2

Revisiting the Ceschino Interpolation Method

Alain Hébert
École Polytechnique de Montréal
Canada

1. Introduction

The Ceschino polynomial expansion method is a generalization of the Taylor polynomial expansion method where higher derivatives of a function are predicted in addition to the value of the function itself. This technique was first introduced by (Ceschino, 1956), but was largly forgotten afterward. An unsuccessfull atempt was tried in 1975 to apply the Ceschino coupling relations to the solution of an elliptic space–dependent differential equation, but the resulting spatial discretization was found to be less accurate than competing finite-element approaches, as presented by (Pageau, 1975). No further published work was reported after the Pageau thesis.

Here, we propose to apply the Ceschino coupling relations to the basic interpolation problem, as an alternative to existing univariate interpolation schemes, such as the cubic spline approach. The interpolation problem consists to evaluate a functional $\mathcal{I}\{f(x);\xi\}$ of a continuous function (or dependent variable) $f(x)$ at a specific point ξ in the case where function $f(x)$ is only known at tabulated abscissa (or independent variables) $\{x_{m+1/2}\,;\ m = 0, M\}$. We also introduce the concept of interpolation factors (a. k. a., *terp factors*) that are useful for interpolating large databases with respect to a small number of independent variables, as presented by (MacFarlane, 1984). The Ceschino polynomial expansion method is the core component of the multiparameter reactor database system used in the reactor physics code DRAGON for performing cross section interpolation (Hébert, 2009). We will show that Ceschino polynomial expansion theory is an attractive choice for computing such interpolation factors and propose sample Matlab scripts for performing this task.

2. Ceschino polynomial expansion theory

The polynomial expansion theory is first applied over the one-dimensional domain depicted in Fig. 1. A continuous function $f(x)$ is defined over this domain and is known at specific abscissa points $x_{m+1/2}$. A $(J+1)$–th order Taylor series expansion of $f(x)$ around $x = x_{m-1/2}$ is written

$$f_{m+1/2} = \sum_{j=0}^{J} (\Delta x_m)^j \, M^{(j)}_{m-1/2} + \mathcal{O}(\Delta x_m)^{J+1} \tag{1}$$

where the mesh width is equal to

$$\Delta x_m = x_{m+1/2} - x_{m-1/2} \tag{2}$$

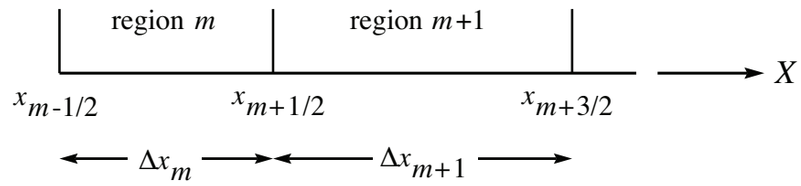

Fig. 1. Definition of the 1D domain.

and where

$$f_{m+1/2} \equiv f(x_{m+1/2}) \equiv M^{(0)}_{m+1/2} \quad \text{and} \quad M^{(j)}_{m-1/2} \equiv \frac{1}{j!}\left.\frac{d^j f}{dx^j}\right|_{x_{m-1/2}} . \tag{3}$$

A Ceschino expansion is nothing but the Taylor's expansion for the derivatives $f^{(k)}(x)$ of function $f(x)$. It is written

$$M^{(k)}_{m+1/2} = \sum_{j=k}^{J} (\Delta x_m)^{j-k} \binom{j}{k} M^{(j)}_{m-1/2} + \mathcal{O}(\Delta x_m)^{J-k+1} \tag{4}$$

where the binomial coefficients are defined as

$$\binom{j}{k} \equiv \frac{j!}{(j-k)!\,k!} \ . \tag{5}$$

Our interpolation strategy is based on two– and three–point coupling relations obtained directly from the Ceschino polynomial expansion (4). Two points relations are used at the extremities of the domain and three–point relations are used inside. Cubic Hermite polynomials will also be introduced to perform the interpolation operation.

2.1 Two–points Ceschino coupling relations

Our relations are coupling the first N derivatives of $f(x)$, with $N = 1$ leading to a cubic interpolation strategy. We set $J = 2N$ in Eq. (4), leading to a truncation error of order $2N+1$ if $k = 0$. We next perform a linear combination of the first N components $M^{(k)}_{m+1/2}$, introducing coefficients θ_k. The linear combination permits to maintain the order of the truncation error to $2N+1$. We write

$$\sum_{k=0}^{N} \theta_k M^{(k)}_{m+1/2} = \sum_{k=0}^{N} \sum_{j=k}^{2N} \theta_k (\Delta x_m)^{j-k} \binom{j}{k} M^{(j)}_{m-1/2} + \mathcal{O}(\Delta x_m)^{2N+1} \ . \tag{6}$$

After permutation of the two summations with the corresponding indices j and k in the right-hand-side, we get

$$\begin{aligned} \sum_{k=0}^{N} \theta_k M^{(k)}_{m+1/2} = & \sum_{k=0}^{N} \sum_{j=0}^{k} \theta_j (\Delta x_m)^{k-j} \binom{k}{j} M^{(k)}_{m-1/2} \\ & + \sum_{k=N+1}^{2N} \sum_{j=0}^{N} \theta_j (\Delta x_m)^{k-j} \binom{k}{j} M^{(k)}_{m-1/2} + \mathcal{O}(\Delta x_m)^{2N+1} . \end{aligned} \tag{7}$$

We choose coefficients θ_j in such a way that

$$\sum_{j=0}^{N} \theta_j \, (\Delta x_m)^{k-j} \binom{k}{j} = 0 \; ; \;\; k = N+1, 2N \tag{8}$$

and we define coefficients $\bar{\theta}_k$ as

$$\bar{\theta}_k = -\sum_{j=0}^{k} \theta_j \, (\Delta x_m)^{k-j} \binom{k}{j} \; ; \;\; k = 0, N \; . \tag{9}$$

We have obtained our $(2N+1)$–th order two–points Ceschino coupling relations as

$$\sum_{k=0}^{N} \left[\bar{\theta}_k \, M^{(k)}_{m-1/2} + \theta_k \, M^{(k)}_{m+1/2} \right] = 0 \; . \tag{10}$$

where the $\mathcal{O}(\Delta x_m)^{2N+1}$ error term is not given.
We need to determine a set of $2(N+1)$ coefficients θ_k and $\bar{\theta}_k$. Equations (8) and (9) permit to determine $2N+1$ of them, leaving θ_0 to be fixed. However, all values of θ_0 leads to valid solutions, making this choice arbitrary. We have chosen $\theta_0 = 1/(\Delta x_m)^2$ in order to simplify the resulting mathematical formalism.
In the specific case of cubic Ceschino interpolation, we set $N = 1$, so that Eqs. (8) and (9) reduce to

$$\begin{aligned} 2\Delta x_m \, \theta_1 &= -(\Delta x_m)^2 \, \theta_0 \\ \bar{\theta}_0 &= -\theta_0 \\ \text{and} \;\; \bar{\theta}_1 &= -\Delta x_m \, \theta_0 - \theta_1 \end{aligned} \tag{11}$$

so that our coefficients are

$$\begin{aligned} \bar{\theta}_0 &= -\frac{1}{(\Delta x_m)^2} \; , \quad \theta_0 = \frac{1}{(\Delta x_m)^2} \\ \bar{\theta}_1 &= -\frac{1}{2\Delta x_m} \;\; \text{and} \;\; \theta_1 = -\frac{1}{2\Delta x_m} \; . \end{aligned} \tag{12}$$

2.2 Three–points Ceschino coupling relations

The three–points Ceschino coupling relations span two consecutive regions along the X axis, as depicted in Fig. 1. We set $J = 3N$ in Eq. (4), leading to a truncation error of order $3N+1$ if $k = 0$. The Ceschino expansion are written

$$\begin{aligned} M^{(k)}_{m-1/2} &= \sum_{j=k}^{3N} (-\Delta x_m)^{j-k} \binom{j}{k} M^{(j)}_{m+1/2} + \mathcal{O}(\Delta x_m)^{3N-k+1} \\ M^{(k)}_{m+3/2} &= \sum_{j=k}^{3N} (\Delta x_{m+1})^{j-k} \binom{j}{k} M^{(j)}_{m+1/2} + \mathcal{O}(\Delta x_{m+1})^{3N-k+1} \end{aligned} \tag{13}$$

where the mesh widths are equal to

$$\Delta x_m = x_{m+1/2} - x_{m-1/2} \quad \text{and} \quad \Delta x_{m+1} = x_{m+3/2} - x_{m+1/2} \; . \tag{14}$$

We next perform a linear combination of the first N components $M^{(k)}_{m-1/2}$ and $M^{(k)}_{m+3/2}$, introducing coefficients $\breve{\beta}_k$ and β_k. The linear combination permits to maintain the order of the truncation error to $3N+1$. We write

$$\sum_{k=0}^{N} \breve{\beta}_k \; M^{(k)}_{m-1/2} + \beta_k \, M^{(k)}_{m+3/2}$$
$$= \sum_{k=0}^{N} \sum_{j=k}^{3N} \left[\breve{\beta}_k \, (-\Delta x_m)^{j-k} + \beta_k \, (\Delta x_{m+1})^{j-k} \right] \binom{j}{k} M^{(j)}_{m+1/2} \tag{15}$$

where the truncation error is a linear combination of $\mathcal{O}(\Delta x_m)^{3N+1}$ and $\mathcal{O}(\Delta x_{m+1})^{3N+1}$.
After permutation of the two summations with the corresponding indices j and k in the right-hand-side, we get

$$\sum_{k=0}^{N} \breve{\beta}_k \; M^{(k)}_{m-1/2} + \beta_k \, M^{(k)}_{m+3/2}$$
$$= \sum_{k=0}^{N} \sum_{j=0}^{k} \left[\breve{\beta}_j \, (-\Delta x_m)^{k-j} + \beta_j \, (\Delta x_{m+1})^{k-j} \right] \binom{k}{j} M^{(k)}_{m+1/2}$$
$$+ \sum_{k=N+1}^{3N} \sum_{j=0}^{N} \left[\breve{\beta}_j \, (-\Delta x_m)^{k-j} + \beta_j \, (\Delta x_{m+1})^{k-j} \right] \binom{k}{j} M^{(k)}_{m+1/2} \; . \tag{16}$$

We choose coefficients $\breve{\beta}_j$ and β_j in such a way that

$$\sum_{j=0}^{N} \left[\breve{\beta}_j \, (-\Delta x_m)^{k-j} + \beta_j \, (\Delta x_{m+1})^{k-j} \right] \binom{k}{j} = 0 \; ; \;\; k = N+1, 3N \tag{17}$$

and we define coefficients $\bar{\beta}_k$ as

$$\bar{\beta}_k = - \sum_{j=0}^{k} \left[\breve{\beta}_j \, (-\Delta x_m)^{k-j} + \beta_j \, (\Delta x_{m+1})^{k-j} \right] \binom{k}{j} \; ; \;\; k = 0, N \; . \tag{18}$$

We have obtained our $(3N+1)$–th order three–points Ceschino coupling relations as

$$\sum_{k=0}^{N} \left[\breve{\beta}_k \, M^{(k)}_{m-1/2} + \bar{\beta}_k \, M^{(k)}_{m+1/2} + \beta_k \, M^{(k)}_{m+3/2} \right] = 0 \; . \tag{19}$$

We need to determine a set of $3(N+1)$ coefficients $\breve{\beta}_k$, $\bar{\beta}_k$ and β_k. Equations (18) and (19) permit to determine $3N+1$ of them, leaving $\breve{\beta}_0$ and β_0 to be fixed. A first set of coefficients can be obtained by setting $\breve{\beta}_0 = -1/(\Delta x_m)^2$ and $\beta_0 = 1/(\Delta x_{m+1})^2$. A second independent set can be obtained by setting $\breve{\beta}'_0 = 1/(\Delta x_m)^3$ and $\beta'_0 = 1/(\Delta x_{m+1})^3$. Any other consistent set is a linear combination of these two.
In the specific case of cubic Ceschino interpolation, we set $N = 1$, so that Eqs. (17) and (18) reduce to

$$\begin{aligned} -2\Delta x_m \, \breve{\beta}_1 + 2\Delta x_{m+1} \, \beta_1 &= -(\Delta x_m)^2 \, \breve{\beta}_0 - (\Delta x_{m+1})^2 \, \beta_0 \\ 3(\Delta x_m)^2 \, \breve{\beta}_1 + 3(\Delta x_{m+1})^2 \, \beta_1 &= (\Delta x_m)^3 \, \breve{\beta}_0 - (\Delta x_{m+1})^3 \, \beta_0 \\ \bar{\beta}_0 &= -(\breve{\beta}_0 + \beta_0) \\ \text{and} \quad \bar{\beta}_1 &= \Delta x_m \, \breve{\beta}_0 - \Delta x_{m+1} \, \beta_0 - (\breve{\beta}_1 + \beta_1) \end{aligned} \tag{20}$$

so that our two independent sets of coefficients are

$$\begin{aligned} \breve{\beta}_0 &= -\frac{1}{(\Delta x_m)^2}\ ,\ \bar{\beta}_0 = \frac{1}{(\Delta x_m)^2} - \frac{1}{(\Delta x_{m+1})^2}\ ,\ \beta_0 = \frac{1}{(\Delta x_{m+1})^2}\ , \\ \breve{\beta}_1 &= -\frac{1}{3\Delta x_m}\ ,\ \bar{\beta}_1 = -\frac{2}{3}\left[\frac{1}{\Delta x_m} + \frac{1}{\Delta x_{m+1}}\right]\ ,\ \beta_1 = -\frac{1}{3\Delta x_{m+1}} \end{aligned} \tag{21}$$

and

$$\begin{aligned} \breve{\beta}_0 &= \frac{1}{(\Delta x_m)^3}\ ,\ \bar{\beta}_0 = -\frac{1}{(\Delta x_m)^3} - \frac{1}{(\Delta x_{m+1})^3}\ ,\ \beta_0 = \frac{1}{(\Delta x_{m+1})^3}\ , \\ \breve{\beta}_1 &= \frac{1}{2(\Delta x_m)^2}\ ,\ \bar{\beta}_1 = \frac{1}{2}\left[\frac{1}{(\Delta x_m)^2} - \frac{1}{(\Delta x_{m+1})^2}\right]\ ,\ \beta_1 = -\frac{1}{2(\Delta x_{m+1})^2}\ . \end{aligned} \tag{22}$$

2.3 Interpolation with cubic Hermite polynomials

Knowledge of $M^{(0)}_{m+1/2}$ and the capability to easily obtain $M^{(1)}_{m+1/2}$ on each tabulated point $x_{m+1/2}$ makes possible the interpolation of function $f(x)$ at each values of the independent variable x with a cubic Hermite polynomial in x. Such polynomial guarantee that the interpolated value and first derivative of the dependent variable remains continuous in x over the complete domain. As pointed out by (Rozon et al., 1981), this continuity property of the first derivative is often required in numerical applications such as those based on perturbation theory.

The first operation consists to solve a tridiagonal linear matrix system for obtaining the unknown vector $\boldsymbol{M}^{(1)} = \text{col}\{M^{(1)}_{m+1/2};\ m = 0, M\}$ over a M–region domain, considering the known values $M^{(0)}_{m+1/2}$ of $f(x)$ at tabulation points $x_{m+1/2}$. The linear matrix system is made with the first independent set of coefficients from Eq. (21) for linking the unknowns inside the domain. We have selected the first set in order to obtain a symmetric $\mathbb{C}$ matrix with minimum powers of Δx_m as coefficients. The first and last line coefficients are obtained from Eq. (12). Using coefficients from Eq. (12) with those from Eq. (22) leads to a singular $\mathbb{C}$ matrix. This last observation gives an additional clue for selecting three-point coefficients from Eq. (21).

The linear system is written

$$\mathbb{C}\ \boldsymbol{M}^{(1)} = \boldsymbol{S}^{(0)} \tag{23}$$

where the symmetric tridiagonal matrix is written

$$\mathbb{C} = \begin{bmatrix} \frac{1}{\Delta x_1} & \frac{1}{\Delta x_1} & 0 & \dots & 0 \\ \frac{1}{\Delta x_1} & 2\left(\frac{1}{\Delta x_1} + \frac{1}{\Delta x_2}\right) & \frac{1}{\Delta x_2} & \dots & 0 \\ 0 & \frac{1}{\Delta x_2} & 2\left(\frac{1}{\Delta x_2} + \frac{1}{\Delta x_3}\right) & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \frac{1}{\Delta x_M} \end{bmatrix} \tag{24}$$

and where the source term is written

$$\boldsymbol{S}^{(0)} = \begin{bmatrix} \frac{2}{(\Delta x_1)^2}\left(M^{(0)}_{3/2} - M^{(0)}_{1/2}\right) \\ \frac{3}{(\Delta x_1)^2}\left(M^{(0)}_{3/2} - M^{(0)}_{1/2}\right) + \frac{3}{(\Delta x_2)^2}\left(M^{(0)}_{5/2} - M^{(0)}_{3/2}\right) \\ \frac{3}{(\Delta x_2)^2}\left(M^{(0)}_{5/2} - M^{(0)}_{3/2}\right) + \frac{3}{(\Delta x_3)^2}\left(M^{(0)}_{7/2} - M^{(0)}_{5/2}\right) \\ \vdots \\ \frac{2}{(\Delta x_M)^2}\left(M^{(0)}_{M+1/2} - M^{(0)}_{M-1/2}\right) \end{bmatrix} . \tag{25}$$

The solution of the linear system in Eq. (23) can be performed without pivoting, as matrix $\mathbb{C}$ is diagonally dominant.

We next introduce the cubic Hermite polynomials defined over a reference region $-1/2 \leq u \leq 1/2$. They are

$$\begin{aligned} H_1(u) &= 3\left(\frac{1}{2} - u\right)^2 - 2\left(\frac{1}{2} - u\right)^3 \\ H_2(u) &= \left(\frac{1}{2} - u\right)^2 - \left(\frac{1}{2} - u\right)^3 \\ H_3(u) &= 3\left(\frac{1}{2} + u\right)^2 - 2\left(\frac{1}{2} + u\right)^3 \\ H_4(u) &= \left(\frac{1}{2} + u\right)^2 + \left(\frac{1}{2} + u\right)^3 \end{aligned} \tag{26}$$

so that a function $f(u)$ defined over this domain can be expressed as

$$f(u) \simeq f(-1/2)\, H_1(u) + f'(-1/2)\, H_2(u) + f(1/2)\, H_3(u) + f'(1/2)\, H_4(u) \tag{27}$$

where $-1/2 \leq u \leq 1/2$.

The above relation can be generalized to the interpolation of function $f(x)$ at ξ over region m. We first perform the change of variable

$$u = \frac{1}{\Delta x_m}\left[\xi - \frac{1}{2}\,(x_{m-1/2} + x_{m+1/2})\right] \tag{28}$$

so that

$$\begin{aligned} \mathcal{I}\{f(x);\xi\} = {} & M^{(0)}_{m-1/2}\, H_1(u) + \Delta x_m\, M^{(1)}_{m-1/2}\, H_2(u) + M^{(0)}_{m+1/2}\, H_3(u) \\ & + \Delta x_m\, M^{(1)}_{m+1/2}\, H_4(u) \end{aligned} \tag{29}$$

where $x_{m-1/2} \leq \xi \leq x_{m+1/2}$.

2.4 Introduction of interpolation factors

Interpolation factors are useful to interpolate a large number of dependent variables at a unique value ξ of the independent variable. The interpolation factors are function only of the tabulated abscissas $\{x_{m+1/2}\,;\ m = 0, M\}$ and on the interpolation abscissa x. Using interpolation factors $\{t_{m+1/2}(\xi)\,;\ m = 0, M\}$, an interpolated dependent variable $\mathcal{I}\{f(x);\xi\}$ of $f(\xi)$ is obtained from

$$\mathcal{I}\{f(x);\xi\} = \sum_{m=0}^{M} t_{m+1/2}(\xi)\, f(x_{m+1/2}) \tag{30}$$

where

$$\sum_{m=0}^{M} t_{m+1/2}(\xi) = 1 \ . \tag{31}$$

Interpolation factors can be obtained if the interpolation operation is *distributive*, that is, if it can be distributed to the sum of two functions $f(x)$ and $g(h)$ according to

$$\begin{aligned}\mathcal{I}\{f(x)+g(x);\xi\} &= \sum_{m=0}^{M} t_{m+1/2}(\xi)\,[f(x_{m+1/2})+g(x_{m+1/2})] \\ &= \mathcal{I}\{f(x);\xi\} + \mathcal{I}\{g(x);\xi\} \ .\end{aligned} \tag{32}$$

The simplest form of interpolation factors are those corresponding to linear Lagrange interpolation. In this case, the interpolated value of $f(x)$, with $x_{m-1/2} \le \xi \le x_{m+1/2}$, is given by Eq. (30) with

$$t_\alpha(\xi) = \begin{cases} \frac{1}{2} - u, & \text{if } \alpha = m - 1/2\,; \\ \frac{1}{2} + u, & \text{if } \alpha = m + 1/2\,; \\ 0, & \text{otherwise.} \end{cases} \tag{33}$$

Similar interpolation factors exist for cubic Ceschino interpolation and can be obtained with the following procedure. The source term defined in Eq. (25) can be written in matrix form as

$$\boldsymbol{S}^{(0)} = \mathbb{S}\,\boldsymbol{M}^{(0)} \tag{34}$$

where

$$\mathbb{S} = \begin{bmatrix} -\frac{2}{(\Delta x_1)^2} & \frac{2}{(\Delta x_1)^2} & 0 & \cdots & 0 \\ -\frac{3}{(\Delta x_1)^2} & \frac{3}{(\Delta x_1)^2} - \frac{3}{(\Delta x_2)^2} & \frac{3}{(\Delta x_2)^2} & \cdots & 0 \\ 0 & -\frac{3}{(\Delta x_2)^2} & \frac{3}{(\Delta x_2)^2} - \frac{3}{(\Delta x_3)^2} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{2}{(\Delta x_M)^2} \end{bmatrix} . \tag{35}$$

The interpolated value of $f(\xi)$, with $x_{m-1/2} \le \xi \le x_{m+1/2}$, is therefore given by the relation

$$\mathcal{I}\{f(x);\xi\} = \left[\boldsymbol{H}_1(\xi)^\top + \boldsymbol{H}_2(\xi)^\top\, \mathbb{C}^{-1}\, \mathbb{S}\right] \boldsymbol{M}^{(0)} \tag{36}$$

where $\boldsymbol{H}_1(\xi) = \{H_{1,m+1/2}(\xi)\,;\ m = 0, M\}$ with

$$H_{1,\alpha}(\xi) = \begin{cases} 3\left(\frac{1}{2}-u\right)^2 - 2\left(\frac{1}{2}-u\right)^3, & \text{if } \alpha = m-1/2\ ; \\ 3\left(\frac{1}{2}+u\right)^2 - 2\left(\frac{1}{2}+u\right)^3, & \text{if } \alpha = m+1/2\ ; \\ 0, & \text{otherwise} \end{cases} \tag{37}$$

and $\boldsymbol{H}_2(\xi) = \{H_{2,m+1/2}(\xi)\,;\ m = 0, M\}$ with

$$H_{2,\alpha}(\xi) = \begin{cases} \left(\frac{1}{2}-u\right)^2 - \left(\frac{1}{2}-u\right)^3, & \text{if } \alpha = m-1/2\ ; \\ \left(\frac{1}{2}+u\right)^2 + \left(\frac{1}{2}+u\right)^3, & \text{if } \alpha = m+1/2\ ; \\ 0, & \text{otherwise.} \end{cases} \tag{38}$$

The vector $\boldsymbol{T}(\xi) = \{t_{m+1/2}(\xi)\,;\ m = 0, M\}$ of interpolation factors is obtained after transposition of Eq. (36), leading to

$$\mathcal{I}\{f(x);\xi\} = \boldsymbol{M}^{(0)^\top}\left[\boldsymbol{H}_1(\xi) + \mathbb{S}^\top\,\mathbb{C}^{-1}\,\boldsymbol{H}_2(\xi)\right] \tag{39}$$

so that

$$\boldsymbol{T}(\xi) = \boldsymbol{H}_1(\xi) + \mathbb{S}^\top\,\mathbb{C}^{-1}\,\boldsymbol{H}_2(\xi)\ . \tag{40}$$

3. Matlab scripts and numerical examples

Two Matlab scripts are proposed in Appendices A and B as prototypes of the cubic Ceschino interpolation method. The first script, `alterp()` is used to obtain the terp factors corresponding to an interpolation (if `lderiv=false`) or to a derivation (if `lderiv=true`). The second script, `alteri()` is used to obtain the terp factors corresponding to the definite integration of $f(x)$. The following Matlab session is an example of interpolation similar to the `spline` Matlab tutorial.

```
x=0:10;  y=sin(x);
xx=0:.25:10;
yy=zeros(1,size(xx,2));
for i=1:size(xx,2)
   yy(i)=y*alterp(x,xx(i),false);
end
plot(x,y,'o',xx,yy)
```

Execution of the above script leads to Fig. 2. Similarly, the first derivative of $f(x) = sin(x)$ can be computed by setting `lderiv` $=$ `true`, as described in the following Matlab session.

```
yy=zeros(1,size(xx,2));
for i=1:size(xx,2)
   yy(i)=y*alterp(x,xx(i),true);
end
plot(x,cos(x),'o',xx,yy)
```

Execution of the above script leads to Fig. 3. We observe that the order of the numerical derivation approximation is less than the order of the interpolation, as expected. The higher derivation errors are observed at extremities of the domain, where two-point Ceschino coupling relation are used.

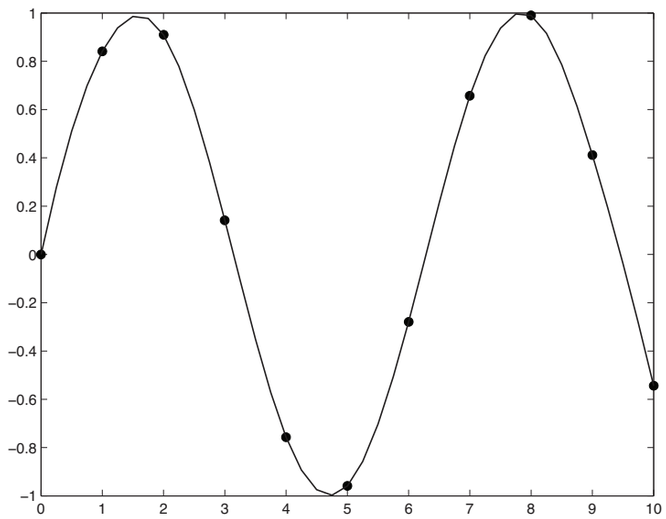

Fig. 2. Interpolation example.

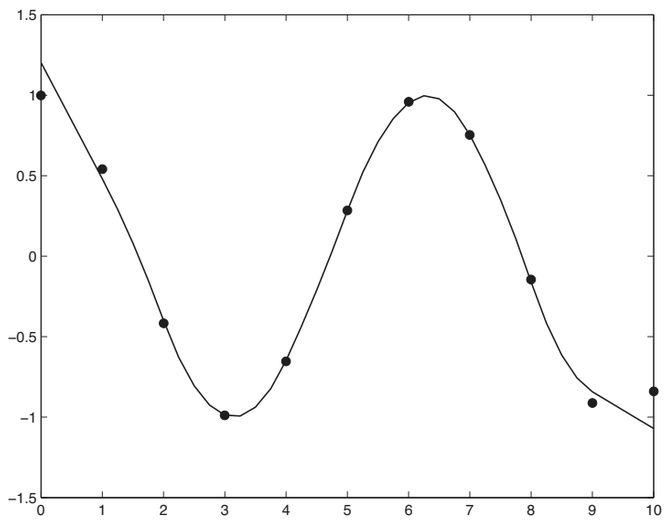

Fig. 3. Derivation example.

4. Conclusion

We have presented a straightforward numerical technique based on Ceschino polynomial expansion. Three applications of this approach permit to perform interpolation, derivation and definite integration of tabulated data. Equation (36) is efficient to interpolate few dependent variables over a large number of points ξ. Equation (39) introduces the concept of *interpolation factors* and is efficient to interpolate a large number of dependent variables over a few number of points ξ. Matlab scripts are provided as basic implementation of the Ceschino interpolating method.

The Ceschino interpolation technique is an alternative to the cubic spline approach based on different mathematical bases. In fact, the interpolating function obtained by this method is a piecewise polynomial function of degree 3 which is only a $\mathcal{C}^1$ function compared to the cubic spline which is a $\mathcal{C}^2$ function. It would be important to obtain error estimates to compare both approaches. However, the Ceschino interpolation technique is currently implemented in legacy applications and its behavior is already found acceptable.

Appendix A

The first Matlab script is used to compute interpolation/derivation factors (a. k. a., terp factors) using Eq. (40). The user must provide the tabulated abscissa defined as $\{x_{m+1/2}\,;\, m = 0, M\}$ and one interpolation point ξ. A logical variable, `lderiv`, select interpolation or derivation mode. The script returns a column vector containing the corresponding terp factors $\{t_{m+1/2}(\xi)\,;\, m = 0, M\}$.

```
function terp=alterp(x,val,lderiv)
% determination of the terp interpolation/derivation factors using
% the order 4 Ceschino method with cubic Hermite polynomials.
% function terp=alterp(x,val,lderiv)
% input parameters:
% x       abscissas (row vector)
% val     abscissa of the interpolated point.
% lderiv  set to true to compute the first derivative with respect to x.
%         set to false to interpolate.
% output parameters:
% terp    interpolation factors (column vector)
% (c) 2007 Alain Hebert, Ecole Polytechnique de Montreal
   n=size(x,2) ;
   if n <= 1
      error('invalid number of points')
   end
   terp=zeros(n,1) ;
   if n == 2
      if lderiv
         terp(1)=-1.0/(x(2)-x(1)) ;
         terp(2)=1.0/(x(2)-x(1)) ;
      else
         terp(1)=(x(2)-val)/(x(2)-x(1)) ;
         terp(2)=1.0-terp(1) ;
      end
   else
      wk=zeros(3,n) ;
%----
%  interval identification.
%----
      temp1=find(val>=x(1:end-1)) ;
      temp2=find(val<=x(2:end)) ;
      if (size(temp1,2) == 0) | (size(temp2,2) == 0)
         error('unable to interpolate')
      end
      i0=temp1(end) ;
      dx=x(i0+1)-x(i0) ;
      u=(val-0.5*(x(i0)+x(i0+1)))/dx ;
```

```
      if lderiv
         h1=(-6.0*(0.5-u)+6.0*(0.5-u)^2)/dx ;
         h2=(-2.0*(0.5-u)+3.0*(0.5-u)^2)/dx ;
         h3=(6.0*(0.5+u)-6.0*(0.5+u)^2)/dx ;
         h4=(-2.0*(0.5+u)+3.0*(0.5+u)^2)/dx ;
         test=0.0 ;
      else
         h1=3.0*(0.5-u)^2-2.0*(0.5-u)^3 ;
         h2=(0.5-u)^2-(0.5-u)^3 ;
         h3=3.0*(0.5+u)^2-2.0*(0.5+u)^3 ;
         h4=-(0.5+u)^2+(0.5+u)^3 ;
         test=1.0 ;
      end
      terp(i0)=h1 ;
      terp(i0+1)=h3 ;
      wk(3,i0)=h2*dx ;
      wk(3,i0+1)=h4*dx ;
%----
%  compute the coefficient matrix.
%----
      hp=1.0/(x(2)-x(1)) ;
      wk(1,1)=hp ;
      wk(2,1)=hp ;
      for i=2:n-1
         hm=hp ;
         hp=1.0/(x(i+1)-x(i)) ;
         wk(1,i)=2.0*(hm+hp) ;
         wk(2,i)=hp ;
      end
      wk(1,n)=hp ;
      wk(2,n)=hp ;
%----
%  forward elimination.
%----
      pmx=wk(1,1) ;
      wk(3,1)=wk(3,1)/pmx ;
      for i=2:n
         gar=wk(2,i-1) ;
         wk(2,i-1)=wk(2,i-1)/pmx ;
         pmx=wk(1,i)-gar*wk(2,i-1) ;
         wk(3,i)=(wk(3,i)-gar*wk(3,i-1))/pmx ;
      end
%----
%  back substitution.
%----
      for i=n-1:-1:1
         wk(3,i)=wk(3,i)-wk(2,i)*wk(3,i+1) ;
      end
%----
%  compute the interpolation factors.
%----
      gar=zeros(1,n+2) ;
      gar(2:n+1)=wk(3,:) ;
      wk=zeros(3,n) ;
      hp2=1.0/(x(2)-x(1)) ;
```

```
      wk(2,1)=-2.0*hp2*hp2 ;
      wk(1,2)=2.0*hp2*hp2 ;
      for i=2:n-1
         hp1=hp2 ;
         hp2=1.0/(x(i+1)-x(i)) ;
         wk(3,i-1)=-3.0*hp1*hp1 ;
         wk(2,i)=3.0*hp1*hp1-3.0*hp2*hp2 ;
         wk(1,i+1)=3.0*hp2*hp2 ;
      end
      wk(3,n-1)=-2.0*hp2*hp2 ;
      wk(2,n)=2.0*hp2*hp2 ;
      for j=1:n
         terp(j)=terp(j)+gar(j:j+2)*wk(:,j) ;
         test=test-terp(j) ;
      end
      if abs(test) > 1.0e-5
         error('wrong terp factors')
      end
      terp(find(abs(terp) <= 1.0e-7))=0.0 ;
   end
```

Appendix B

The second Matlab script is used to compute integration factors permitting to evaluate a definite integral. The user must provide the tabulated abscissa $\{x_{m+1/2};\ m = 0, M\}$ and the integration limits. The script returns a column vector containing the corresponding terp factors.

```
function terp=alteri(x,val0,val1)
% determination of the terp integration factors using the order 4
% Ceschino method with cubic Hermite polynomials.
% function terp=alteri(x,val0,val1)
% input parameters:
% x       abscissas (row vector)
% val0    left integration limit.
% val1    right integration limit.
% output parameters:
% terp    integration factors (column vector)
% (c) 2007 Alain Hebert, Ecole Polytechnique de Montreal
   n=size(x,2) ;
   if n <= 1
      error('invalid number of points')
   elseif val1 <= val0
      error('invalid limits')
   elseif (val0 < x(1)) | (val1 > x(n))
      error('unable to integrate')
   end
   terp=zeros(n,1) ;
   if n == 2
      terp(1)=(x(2)-0.5*(val0+val1))*(val1-val0)/(x(2)-x(1)) ;
      terp(2)=(0.5*(val0+val1)-x(1))*(val1-val0)/(x(2)-x(1)) ;
   else
      wk=zeros(3,n) ;
%----
```

```
%  interval identification.
%----
      for i0=1:n-1
         if (val0 < x(i0+1)) & (val1 > x(i0))
            a=max(val0,x(i0)) ;
            b=min(val1,x(i0+1)) ;
            cc=0.5*(b-a) ;
            dx=x(i0+1)-x(i0) ;
            u1=(a-0.5*(x(i0)+x(i0+1)))/dx ;
            u2=(b-0.5*(x(i0)+x(i0+1)))/dx ;
            uu(1)=0.5*(-(u2-u1)/sqrt(3.0)+u1+u2) ;
            uu(2)=0.5*((u2-u1)/sqrt(3.0)+u1+u2) ;
            for js=1:2
               h1=(3.0*(0.5-uu(js))^2-2.0*(0.5-uu(js))^3)*cc ;
               h2=((0.5-uu(js))^2-(0.5-uu(js))^3)*cc ;
               h3=(3.0*(0.5+uu(js))^2-2.0*(0.5+uu(js))^3)*cc ;
               h4=(-(0.5+uu(js))^2+(0.5+uu(js))^3)*cc ;
               terp(i0)=terp(i0)+h1 ;
               terp(i0+1)=terp(i0+1)+h3 ;
               wk(3,i0)=wk(3,i0)+h2*dx ;
               wk(3,i0+1)=wk(3,i0+1)+h4*dx ;
            end
         end
      end
%----
%  compute the coefficient matrix.
%----
      hp=1.0/(x(2)-x(1)) ;
      wk(1,1)=hp ;
      wk(2,1)=hp ;
      for i=2:n-1
         hm=hp ;
         hp=1.0/(x(i+1)-x(i)) ;
         wk(1,i)=2.0*(hm+hp) ;
         wk(2,i)=hp ;
      end
      wk(1,n)=hp ;
      wk(2,n)=hp ;
%----
%  forward elimination.
%----
      pmx=wk(1,1) ;
      wk(3,1)=wk(3,1)/pmx ;
      for i=2:n
         gar=wk(2,i-1) ;
         wk(2,i-1)=wk(2,i-1)/pmx ;
         pmx=wk(1,i)-gar*wk(2,i-1) ;
         wk(3,i)=(wk(3,i)-gar*wk(3,i-1))/pmx ;
      end
%----
%  back substitution.
%----
      for i=n-1:-1:1
         wk(3,i)=wk(3,i)-wk(2,i)*wk(3,i+1) ;
      end
```

```
%----
%  compute the integration factors.
%----
      test=1.0 ;
      gar=zeros(1,n+2) ;
      gar(2:n+1)=wk(3,:) ;
      wk=zeros(3,n) ;
      hp2=1.0/(x(2)-x(1)) ;
      wk(2,1)=-2.0*hp2*hp2 ;
      wk(1,2)=2.0*hp2*hp2 ;
      for i=2:n-1
         hp1=hp2 ;
         hp2=1.0/(x(i+1)-x(i)) ;
         wk(3,i-1)=-3.0*hp1*hp1 ;
         wk(2,i)=3.0*hp1*hp1-3.0*hp2*hp2 ;
         wk(1,i+1)=3.0*hp2*hp2 ;
      end
      wk(3,n-1)=-2.0*hp2*hp2 ;
      wk(2,n)=2.0*hp2*hp2 ;
      for j=1:n
         terp(j)=terp(j)+gar(j:j+2)*wk(:,j) ;
         test=test-terp(j)/(val1-val0) ;
      end
      if abs(test) > 1.0e-5
         error('wrong terp factors')
      end
      terp(find(abs(terp) <= 1.0e-7))=0.0 ;
   end
```

5. References

Ceschino, F. (1956). *L'intégration approchée des équations différentielles*, Compte Rendu de l'Académie des Sciences, Paris, 243, pp. 1478 – 1479.

Hébert, A. (2009). *Applied Reactor Physics*, Presses Internationales Polytechnique, ISBN 978-2-553-01436-9, 424 p., Montréal.

MacFarlane, R. E. (1984). *TRANSX-CTR: A code for Interfacing MATXS Cross-Section Libraries to Nuclear Transport Codes for Fusion Systems Analysis*, LA-9863-MS, Los Alamos Scientific Laboratory, New Mexico.

Pageau, R. (1975). *Application des méthodes TCNR et des séries de Fourier aux Problèmes statique bi-dimensionnels en physique des réacteurs*, Master Thesis, École Polytechnique de Montréal.

Rozon, D.; Hébert, A.; McNabb, D. (1981). *The Application of Generalized Perturbation Theory and Mathematical Programming to Equilibrium Refueling Studies of a CANDU Reactor*, Nucl. Sci. Eng., 78, pp. 211 – 226.

MATLAB - A Ubiquitous Tool for the Practical Engineer

Edited by Prof. Clara Ionescu

ISBN 978-953-307-907-3
Hard cover, 564 pages
Publisher InTech
Published online 13, October, 2011
Published in print edition October, 2011

A well-known statement says that the PID controller is the â€œbread and butterâ€ of the control engineer. This is indeed true, from a scientific standpoint. However, nowadays, in the era of computer science, when the paper and pencil have been replaced by the keyboard and the display of computers, one may equally say that MATLAB is the â€œbreadâ€ in the above statement. MATLAB has became a de facto tool for the modern system engineer. This book is written for both engineering students, as well as for practicing engineers. The wide range of applications in which MATLAB is the working framework, shows that it is a powerful, comprehensive and easy-to-use environment for performing technical computations. The book includes various excellent applications in which MATLAB is employed: from pure algebraic computations to data acquisition in real-life experiments, from control strategies to image processing algorithms, from graphical user interface design for educational purposes to Simulink embedded systems.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Alain Hébert (2011). Revisiting the Ceschino Interpolation Method, MATLAB - A Ubiquitous Tool for the Practical Engineer, Prof. Clara Ionescu (Ed.), ISBN: 978-953-307-907-3, InTech, Available from: http://www.intechopen.com/books/matlab-a-ubiquitous-tool-for-the-practical-engineer/revisiting-the-ceschino-interpolation-method

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen