac{4-2}{3-2} = 2 & & \dfrac{0-\frac{1}{2}}{1-0} = -\dfrac{1}{2} \\
3 \mapsto 4 & & \dfrac{2-2}{1-2} = 0 & \\
 & \dfrac{0-4}{1-3} = 2 & & \\
1 \mapsto 0 & & &
\end{array}$$

$$\begin{aligned}
P_3(x) &= 1 + \frac{1}{2}\cdot(x-0) + \frac{1}{2}\cdot(x-0)(x-2) - \frac{1}{2}\cdot(x-0)(x-2)(x-3) \\
&= P_2(x) - \frac{1}{2}\cdot(x-0)(x-2)(x-3).
\end{aligned}$$

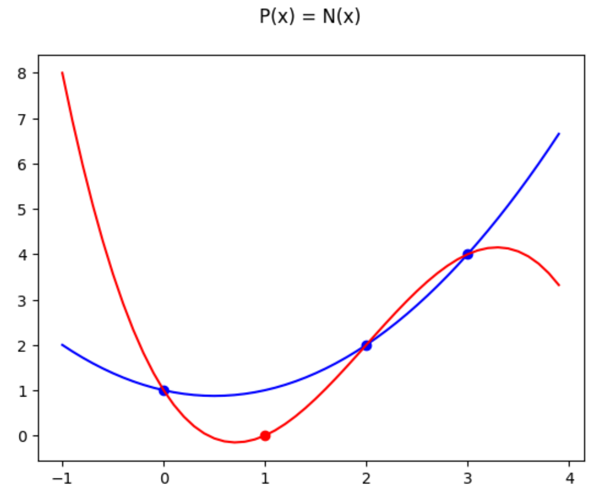
code

```
import matplotlib.pyplot as plt
import numpy as np

#https://pythonnumericalmethods.berkeley.edu/notebooks/chapter17.05-Newtons-Polynomial-Interpolation.html

def divided_diff(x, y):
  '''
  function to calculate the divided
  differences table
  '''
```

```
n = len(y)
coef = np.zeros([n, n])
# the first column is y
```



example 05

apply newtons divided difference to example 02.

$(0,2),(1,1),(2,0),(3,-1)$

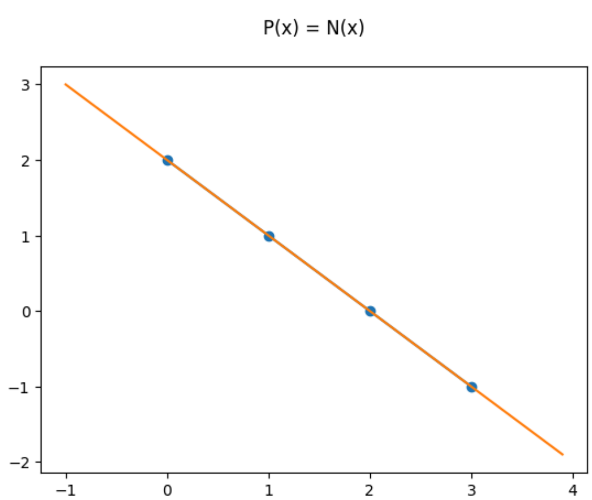
$$
\begin{array}{lllll}
0 \mapsto & 2 & & & \\
 & & -1 & & \\
1 \mapsto & 1 & & 0 & \\
 & & -1 & & 0 \\
2 \mapsto & 0 & & 0 & \\
 & & -1 & & \\
3 \mapsto & -1 & & &
\end{array}
$$

$\Rightarrow P(x) = 2 + (-1)(x-0) = 2 - x. \quad \checkmark$

code

```
import matplotlib.pyplot as plt
import numpy as np

def main():
  xs = np.array([0,1,2,3])
  ys = np.array([2,1,0,-1])
  # get the divided difference coef
```



3 how many degree d polynomials pass through n points?

theorem 02 states can be only one polynomial of degree $d \leq n-1$ that passes through n points. how many of degree n?

adding another points adds another degree but thats cheating? the new polynomial still interpolates the original n points and now it has degree of at least n. however, that extra point can be added in as many ways as there are numbers to add.

$$P_n(x) = P_{n-1}(x) + cx(x-x_1)(x-x_2)\dots(x-x_n), \quad c \neq 0.$$

so how many polynomials of degree n can interpolate n data points? :D

example 06

how many polynomials of each degree $0 \leq d \leq 5$ pass through points $(-1,-5),(0,-1),(2-1),(3,11)$?

$$\begin{array}{rcccc}
-1 \mapsto -5 & & & & \\
 & 4 & & & \\
0 \mapsto -1 & & -1 & & \\
 & 1 & & 1 & \\
2 \mapsto \quad 1 & & 3 & & \\
 & 10 & & & \\
3 \mapsto \ 11 & & & & \\
\end{array}$$

$$\Downarrow$$

$$\begin{aligned}
P_3(x) &= -5 + 4(x+1) - 1(x+1)(x-0) + 1(x+1)(x-0)(x-2) \\
&= x^3 - 2x^2 + x - 1 \sim \text{only polynomial of degree } d \leq n-1 = 4-1 = 3.
\end{aligned}$$

$$P_4(x) = P_3(x) + c_1 x(x+1)(x-0)(x-2)(x-3), \quad c_1 \neq 0 \sim \infty\text{-ly many.}$$

$$P_5(x) = P_3(x) + c_2 x^2(x+1)(x-0)(x-2)(x-3), \quad c_2 \neq 0 \sim \infty\text{-ly many.}$$

4 compression

what does interpolation compress? the degree $n-1$ polynomial characterized by n coefficients is a "compressed" version of $f(x)$. eg, "$sin\ x$" is stored computationally as coefficients and its calculation relies on interpolation. this type of compression is "lossy compression" bc error will occur as sine is not a polynomial.

5 representing functions by approximating polynomials

a major use of polynomial interpolation is to replace evaluation of a complicated function by evaluation of a polynomial - which involves only elementary operations. consider this simplification as a form of compression.

example 07

interpolate $f(x) = sin\ x$ at four equally spaced points on $[0, \frac{\pi}{2}]$.

$$\begin{array}{rcccc}
0 \mapsto 0.0000 & & & & \\
 & 0.9549 & & & \\
\frac{\pi}{6} \mapsto 0.5000 & & -0.2443 & & \\
 & 0.6990 & & -0.1139 & \\
\frac{\pi}{3} \mapsto 0.8660 & & -0.4232 & & \\
 & 0.2559 & & & \\
\frac{\pi}{2} \mapsto 1.0000 & & & & \\
\end{array}$$

$$\begin{aligned}
\Rightarrow P_3(x) &= 0 + 0.9549x - 0.2443x(x-\frac{\pi}{6}) - 0.1139x(x-\frac{\pi}{6})(x-\frac{\pi}{3}) \\
&= 0 + x(0.9549 + (x-\frac{\pi}{6})(-0.2443 + (x-\frac{\pi}{3})(-0.1139))).
\end{aligned}$$

code

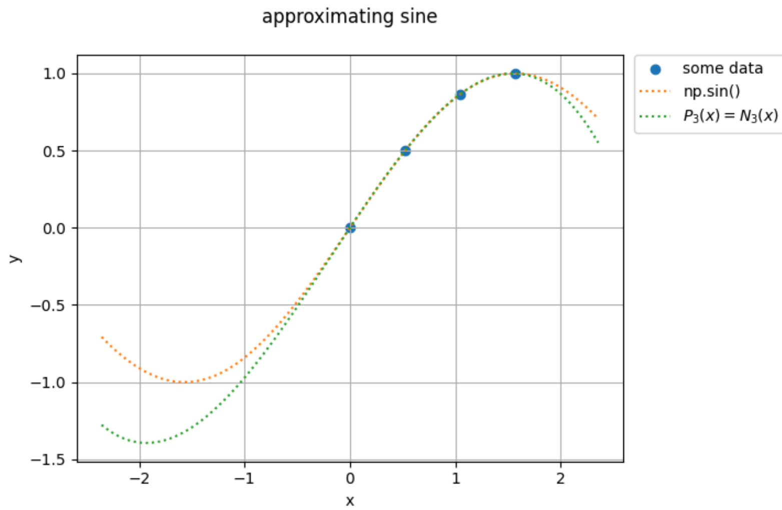
```
import matplotlib.pyplot as plt
import numpy as np

p = lambda x: x*(0.9549 + (x - np.pi/6)*(-0.2443 + (x - np.pi/3)*(-0.1139)))
```

```python

def main():
  # known
  xs = np.linspace(0,np.pi/2,3+1)

```




```python
from tabulate import tabulate

def main():⋯

if __name__ == "__main__":⋯
```

```
|   x (rad) |   np.sin() |   $N_3,mod$ |       error |
|-----------|------------|-------------|-------------|
|         1 |   0.841471 |    0.841076 | 0.000394766 |
|         2 |   0.909297 |    0.910169 | 0.000871189 |
|         3 |   0.14112  |    0.142842 | 0.0017216   |
|         4 |  -0.756802 |   -0.755661 | 0.0011416   |
|        14 |   0.990607 |    0.992824 | 0.00221669  |
|      1000 |   0.82688  |    0.826294 | 0.000585579 |
```