UCLouvain

LINMA2171 - Numerical Analysis: Approximation, Interpolation, Integration

Homework 1

# Hermite Interpolation

***Student***
Lucas Ahou (35942200)

***Teacher***
P-A. Absil

October 01, 2025

# Question 1

We need to show that $q_{2n+1}(x_i) = f(x_i)$ and $q'_{2n+1}(x_i) = f'(x_i)$ for $i = 0, ..., n$ :

$\underline{q_{2n+1}(x_i) = f(x_i)}$**:**

Knowing that the Lagrange polynomials equals to 1 on the node it is defined on and cancels on every other node, we can compute the following :

$$\begin{aligned} H_i(x_i) &= L_i^2(x_i) = 1 \\ H_j\left(x_{i\neq j}\right) &= 0 \\ K_i(x_i) &= K_j\left(x_{i\neq j}\right) = 0 \end{aligned}$$

With that in mind, we find that $q_{2n+1}$ evaluated at $x_i$ reduces to :

$$\begin{aligned} q_{2n+1}(x_i) &= \sum_{j=0}^{n} f\left(x_j\right)H_j(x_i) + \sum_{j=0}^{n} f'\left(x_j\right)K_j(x_i) \\ &= f(x_i) \end{aligned}$$

$\underline{q'_{2n+1}(x_i) = f'(x_i)}$**:**

Let's first compute the derivative of $H_i$ and $K_i$ :

$$\begin{aligned} H'_i(x) &= -2L'_i(x_i)L_i^2(x) + 2L_i(x)L'_i(x)(1 - 2L'_i(x_i)(x - x_i)) \\ K'_i(x) &= L_i^2(x) + 2(x - x_i)L_i(x)L'_i(x) \end{aligned}$$

When evaluated at the nodes $x_i$, we have :

$$\begin{cases} H'_i(x_i) = H_j\left(x_{i\neq j}\right) = 0 \\ K'_i(x_i) = 1 \\ K'_j\left(x_{i\neq j}\right) = 0 \end{cases}$$

Knowing that, we find that $q'_{2n+1}$ evaluated at $x_i$ reduces to :

$$\begin{aligned} q'_{2n+1}(x_i) &= \sum_{j=0}^{n} f\left(x_j\right)H_j{}'(x_i) + \sum_{j=0}^{n} f'\left(x_j\right)K'_j(x_i) \\ &= f'(x_i) \quad \square \end{aligned}$$

# Question 2

To interpolate $(x_i, f(x_i), f'(x_i), f''(x_i))_{i=0}^n$, we need to impose :

$$\begin{cases} \alpha_i(x_j) = \delta_{ij}, & \alpha_i'(x_j) = 0, & \alpha_i''(x_j) = 0 \\ \beta_i(x_j) = 0, & \beta_i'(x_j) = \delta_{ij}, & \beta_i''(x_j) = 0 \\ \gamma_i(x_j) = 0, & \gamma_i'(x_j) = 0, & \gamma_i''(x_j) = \delta_{ij} \end{cases} \tag{1}$$

$\forall j \in \{0, 1, ..., n\}$.
To achieve this, we can build them from the following polynomial :

$$q(x) = L_i^3(x)(a + bx + cx^2)$$

where $a, b, c \in \mathbb{R}$ are coefficients.
We have that $q \in \mathcal{P}_{3n+2}$, while still having enough degree of freedom to satisfy (1). Let's now compute the coefficients for the functions $\alpha_i, \beta_i$ and $\gamma_i$ :

**$\underline{\alpha_i(x)}$:**
First, let's compute the derivative of $q(x)$ :

$$\begin{aligned} & q(x) = L_i^3(x)(a + bx + cx^2) \\ \Rightarrow\ & q'(x) = 3L_i^2(x)L_i'(x)(a + bx + cx^2) + L_i^3(x)(b + 2cx) \\ \Rightarrow\ & q''(x) = 3\Big(2L_i(x)(L_i'(x))^2 + L_i^2(x)L_i''(x)\Big)(a + bx + cx^2) + 6L_i^2(x)L_i'(x)(b + 2cx) + 2cL_i^3(x) \end{aligned}$$

Now, let's find the coefficients to satisfy the constraints:

1. To satisfy $\alpha_i(x_j) = \delta_{ij}$, we can simply impose $\boldsymbol{a + bx_i + cx_i^2 = 1}$ because $L_i(x_j) = \delta_{ij} \forall j \in \{0, ..., n\}$.
2. To satisfy $\alpha_i'(x_j) = 0$, we can simply impose that $\alpha_i'(x_i) = 0$.
   We thus obtain $\boldsymbol{b + 2cx_i = -3L_i'(x_i)}$.
3. Finally, to satisfy $\alpha_i''(x_j) = 0$, we simply impose $\alpha_i''(x_i) = 0$.
   We thus obtain $\boldsymbol{3L_i''(x_i) - 12(L_i(x_i))^2 + 2c = 0}$

The system to solve is thus :

$$\begin{cases} a + bx_i + cx_i^2 = 1 \\ b + 2cx_i = -3L_i'(x_i) \\ 3L_i''(x_i) - 12(L_i(x_i))^2 + 2c = 0 \end{cases} \tag{2}$$

After solving, we obtain :

$$\begin{cases} a = 1 + 3x_iL_i'(x_i) + 6x_i^2(L_i'(x_i))^2 - \frac{3}{2}x_i^2L_i''(x_i) \\ b = -3L_i'(x_i) - 12x_i(L_i'(x_i))^2 + 3x_iL_i''(x_i) \\ c = 6(L_i'(x_i))^2 - \frac{3}{2}L_i''(x_i) \end{cases}$$

$\underline{\beta_i(x)}$**:**

Let's proceed in the same way for $\beta_i$:

1. To satisfy $\beta_i(x_j) = 0$, we can simply impose $\beta_i(x_i) = 0$.
   We thus obtain $\boldsymbol{a + bx_i + cx_i^2 = 0}$
2. To satisfy $\beta_i'(x_j) = \delta_{ij}$, we can simply impose $\beta_i(x_i) = 1$.
   We thus obtain $\boldsymbol{b + 2cx_i = 1}$
3. Finally, to satisfy $\beta_i''(x_j) = 0$, we simply impose $\beta_i''(x_i) = 0$.
   We thus obtain $\boldsymbol{6L_i'(x_i) + 2c = 0}$

After solving that, we obtain :

$$\begin{cases} a = -3x_i^2 L_i'(x_i) - x_i \\ b = 1 + 6x_i L_i'(x_i) \\ c = -3L_i'(x_i) \end{cases}$$

$\underline{\gamma_i(x)}$**:**

Finally, let's compute $\gamma_i$:

1. To satisfy $\gamma_i(x_j) = 0$, we can simply impose $\gamma_i(x_i) = 0$.
   We thus obtain $\boldsymbol{a + bx_i + cx_i^2 = 0}$
2. To satisfy $\gamma_i'(x_j) = 0$, we can simply impose $\gamma_i(x_i) = 0$.
   We thus obtain $\boldsymbol{b + 2cx_i = 0}$
3. Finally, to satisfy $\gamma_i''(x_j) = \delta_{ij}$, we simply impose $\gamma_i''(x_i) = 1$.
   We thus obtain $\boldsymbol{2c = 1}$

After solving that, we obtain :

$$\begin{cases} a = \frac{1}{2}x_i^2 \\ b = -x_i \\ c = \frac{1}{2} \end{cases}$$

# Question 3

Let's now implement the Hermite interpolation $p_{3n+2}$ defined before on $f(x) = e^{-x^2/2}$ on the interval $[-5, 5]$ in Python. Here are the plots of the interpolation with 1, 4, 9 and 15 nodes respectively[1]:

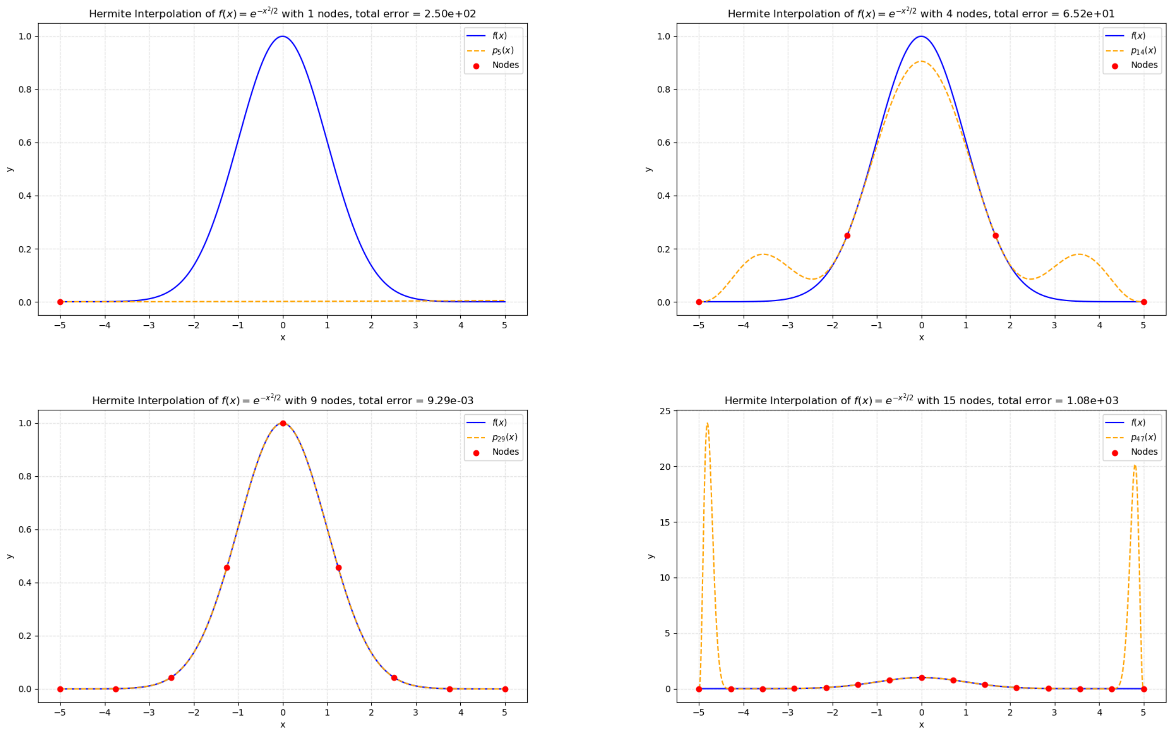


Figure 1 – Hermite interpolation of $f(x) = e^{-x^2/2}$ on $[-5, 5]$ with 1, 4, 9 and 15 nodes

As expected, a too small number of nodes (like 1 and 3) will result in a bad interpolation of the original function. However, too many nodes makes the interpolation polynomial explode on the end nodes. This is due to the fact that the polynomial is of order $3n + 2$, which can become quite large even for moderate values of $n$. This results in very high-degree polynomials as $n$ increases, which in return causes high oscillations.

Below, you will find a plot of the total errors of the interpolation, computed as an approximation of $\int_{-5}^{5} |f(x) - p_{3n+2}(x)| \,\mathrm{d}x$, with respect to $n$:

[1] The nodes are uniformly spaced with both ends of the interval included

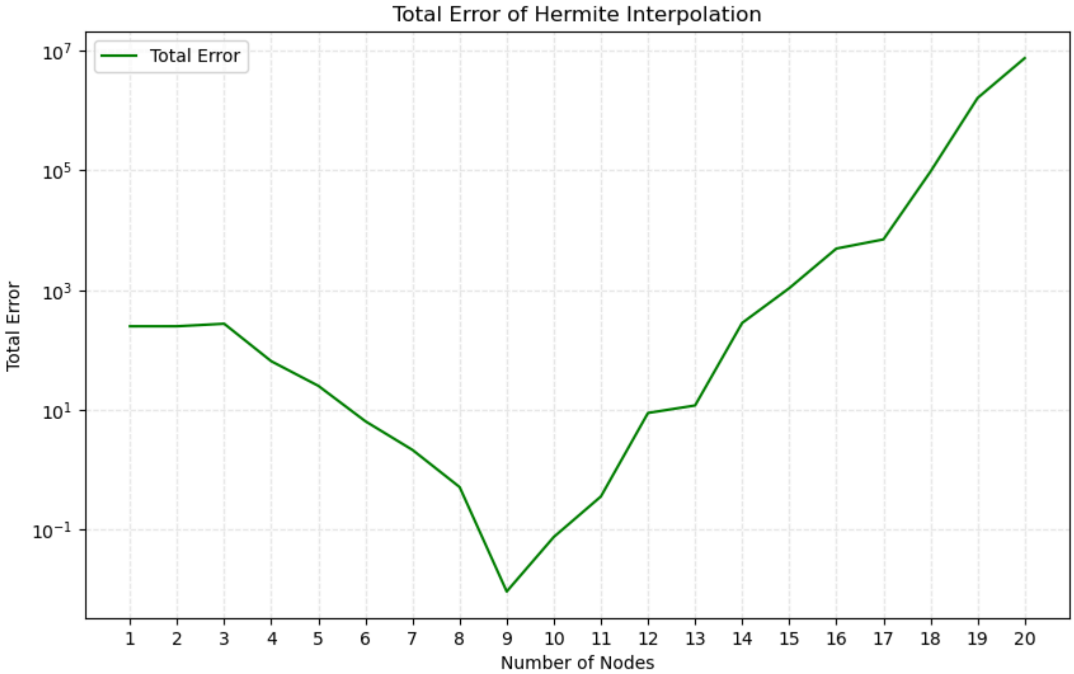


Figure 2 – Total interpolation error with respect to the number of nodes

As we can see, the error decreases as $n$ increases, but starts to increase again after a certain point. This is due to the Runge's phenomenon[2], which states that increasing the number of interpolation nodes can lead to worse approximations for high-degree polynomials. In this case, the error starts to increase again after $n = 9$. This is because the polynomial starts to oscillate wildly between the nodes, leading to large errors in the approximation. Thus, we can deduce that there is an optimal number of nodes that minimizes the interpolation error, and adding more nodes beyond that point can actually worsen the approximation.

[2] *https://en.wikipedia.org/wiki/Runge%27s_phenomenon*