# INTRODUCTION TO COMPUTATIONAL PHYSICS

## First Coursework

U24200 –Academic Session 2019–2020

**Instructions**

a) This is worth **50%** of your total mark for this unit.

b) **You must undertake this assignment individually.**

c) Submission method: **by Moodle through the available dropbox**.

d) Submission deadline: **January 13, 2019**

## 1 Introduction

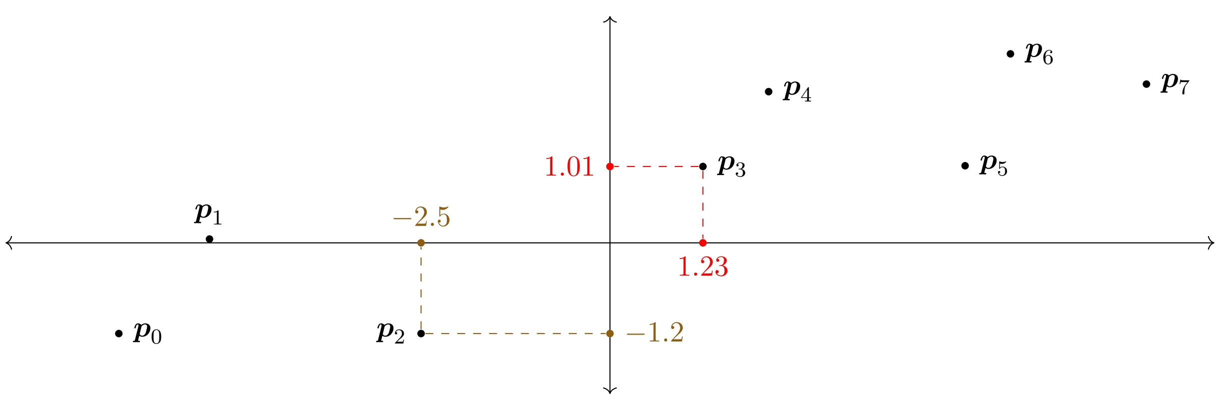
**Interpolation** consists in building a simple function $f$ (here, a *polynomial*, $f(x) = a_0 + a_1 x + \cdots + a_k x^k$) whose graph curve passes through a given set of points

$$\boldsymbol{p}_0 = (x_0, y_0), \quad \boldsymbol{p}_1 = (x_1, y_1), \quad \ldots, \quad \boldsymbol{p}_m = (x_m, y_m),$$

i.e. such that $f(x_0) = y_0,\ f(x_1) = y_1,\ \ldots,\ f(x_m) = y_m$. This can be seen, for instance in the case in which

$$\begin{array}{llll} \boldsymbol{p}_0 = (-6.5, -1.2), & \boldsymbol{p}_1 = (-5.3, 0.05), & \boldsymbol{p}_2 = (-2.5, -1.2), & \boldsymbol{p}_3 = (1.23, 1.01), \\ \boldsymbol{p}_4 = (2.1, 2), & \boldsymbol{p}_5 = (4.7, 1.02), & \boldsymbol{p}_6 = (5.3, 2.5), & \boldsymbol{p}_7 = (7.1, 2.1). \end{array}$$

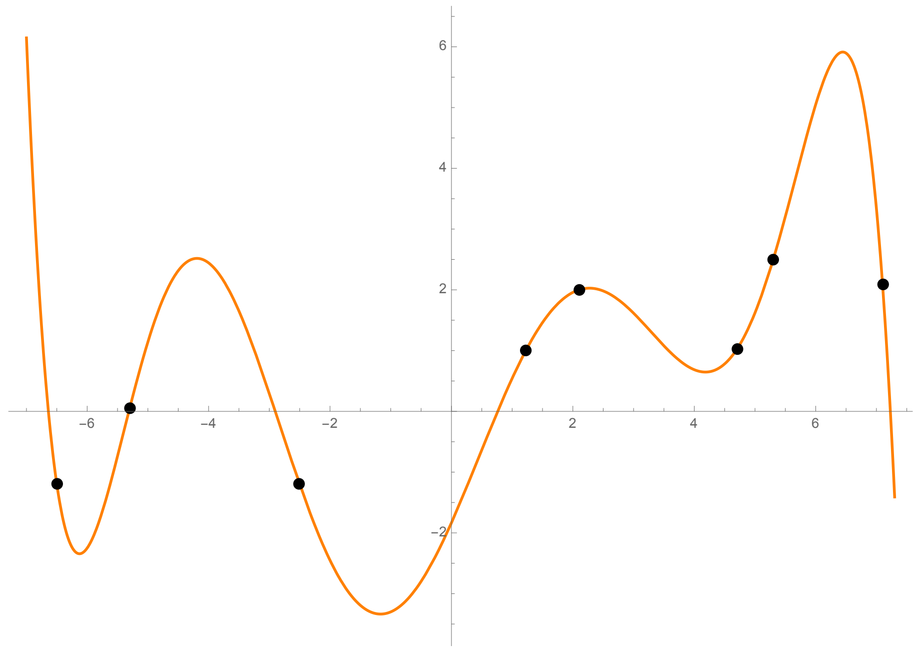
Let us first represent these points graphically:




We wish to find a polynomial whose graph traverses each one of these points. After you have finished your program, you will find that such a function can be approximated as

$$f(x) = -0.000175066x^7 + 0.000289133x^6 + 0.014541x^5 - 0.0211649x^4 - 0.34218x^3 + 0.477095x^2 + 2.24967x - 1.83489,$$

and has the shape shown in the next page. We draw it along with the original points $\boldsymbol{p}_i$.




The following result is fundamental to our purpose:

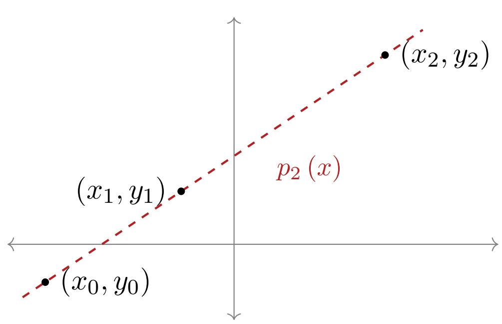
**Theorem** (**Existence and uniqueness of the interpolating polynomial**)**.** *Let* $(x_0, y_0), \ldots, (x_n, y_n)$ *be* $n+1$ *points in the plane such that* $x_i$ *are pairwise different* ($x_i \neq x_j$ *if* $i \neq j$)*. Then there exists a unique polynomial of degree* at most $n$, $p_n(x) = a_0 + a_1 x + \cdots + a_n x^n$, *interpolating these points, i.e. such that*

$$p(x_0) = y_1, \quad p(x_1) = y_2, \quad , \ldots, \quad p(x_n) = y_n. \tag{1}$$

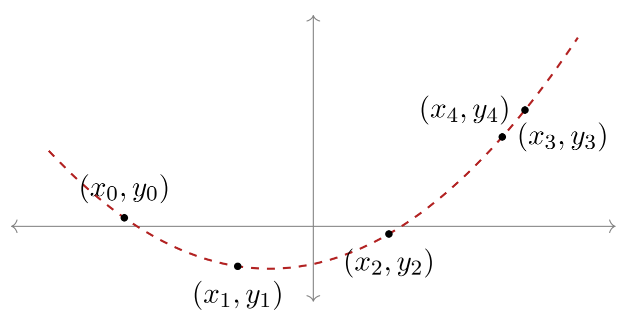
**Remarks.**

1. A well-known fact in Geometry, namely that any two points are traversed simultaneously by a unique line, is a particular case of the above: a line is the graph of a <u>degree-one</u> polynomial $y = ax + b$, and using the above two notation we would have $\underline{n = 1}$ for two points $(x_0, y_0), (x_1, y_1)$.

2. $n$ is the *maximum* value (hence an upper bound) of the degree of the interpolating polynomial but the actual degree could be less than $n$ depending on the disposition of the points. For instance,




Three points hence $n = 2$ but they are *aligned*, thus interpolating polynomial is that line: $p(x) = a + bx + \boxed{0}\,x^2$




Five points ($n = 4$) but they are *all in the same parabola*, thus interpolating polynomial is that parabola: $p_4(x) = A + Bx + Cx^2 + D\,\boxed{0}\,x^3 + E\,\boxed{0}\,x^4$

3. Interpolation (and a similar concept called *extrapolation*) will be useful whenever you are given a table of experimental data and need to guess theoretical outputs for values not belonging to that table.

There are many ways of computing the interpolating polynomial $p_n$ of $n+1$ points (but remember: the polynomial, due to its uniqueness, *is still the same* and depends only on the points chosen). Most notably:

- solving the linear system defined by the interpolating conditions
- method of Lagrange polynomials;
- Newton's method of divided differences;
- Aitken's method, Neville's method, etc.

We will see the third method. The first two are mostly of theoretical utility but we will give you an example of application of the second one to further illustrate the uniqueness of the polynomial for each table.

## 2 Lagrange polynomials

Let $x_0, \ldots, x_n$ be $n+1$ pairwise different abscissae. For every $k = 0, 1, \ldots, n$, the $k^{\text{th}}$ **Lagrange polynomial** linked to the abscissae $x_0, \ldots, x_n$ is

$$L_k = \prod_{i=0,\ i\neq k}^{n} \frac{x - x_i}{x_k - x_i} = \frac{(x - x_0) \cdots (x - x_{k-1})\,(x - x_{k+1}) \cdots (x - x_n)}{(x_k - x_0) \cdots (x_k - x_{k-1})\,(x_k - x_{k+1}) \cdots (x_k - x_n)}$$

Then, the interpolating polynomial for the table $\{(x_0, y_0), \ldots, (x_n, y_n)\}$ is

$$p_n(x) = y_0 L_0(x) + y_1 L_1(x) + \cdots + y_n L_n(x).$$

**Example.** Assume we want to find the interpolating polynomial for the following table:

| $x$ | 1 | 2 | 4 | 8 | 15 |
|---|---|---|---|---|---|
| $f(x)$ | $-0.5$ | 0.4 | 0.9 | 1.5 | 1.9 |

(2)

The Lagrange polynomials are:

$$\begin{aligned}
L_0(x) &= \frac{(x-2)(x-4)(x-8)(x-15)}{(1-2)(1-4)(1-8)(1-15)} = \frac{(x-2)(x-4)(x-8)(x-15)}{294}, \\
L_1(x) &= \frac{(x-1)(x-4)(x-8)(x-15)}{(2-1)(2-4)(2-8)(2-15)} = -\frac{(x-1)(x-4)(x-8)(x-15)}{156}, \\
L_2(x) &= \frac{(x-1)(x-2)(x-8)(x-15)}{(4-1)(4-2)(4-8)(4-15)} = \frac{(x-1)(x-2)(x-8)(x-15)}{264}, \\
L_3(x) &= \frac{(x-1)(x-2)(x-4)(x-15)}{(8-1)(8-2)(8-4)(8-15)} = -\frac{(x-1)(x-2)(x-4)(x-15)}{1176}, \\
L_4(x) &= \frac{(x-1)(x-2)(x-4)(x-8)}{(15-1)(15-2)(15-4)(15-8)} = \frac{(x-1)(x-2)(x-4)(x-8)}{14014}
\end{aligned}$$

and the interpolating polynomial is

$$\begin{aligned}
p_4(x) &= (-0.5)\frac{(x-2)(x-4)(x-8)(x-15)}{294} - (0.4)\frac{(x-1)(x-4)(x-8)(x-15)}{156} + (\ldots) \\
&- (1.5)\frac{(x-1)(x-2)(x-4)(x-15)}{1176} + (1.9)\frac{(x-1)(x-2)(x-4)(x-8)}{14014} \\
&= -2.18962 + 2.18947x - 0.55636x^2 + 0.0585058x^3 - 0.00199562x^4, \qquad (3)
\end{aligned}$$

or $p_4(x) = -\frac{153427}{70070} + \frac{306833}{140140}x - \frac{6683}{12012}x^2 + \frac{8199}{140140}x^3 - \frac{839}{420420}x^4$ if you had been working with exact rational amounts (i.e. replacing $-0.5$ by $-\frac{1}{2}$, 0.4 by $\frac{2}{5}$, etc in the original table). In general, you will not have the easy option to convert to rational form and you will need to work with `float` as done in (3).

## 3 Newton's divided differences

Assume $p_n$ interpolates $\{(x_k, y_k) : 0 \leq k \leq n\}$ and we wish to find $p_n(x)$ for different values of $x$ (for instance, $p_n(3), p_n(3.4), p_n(6.7)$). The method described in §2 would make this difficult unless we perform the final expansion (3); otherwise we would need to compute new Lagrange polynomials for each value of $x$. If, however, we could fix some coefficients $A_0, A_1, \ldots, A_n$ *depending only on the fixed abscissae* $x_0, \ldots, x_n$ *and not on the mobile point* $x$ and could fit them in an expression

$$\begin{aligned} p_n(x) &= A_0 + A_1(x-x_0) + A_2(x-x_0)(x-x_1) + A_3(x-x_0)(x-x_1)(x-x_2) + \cdots + \\ &+ A_n(x-x_0)(x-x_1)\cdots(x-x_{n-1}) \end{aligned} \tag{4}$$

this would make it easier for us to work out $p_n(x)$ in *any* $x$ using a tool that reduces the number of operations: **generalised Horner's method**: start with $U_n = A_n$ and perform the following:

$$U_k = U_k(x - x_{k-1}) \quad \text{and} \quad U_{k-1} = U_k + A_{k-1}, \qquad k = n, \ldots, 1. \tag{5}$$

For instance, for $n = 3$ we would have

$$p_3(x) = \overbrace{A_0 + (x-x_0)\overbrace{\left\{A_1 + (x-x_1)\overbrace{\left[A_2 + \overbrace{(x-x_2)A_3}^{U_3}\right]}^{U_2}\right\}}^{U_1}}^{U_0}.$$

**Newton's method of divided differences** provides an expression of the form (4). Given a table $\{(x_k, y_k) : k = 0, \ldots, n\}$, $x_0, \ldots, x_m$ pairwise different, we recursively define the **divided differences** as:

$$\begin{aligned} \boxed{y[x_k]} &= \boxed{y_k} = y_k, \\ \boxed{y[x_k, x_{k+1}]} &= \boxed{y_{k,k+1}} = \frac{y_{k+1} - y_k}{x_{k+1} - x_k}, \\ \boxed{y[x_k, x_{k+1}, x_{k+2}]} &= \boxed{y_{k,k+1,k+2}} = \frac{y_{k+1,k+2} - y_{k,k+1}}{x_{k+2} - x_k}, \\ &\vdots \\ \boxed{y[x_k, \ldots, x_{k+m}]} &= \boxed{y_{k,k+1,\ldots,k+m}} = \frac{y_{k+1,\ldots,k+m} - y_{k,\ldots,k+m-1}}{x_{k+m} - x_k}. \end{aligned}$$

Both notations (boxed or double-boxed) are equally correct. These provide the $A_0, \ldots, A_n$ in (4) for $p_n$:

**Theorem.** *The interpolating polynomial for a table* $\{(x_k, y_k) : 0 \leq k \leq n\}$ *is*

$$\begin{aligned} p_n(x) &= y[x_0] + y[x_0, x_1](x-x_0) + y[x_0, x_1, x_2](x-x_0)(x-x_1) \\ &+ y[x_0, x_1, x_2, x_3](x-x_0)(x-x_1)(x-x_2) + \cdots + \\ &+ y[x_0, x_1, \ldots, x_n](x-x_0)\cdots(x-x_{n-1}) \end{aligned} \tag{6}$$

**Example.** We return to the table in (2). The divided difference scheme if we use pen and paper is

| $k$ | $x_k$ | $y_k$ | $y_{k-1,k}$ | $y_{k-2,k-1,k}$ | $y_{k-3,k-2,k-1,k}$ | $y_{k-4,\ldots,k}$ |
|---|---|---|---|---|---|---|
| 0 | 1 | $-\frac{1}{2}$ | | | | |
| | | | $\frac{\frac{4}{10}-\left(-\frac{1}{2}\right)}{2-1}=\frac{9}{10}$ | | | |
| 1 | 2 | $\frac{4}{10}$ | | $\frac{\frac{1}{4}-\frac{9}{10}}{4-1}=-\frac{13}{60}$ | | |
| | | | $\frac{\frac{9}{10}-\frac{4}{10}}{4-2}=\frac{1}{4}$ | | $\frac{1}{35}$ | |
| 2 | 4 | $\frac{9}{10}$ | | $\frac{\frac{3}{20}-\frac{1}{4}}{8-2}=-\frac{1}{60}$ | | $-\frac{839}{420420}$ |
| | | | $\frac{\frac{3}{2}-\frac{9}{10}}{8-4}=\frac{3}{20}$ | | $\frac{19}{30030}$ | |
| 3 | 8 | $\frac{3}{2}$ | | $\frac{\frac{2}{35}-\frac{3}{20}}{15-4}=-\frac{13}{1540}$ | | |
| | | | $\frac{\frac{19}{10}-\frac{3}{2}}{15-8}=\frac{2}{35}$ | | | |
| 4 | 15 | $\frac{19}{10}$ | | | | |

(7)

and we keep the elements in the upper diagonal row. The polynomial is

$$\begin{aligned}
p_4(x) &= y_0 + y_{0,1}(x-x_0) + y_{0,1,2}(x-x_0)(x-x_1) + y_{0,1,2,3}(x-x_0)(x-x_1)(x-x_2) \\
&\quad + y_{0,1,2,3,4}(x-x_0)(x-x_1)(x-x_2)(x-x_3) \\
&= -\frac{1}{2}+\frac{9}{10}(x-1)-\frac{13}{60}(x-1)(x-2)+\frac{1}{35}(x-1)(x-2)(x-4)-\frac{839}{420420}(x-1)(x-2)(x-4)(x-8) \\
&= -\frac{153427}{70070}+\frac{306833}{140140}x-\frac{6683}{12012}x^2+\frac{8199}{140140}x^3-\frac{839}{420420}x^4 \qquad (8)
\end{aligned}$$

Unsurprisingly, the same polynomial we found for this given set of points using Lagrange's method. If we want to compute $p_4(x)$ for any $x$, if we use (8) we need to carry out 14 $+/-$, 10 $*$. However, using Horner,

$$p_4(x) = -\frac{1}{2}+(x-1)\left\{\frac{9}{10}+(x-2)\left[-\frac{13}{60}+(x-4)\left(\frac{1}{35}-\frac{839}{420420}(x-8)\right)\right]\right\}$$

which only requires eight sums or subtractions and four products.

Again, in general (**and more specifically in this coursework**) you will not have the luxury of working with pen and paper, thus you will need to keep numbers in `float` decimal point form:

| $x_k$ | $y_k$ | $y_{k-1,k}$ | $y_{k-2,k-1,k}$ | $y_{k-3,k-2,k-1,k}$ | $y_{k-4,\ldots,k}$ |
|---|---|---|---|---|---|
| `1` | $\boxed{\texttt{-0.5}}$ | | | | |
| | | $\boxed{\frac{0.4-(-0.5)}{2-1}=\texttt{0.9}}$ | | | |
| `2` | `0.4` | | $\boxed{\frac{0.25-0.9}{4-1}\simeq\texttt{-0.216667}}$ | | |
| | | $\frac{0.9-0.4}{4-2}=\texttt{0.25}$ | | $\boxed{\texttt{0.0285714}}$ | |
| `4` | `0.9` | | $\frac{0.15-0.25}{8-2}\simeq\texttt{-0.0166667}$ | | $\boxed{\texttt{-0.00199562}}$ |
| | | $\frac{1.5-0.9}{8-4}=\texttt{0.15}$ | | `0.000632701` | |
| `8` | `1.5` | | $\frac{0.0571429-0.15}{15-4}\simeq\texttt{-0.00844155}$ | | |
| | | $\frac{1.9-1.5}{15-8}\simeq\texttt{0.0571429}$ | | | |
| `15` | `1.9` | | | | |

(9)

The boxed terms in (9) are the ones we keep for the polynomial. Note that regardless of whether you use finite precision (9) or the exact rational values (7), you need to compute *the entire table* to retrieve that upper row of boxed items. Now for every $x$, the value of $p_4(x)$ at $x$ is

$$p_4(x) = -0.5+(x-1)\cdot(0.9+(x-2)\cdot(-0.216667+(x-4)\cdot(0.0285714+(x-8)\cdot(-0.00199562))))$$

which, if expanded, has the same form $p_4 = -2.1896249 + 2.1894750x + \ldots$ as in (3). Instead of expanding it, however, if we wanted to compute it for different values of $x$ we would use Horner (5):

$$\begin{aligned}
U_4 &= -0.00199562, \quad U_4 = U_4(x-8), \quad U_3 = U_4 + 0.0285714, \quad U_3 = U_3(x-4), \\
U_2 &= U_3 - 0.216667, \quad U_2 = U_2(x-2), \quad U_1 = U_2 + 0.9, \quad U_1 = U_1(x-1), \quad U_0 = U_1 - 0.5
\end{aligned}$$

and the value of $p_4(x)$ for that particular $x$ would be $U_0$.

## 4 Coursework exercises

**1.** These will be the functions used to compute $p_n$:

(i) Write up a function `Newton_differences` with the following arguments:

- an array of abscissae $x_0, \ldots, x_n$ that have been previously checked to be pairwise different;
- an array of ordinates $y_0, \ldots, y_n$ comprising the other half of the table we wish to interpolate.

and returning the array of divided differences $(y_0, y_{0,1} \ldots, y_{0,\ldots,n})$.

(ii) Write up a function `Horner` with the following arguments:

- an array of abscissae $x_0, \ldots, x_n$ that have been previously checked to be pairwise different;
- an array of terms $A_0, \ldots, A_n$,
- and a variable floating-point number $x$,

and returning the output $A_0 + (x - x_0)(A_1 + (x - x_1)(A_2 + \ldots))$ in the manner described in (5).

(iii) Combine Exercises (i) and (ii) above to interpolate a given set of points and compute $p_n(x)$ for any given $x$. Think of possible ways to minimise the number of operations used.

**2.** And these constitute an example of how to apply Exercise 1:

(i) Create a text file `table.txt` and fill it with pairs of numbers distributed in two columns:

$$\left.\begin{array}{ll} \texttt{x0} & \texttt{y0} \\ \texttt{x1} & \texttt{y1} \\ \vdots & \vdots \end{array}\right\} \quad \text{make sure } \texttt{x0, x1,...} \text{ are all different} \tag{10}$$

(ii) Write a function `read_from_file` with two arguments: a file `name` (in string form) and a tolerance `tol`. The function will read two-column data such as (10) externally from file `name` and will check that there are no abscissae `x0`, `x1`, ... (in any order) differing from each other by less than `tol` in absolute value.

(iii) Write a function `write_to_files` one of whose arguments must be the number of points of the interpolating polynomial you wish to plot. It will write these points, again in the same disposition as (10), in a new file called `function_to_plot.txt`. It will also store Newton's divided differences in another file named `divided_differences.txt`.

All that is needed for you to upload to Moodle is:

- One Python file `UPXXXXXX.py` containing your student number.
- One Jupyter library showcasing applications of (and any necessary comments about) your Python file.
- One file `table.txt` containing a sample table to interpolate.
- One file `function_to_plot.txt` with a larger ($> 1000$) set of points interpolating those in `table.txt`.
- One file `divided_differences.txt` containing the divided differences for the data in `table.txt`.
- A plot of the data in `function_to_plot.txt` (you can skip this if your Jupyter file contains plots).