Annales Mathematicae et Informaticae
***45** (2015) pp. 151–160*
http://ami.ektf.hu

Exploring Hermite interpolation polynomials using recursion*

Róbert Vajda

Bolyai Institute, University of Szeged
vajdar@math.u-szeged.hu

Submitted February 11, 2015 — Accepted December 16, 2015

Abstract

In this paper we consider the teaching of Hermite interpolation. We propose here two nonstandard approaches for exploring Hermite interpolation polynomials in a computer supported environment.

As an extension to the standard construction of the interpolation polynomials based on either on the fundamental polynomials or the triangular shaped divided difference table, we first investigate the generalization of the Neville type recursive scheme which may be familiar to the reader or to the student from the chapter about Lagrangian interpolation.

Second, we propose an interactive demo tool where by the step-by-step construction of the interpolation polynomial, the interpolation constraints can be considered in an almost arbitrary order. Thus the same interpolating polynomial can be constructed in several different ways. As a by-product, one can also ask an interesting combinatorial problem from the students about the number of compatible orders of the constraints depending on the cardinality of node system.

Keywords: Hermite interpolation, Neville's recursion, divided differences, linear algebra, interpolation sequence, enumeration, sequence A000680, computer supported learning environment, Mathematica.

MSC: 65D05, 97N50, 97N80

*This is a continuation of the work presented at the Conference CADGME 2012 in Novi Sad and is based on the talk given in ICAI 2014 in Eger.

1. Introduction

Numerical methods are taught to a wide variety of university students. The author itself teaches numerical mathematics at the University of Szeged for several years. Approximation of functions and thus polynomial interpolation is a standard part of numerical mathematics. However, the instructor can choose between different approaches when introducing the simplest polynomial interpolation problems such as the univariate *Lagrange* and *Hermite* interpolation. The classic approach usually considers the Lagrangian and Newtonian form of the interpolation polynomial [3, 5, 7, 9, 10] and mostly because of time constraints, some very powerful ideas and interesting relations remain hidden related to the topic. Some argued for a linear algebra approach [8] and it is worth to mention that the topic can be also seen as a special case of Chinese remaindering from the more abstract point of view [11]. As an extension to the classic approach, in this paper we propose to show the students Neville's simple but powerful idea to solve the Hermite interpolation problem recursively.

Linking a Neville type algorithm with Hermite interpolation is not new in computational mathematics and is often emphasized by computer aided geometric design (CAGD) researchers, see e.g. [2, Chap. 3]. However, to our best knowledge, in teaching Hermite interpolation Neville is not the most frequent approach. Therefore, based on the Neville recursive algorithm in the literature, in this article we propose to explore the construction of Hermite interpolation polynomials by combining small degree Lagrange interpolation and Taylor polynomials as basic building blocks into an interpolating polynomial. The exploration path is aided by interactive graphical tools which were developed in *Wolfram Mathematica* [12].

We also reconsider the recursive algorithm which leads to the Newtonian form of the Hermite interpolation polynomial. We investigate all the bases in a vector space which arise when by the step-by-step construction of the interpolation polynomial the interpolation constraints are reordered [6]. For investigating the possible orders and finite polynomial sequences, another interactive graphical tool is offered for the students. As a by-product, we also come across an interesting elementary combinatorial problem.

The paper is structured as follows. Section 2.1 introduces the problem, Section 2.2 reviews Neville's solution for the Lagrange problem. Section 2.3 contains the generalization of Neville's scheme to the Hermite problem. Section 2.4 considers the Hermite interpolation polynomials in different bases. The last section concludes.

2. The Hermite problem

2.1. Problem specification

First we give some basic definitions and specify the problem exactly, that we want to solve together with the students. We want to keep the technicalities in a minimum

level and therefore we do not consider here problems with higher order derivatives and we assume that in all nodes information about both the zero-order and first-order derivatives are given. We work over the reals, that is, we assume that both the nodes and the given function values are real numbers. This approach is taken e.g. in [3, 9].

Given three finite lists X, F, DF with cardinality $|X| = |F| = |DF| = n \in \mathbb{N}$.

Definition 2.1. Assume that the list X consists of n distinct real numbers, we may also assume that elements of X are ordered:

$$X = \{x_1, x_2, \ldots, x_{n-1}, x_n\},\ x_1 < x_2 < \cdots < x_{n-1} < x_n. \tag{2.1}$$

We call the x_i's in (2.1) the *interpolation nodes* and the grid X the *node system.*

No further assumptions over the values in $F = \{f_1, \ldots, f_n\}$ (function values, zeroth derivatives) and $DF = \{df_1, \ldots, df_n\}$ (first derivatives) are made. We wish to construct an interpolating polynomial H with the following constraints:

$$H(x_j) = f_j \ \wedge\ H'(x_j) = df_j \quad (1 \le j \le n). \tag{2.2}$$

Definition 2.2. (A) polynomial H which satisfies all constraints is called (an) *Hermite interpolation polynomial* and we also denote it by $H^{(1,1,\ldots,n,n)}$ to emphasize that it satisfies two interpolation constraints at all nodes.

In the sequel we explore the existence and uniqueness of the interpolation polynomial in a computer supported environment. The learning unit about Hermite interpolation usually follows the Lagrange interpolation unit, which is the simplest interpolation problem type, since only function values f_j's (zeroth derivatives) are known in the nodes. We assume that the students are already familiar with Lagrange interpolation. For the exploration we use the general purpose CAS Mathematica, but the exploration can be adapted to any other computer algebra system. We wish to challenge the students to generalize the Neville recursive scheme to the solve the Hermite problem. Therefore as a first step, we quickly review the Neville's main idea for solving Lagrange interpolation in the next subsection. All steps are elementary.

2.2. Neville solution for the Lagrange problem

Assume that we wish to solve the Lagrangian problem with three nodes, i.e., the Lagrange interpolation problem with initial data $(x_1, f_1), (x_2, f_2), (x_3, f_3)$ is given. We wish to find a polynomial L that satisfies all the three constraints:

$$L(x_1) = f_1,\ L(x_2) = f_2,\ L(x_3) = f_3.$$

Assume moreover, that $L^{(1,2)}$ is the linear polynomial which fits to (x_1, f_1) and (x_2, f_2) and $L^{(2,3)}$ is the linear polynomial which fits to (x_2, f_2) and (x_3, f_3).

Neville's beautiful and powerful idea is to combine $L^{(1,2)}$ and $L^{(2,3)}$ into a polynomial $L = L^{(1,2,3)}$ which satisfies all the three initial interpolation constraints: As a first step we ask the student to construct the quadratics

$$q_1 = \frac{x - x_3}{x_1 - x_3} L^{(1,2)}, \quad q_2 = \frac{x - x_1}{x_3 - x_1} L^{(2,3)}. \tag{2.3}$$

It is easy to see (as Figure 1 indicates) that

$$q_1(x_1) = f_1,\ q_1(x_3) = 0; \quad q_2(x_1) = 0,\ q_2(x_3) = f_3,$$

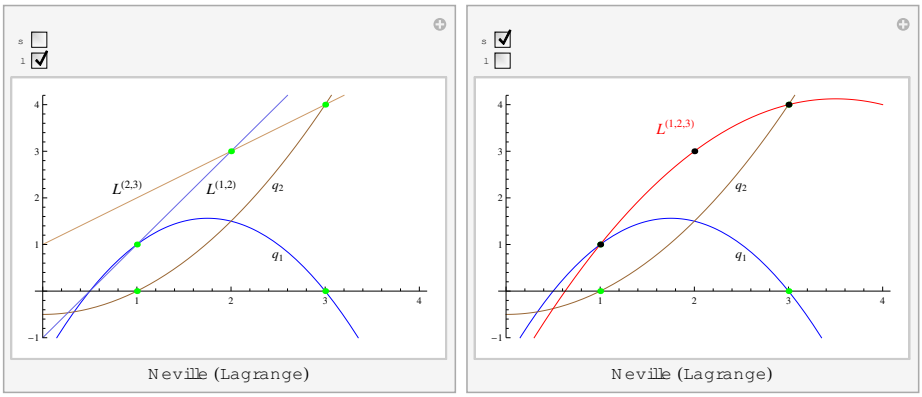

Figure 1: Combining two quadratics into one Lagrange interpolation polynomial

and therefore their sum $q_1 + q_2$ satisfies the first and the last initial interpolation constraints, that is,

$$(q_1 + q_2)(x_1) = f_1 + 0 = f_1, \quad (q_1 + q_2)(x_3) = 0 + f_3 = f_3.$$

But what about the middle point x_2? Is it true that $(q_1 + q_2)(x_2) = f_2$? A short calculation shows that

$$\begin{aligned} L(x_2) &= L^{(1,2,3)}(x_2) = (q_1 + q_2)(x_2) \\ &= \frac{x_2 - x_3}{x_1 - x_3} f_2 + \frac{x_2 - x_1}{x_3 - x_1} f_2 = f_2 \frac{(x_3 - x_2) + (x_2 - x_1)}{x_3 - x_1} = f_2. \end{aligned}$$

In a similar way, we can combine the two polynomials $L^{(1,\ldots,n-1)}$ and $L^{(2,\ldots,n)}$ satisfying all except for the last, and all except for the first interpolating constraints into

$$L = \frac{x - x_n}{x_1 - x_n} L^{(1,\ldots,n-1)} + \frac{x - x_1}{x_n - x_1} L^{(2,\ldots,n)},$$

which satisfies all the n constraints,

$$L(x_j) = f_j \quad (j = 1, \ldots, n).$$

In the next subsection we try to challenge the students to apply the same recursive scheme in order to solve the Hermite interpolation problem given above. We start again with building up an interpolation polynomial satisfying three interpolation constraints by combining two smaller interpolation polynomials. Both smaller polynomials satisfy two interpolation constraints from the initial Hermite constraint-set. However, this time some of the constraint corresponds to the slope of the tangent.

2.3. Neville solution for the Hermite problem

The problem specification of the Hermite interpolation as given in Section 2.1 may make it difficult to come up with the right recursive analog of the idea discussed in the previous Section 2.2. It is clear that uniqueness can be guaranteed by a suitable degree bound. Let us denote the (formally) cubic polynomial which satisfies the constraints of (2.2) on two nodes by $H^{(1,1,2,2)}$, i.e., $H^{(1,1,2,2)}(x_1) = f_1, (H^{(1,1,2,2)})'(x_1) = df_1$, $H^{(1,1,2,2)}(x_2) = f_2, (H^{(1,1,2,2)})'(x_2) = df_2$. It can be shown that we cannot combine the two Hermite (Taylor) polynomials $H^{(1,1)} = T^{(1,1)}$ and $H^{(2,2)} = T^{(2,2)}$ into $H^{(1,1,2,2)}$, but we should rather consider linear Taylor and Lagrange polynomials as atoms for the construction (cf. [7, p. 53]). Therefore we will work with intermediate polynomials which were not present in the initial problem specification.

Definition 2.3. We extend the notation introduced in Definition 2.2 to cover the intermediate polynomials. $H^{(1,1,2)}$ denotes the polynomial which satisfies only the three constraints $H^{(1,1,2)}(x_1) = f_1$, $(H^{(1,1,2)})'(x_1) = df_1$, $H^{(1,1,2)}(x_2) = f_2$ and $H^{(1,2)}$ is simply $L^{(1,2)}$.

As a useful hint, we suggest the students to build up the polynomial $H^{(1,1,2)}$ from $T^{(1,1)}$ and $L^{(1,2)}$. Forming again the two quadratics $Q_1 = \frac{x-x_2}{x_1-x_2}T^{(1,1)}$ and $Q_2 = \frac{x-x_1}{x_2-x_1}L^{(1,2)}$ by just multiplying $T^{(1,1)}$ and $L^{(1,2)}$ with a suitable linear factor as in (2.3), we easily verify that (see also Figure 2).

$$(Q_1 + Q_2)(x_1) = Q_1(x_1) = f_1, (Q_1 + Q_2)(x_2) = Q_2(x_2) = f_2.$$

But what about the derivative at x_1? Let us compute it!

$$(Q_1 + Q_2)'(x_1) = \frac{f_1}{x_1 - x_2} + df_1 + Q_2'(x_1) = \frac{f_1}{x_1 - x_2} + df_1 + \frac{f_1}{x_2 - x_1} = df_1$$

Thus

$$H^{(1,1,2)} = Q_1 + Q_2 = \frac{x - x_2}{x_1 - x_2}H^{(1,1)} + \frac{x - x_1}{x_2 - x_1}H^{(1,2)}.$$

At this point we may already see the generalization of the recursive scheme (Figure 3). In a similar way we construct $H^{(1,2,2)}$ and finally combine $H^{(1,1,2)}$ and $H^{(1,2,2)}$ to get the polynomial $H^{(1,1,2,2)}$.

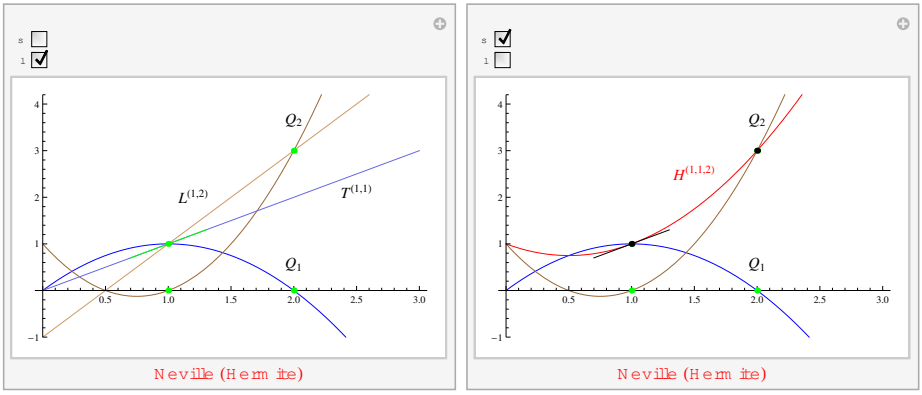

Figure 2: Combining two quadratics into one Hermite interpolation polynomial

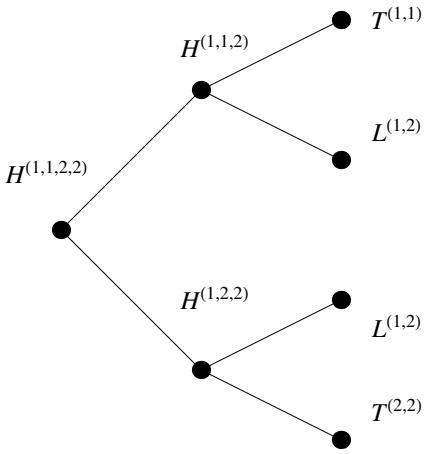

Figure 3: Recursive scheme for the Hermite polynomial $H^{(1,1,2,2)}$

Especially with computer science students, we develop a prototype code in *Mathematica* for the construction of the Hermite polynomial following the Neville's scheme (H0 is the main function, the base cases and the general recursive case are separated using *Mathematica*'s pattern matching capabilities):

```
H0[X_,F_,DF_,var_]:=H[Flatten[Transpose[{X,X}]],Flatten[Transpose[{F,DF}]],var]
H[{x_},{ff_},var_]=ff;
H[{x_,x_},{ff_,df_},var_]=ff+df(var-x);

H[X_,FF_,var_]:=1/(Last(X)-First[X])
((var-First[X]) H[Drop[X,1],Drop[FF,If[X[[1]] === X[[2]], {2}, {1}]],var] -
(var-Last[X]) H[Drop[X,-1],Drop[FF,-1],var])
```

2.4. Order of the interpolation constraints

The classic Newtonian form of the interpolation polynomial using a non-monomial basis

$$B : b_0 = 1, b_1 = (x - x_1)^1, b_2 = (x - x_1)^2, b_3 = (x - x_1)^2(x - x_2)^1, \dots$$

is widely known and taught, see [7, 9, 10]. The construction of the triangular shaped difference table and the Newton polynomial corresponds to a specific order of the $2n$ interpolation constraints. We build up the polynomial $H = H^{(1,1,\dots,n,n)}$ starting from a constant polynomial by successively considering one constraint at a time and forming the finite polynomial sequence

$$h : h_0 = H^{(1)} = f_1, h_1 = H^{(1,1)} = f_1 + df_1(x - x_1), \dots, h_{2n-1} = H.$$

Recall that the first raw of the triangular table containing the generalized divided differences is the coefficient-sequence of the Hermite interpolation polynomial in the basis B. Computation of the coefficients means solving linear equations successively.

In this subsection we reconsider the Hermite interpolation sequence leading to H: We challenge the students to build up H in many different ways, e.g., by extending the Lagrange interpolation polynomial $L^{(1,2,\dots,n)}$ to $H^{(1,1,\dots,n,n)}$. To consider the options we have, we introduce the following definition:

Definition 2.4. We call the order of the interpolation constraints *admissible* or *compatible*, if at each node x_j, we use the constraint on the first derivative df_j only after the constraint on f_j.

To be able to play interactively with the possible interpolation sequences, we developed a Mathematica graphical tool. The user can select successively constraints via a button controller on the right. At the beginning of the experiment, no constraint is selected. Once a constraint is selected and added to the list of used constraints, the user gets a real-time visual feedback: The application shows the graphs of the previous and current interpolation polynomial in the sequence h. Interpolation constraints are also visualized. Moreover, a graph in the middle of the main panel indicates the already used constraints, the last used constraint and the still available constraints. The formula of the last element of the polynomial sequence h is also shown. The following screenshots (Figure 4 and Figure 5) show two possible starting sequences for the same three-node Hermite problem ($X = \{1, 2, 4\}$, $F = \{3, 5, 2\}$, $DF = \{1, 0, -1\}$).

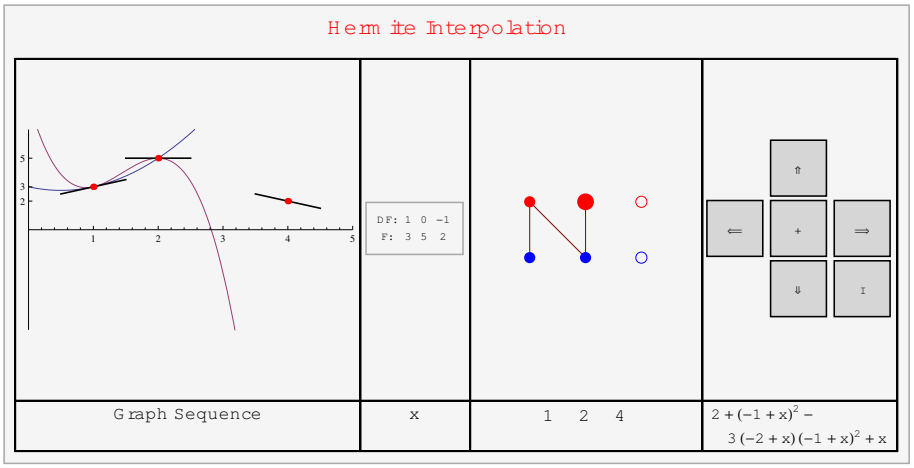

Figure 4: Starting sequence $H^{(1)}, H^{(1,1)}, H^{(1,1,2)}, H^{(1,1,2,2)}$ corresponding to the classic difference scheme ("zigzag")

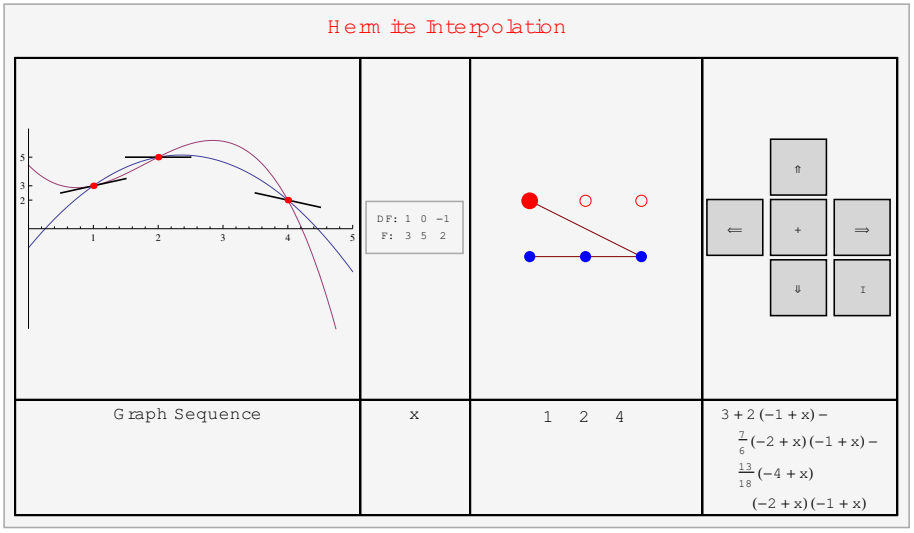

Figure 5: Starting sequence $H^{(1)}, H^{(1,2)}, H^{(1,2,3)}, H^{(1,1,2,3)}$ corresponding to the extension of the Lagrangian polynomial

After the experiment, an interesting question about the number of compatible constraint orderings may pop up among the students. This leads to an elementary combinatorial problem. We identify this scenario as an important by-product of the experiments above: The (oriented) exploration *may lead unexpectedly to new*

nice mathematical questions and conjectures and *can connect seemingly distinct areas.*

We close this subsection with resolving this question by a different type of computer support. Assuming that the node system X consists of $n = 1, 2, 3, 4, \ldots$ distinct elements, we have $1, 6, 90, 2520, \ldots$ many compatible orders. Doesn't that integer sequence look familiar? Typing the first few elements into On-Line Encyclopedia of Integer Sequences (OEIS) [4], we learn that the sequence can be found as A600680 and has a closed form

$$a_n = \frac{(2n)!}{2^n}.$$

It is also very insightful that nth element of the number sequence can be interpreted as the *number of ways that* $2n$ *people of different heights can be arranged (for a photograph) in two rows of equal length so that every person in the front row is shorter than the person immediately behind them in the back row.*

3. Conclusion and future work

We reviewed the possible approaches to the teaching of Hermite interpolation and as an extension to the classic Lagrangian and Newtonian solutions, we proposed to show the students the generalization of Neville's simple and nice recursive idea. We developed several tools which facilitate the exploration of Hermite interpolation in a computer supported environment. One of the tools enabled the interactive exploration of the different forms of the Hermite interpolation polynomial depending on the order in which the interpolation constraints are considered. In the future, in the frame of an Austro-Hungarian project we are also interested in the computer supported verification of the correctness of the generalized Neville algorithm.

Acknowledgements. Author's work is partially supported by the projects IPA HU-SRB/1203/221/024, OTKA K83219 and OMMA 87öu15.

References

[1] Cappello, P.R., Gallopoulos E., Koc C.K., Systolic computation of interpolating polynomials, *Computing*, 45(2) (1990), 95–117.

[2] Goldman, R., *Pyramid Algorithms: A Dynamic Programming Approach to Curves and Surfaces for Geometric Modeling*, The Morgan Kaufmann Series in Computer Graphics, Morgan Kaufmann (2002).

[3] Móricz, F., *Introduction to Numerical Mathematics*, (in Hungarian), Polygon (2008).

[4] OEIS Foundation Inc., The On-Line Encyclopedia of Integer Sequences (2011), `http://oeis.org/A000680`. [access: January, 2015]

[5] Reiter, C.A., Exploring Hermite Interpolation with Mathematica, *Primus*, 2(2) (1992), 173–182.

[6] Stachó, L.L., Vajda, R., Hermite Interpolation Sequences over Fields, *Linear Algebra and its Applications* Vol. 439(1) (2013), 66–77.

[7] Stoer, J., Bulirsch, R., *Introduction to Numerical Analysis*, Springer (2002).

[8] Tassa, T., A Linear Algebraic Approach in Teaching Interpolation, *Mathematics and Computer Education* (2007), 37–35.

[9] Vajda, R., *Numerical Methods: Theorems, Exercises and Computer Experiments*, (in Hungarian), online lecture notes (2011).
`www.model.u-szeged.hu/data/etc/edoc/tan/RVajda` [access: January, 2015]

[10] Wikipedia (the free online encyclopedia), Hermite Interpolation, `http://en.wikipedia.org/wiki/Hermite_interpolation`

[11] Winkler, F., *Polynomial Algorithms in Computer Algebra*, Springer (1996).

[12] Wolfram Research, Inc., *Mathematica*, Version 9.0, Champaign, IL (2012).