CAMBRIDGE

Cambridge University Press
978-0-521-71123-4 - Fundamentals of Engineering Numerical Analysis, Second Edition
Parviz Moin
Excerpt
More information

1 Interpolation

Often we want to fit a smooth curve through a set of data points. Applications might be differentiation or integration or simply estimating the value of the function between two adjacent data points. With interpolation we actually pass a curve *through* the data. If data are from a crude experiment characterized by some uncertainty, it is best to use the method of least squares, which does not require the approximating function to pass through all the data points.

1.1 Lagrange Polynomial Interpolation

Suppose we have a set of $n + 1$ (not necessarily equally spaced) data (x_i, y_i). We can construct a polynomial of degree n that passes through the data:

$$P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n.$$

The $n + 1$ coefficients of P are determined by forcing P to pass through the data. This leads to $n + 1$ equations in the $n + 1$ unknowns, $a_0, a_1, \ldots, a_n$:

$$y_i = P(x_i) = a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_n x_i^n \quad i = 0,\ 1,\ 2, \ldots, n.$$

This procedure for finding the coefficients of the polynomial is not very attractive. It involves solving a system of algebraic equations that is generally ill-conditioned (see Appendix) for large n. In practice we will define the polynomial in an explicit way (as opposed to solving a system of equations). Consider the following polynomial of degree n associated with each point x_j:

$$L_j(x) = \alpha_j (x - x_0)(x - x_1) \cdots (x - x_{j-1})(x - x_{j+1}) \cdots (x - x_n),$$

where α_j is a constant to be determined. In the product notation, L_j is written as follows

$$L_j(x) = \alpha_j \prod_{\substack{i=0 \\ i \neq j}}^{n} (x - x_i).$$

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-71123-4 - Fundamentals of Engineering Numerical Analysis, Second Edition
Parviz Moin
Excerpt
More information

If x is equal to any of the data points except x_j, then $L_j(x_i) = 0$ for $i \neq j$. For $x = x_j$,

$$L_j(x_j) = \alpha_j \prod_{\substack{i=0 \\ i \neq j}}^{n} (x_j - x_i).$$

We now define α_j to be

$$\alpha_j = \left[\prod_{\substack{i=0 \\ i \neq j}}^{n} (x_j - x_i) \right]^{-1}.$$

Then, L_j will have the following important property:

$$L_j(x_i) = \begin{cases} 0 & x_i \neq x_j \\ 1 & x_i = x_j. \end{cases} \tag{1.1}$$

Next we form a linear combination of these polynomials with the data as weights:

$$P(x) = \sum_{j=0}^{n} y_j L_j(x). \tag{1.2}$$

This is a polynomial of degree n because it is a linear combination of polynomials of degree n. It is called a *Lagrange polynomial*. It is the desired interpolating polynomial because by construction, it passes through all the data points. For example, at $x = x_i$

$$P(x_i) = y_0 L_0(x_i) + y_1 L_1(x_i) + \cdots + y_i L_i(x_i) + \cdots + y_n L_n(x_i).$$

Since $L_i(x_k)$ is equal to zero except for $k = i$, and $L_i(x_i) = 1$,

$$P(x_i) = y_i.$$

Note that polynomial interpolation is unique. That is, there is only one polynomial of degree n that passes through a set of $n + 1$ points*. The Lagrange polynomial is just a compact, numerically better behaved way of expressing the polynomial whose coefficients could have also been obtained from solving a system of algebraic equations.

For a large set of data points (say greater than 10), polynomial interpolation for uniformly spaced data can be very dangerous. Although the polynomial is fixed (tied down) at the data points, it can wander wildly between them, which can lead to large errors for derivatives or interpolated values.

* The uniqueness argument goes like this: suppose there are two polynomials of degree n, Z_1 and Z_2 that pass through the same data points, $x_0, x_1, \ldots, x_n$. Let $Z = Z_1 - Z_2$. Z is a polynomial of degree n with $n + 1$ zeros, $x_0, x_1, \ldots, x_n$, which is impossible unless Z is identically zero.

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-71123-4 - Fundamentals of Engineering Numerical Analysis, Second Edition
Parviz Moin
Excerpt
More information

EXAMPLE 1.1 Lagrange Interpolation

Consider the following data, which are obtained from a smooth function also known as Runge's function, $y = (1 + 25x^2)^{-1}$:

x_i	−1.00	−0.80	−0.60	−0.40	−0.20	0.00	0.20	0.40	0.60	0.80	1.00
y_i	0.038	0.058	0.100	0.200	0.500	1.00	0.500	0.200	0.100	0.058	0.038

We wish to fit a smooth curve through the data using the Lagrange polynomial interpolation, for which the value at any point x is simply

$$P(x) = \sum_{j=0}^{n} y_j \prod_{\substack{i=0 \\ i \neq j}}^{n} \frac{x - x_i}{x_j - x_i}.$$

For example at the point $(x = 0.7)$, the interpolated value is

$$\begin{aligned} P(.7) = 0.038 & \frac{(0.7+0.8)(0.7+0.6)\cdots(0.7-0.8)(0.7-1.0)}{(-1.0+0.8)(-1.0+0.6)\cdots(-1.0-0.8)(-1.0-1.0)} \\ & + 0.058 \frac{(0.7+1.0)(0.7+0.6)\cdots(0.7-0.8)(0.7-1.0)}{(-0.8+1.0)(-0.8+0.6)\cdots(-0.8-0.8)(-0.8-1.0)} \\ & + \cdots \\ & + 0.038 \frac{(0.7+1.0)(0.7+0.8)\cdots(0.7-0.6)(0.7-0.8)}{(1.0+1.0)(1.0+0.6)\cdots(1.0-0.6)(1.0-0.8)} = -0.226. \end{aligned}$$

Evaluating the interpolating polynomial at a large number of intermediate points, we may plot the resulting polynomial curve passing through the data points (see Figure 1.1). It is clear that the Lagrange polynomial behaves very poorly between some of the data points, especially near the ends of the interval. The problem does not go away by simply having more data points in the interval and thereby tying down the function further. For example, if instead of eleven points we had twenty-one uniformly spaced data points in the same interval, the overshoots at the ends would have peaked at nearly 60 rather than at 1.9 as they did for eleven points. However, as shown in the following example, the problem can be somewhat alleviated if the data points are non-uniformly spaced with finer spacings near the ends of the interval.

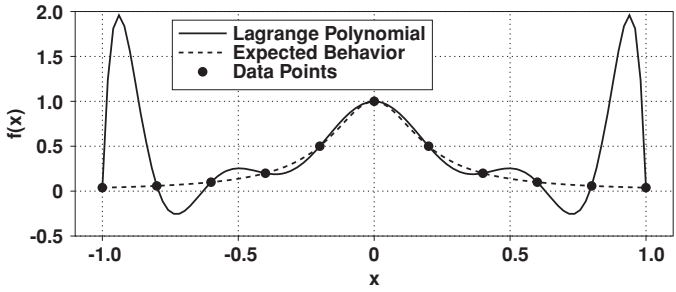

Figure 1.1 Lagrange polynomial interpolation of Runge's function using eleven equally spaced data points.

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-71123-4 - Fundamentals of Engineering Numerical Analysis, Second Edition
Parviz Moin
Excerpt
More information

EXAMPLE 1.2 Lagrange Interpolation With Non-equally Spaced Data

Consider the following data which are again extracted from the Runge's function of Example 1.1. The same number of points are used as in Example 1.1, but the data points x_i are now more finely spaced near the ends (at the expense of coarser resolution near the center).

x_i	−1.00	−0.95	−0.81	−0.59	−0.31	0.00	0.31	0.59	0.81	0.95	1.00
y_i	0.038	0.042	0.058	0.104	0.295	1.00	0.295	0.104	0.058	0.042	0.038

The interpolation polynomial and the expected curve, which in this case (as in Example 1.1) is simply the Runge's function, are plotted in Figure 1.2. It is apparent that the magnitudes of the overshoots at the ends of the interval have been reduced; however, the overall accuracy of the scheme is still unacceptable.

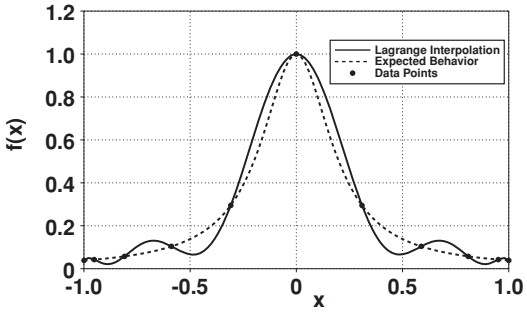

Figure 1.2 Lagrange polynomial interpolation of Runge's function using eleven non-equally spaced data points. The data toward the ends of the interval are more finely spaced.

The wandering problem can also be severely curtailed by *piecewise Lagrange* interpolation. Instead of fitting a single polynomial of degree n to all the data, one fits lower order polynomials to sections of it. This is used in many practical applications and is the basis for some numerical methods. The main problem with piecewise Lagrange interpolation is that it has discontinuous slopes at the boundaries of the segments, which causes difficulties when evaluating the derivatives at the data points. Interpolation with cubic splines circumvents this difficulty.

1.2 Cubic Spline Interpolation

Interpolation with cubic splines is essentially equivalent to passing a flexible plastic ruler through the data points. You can actually hammer a few nails partially into a board and pretend that they are a set of data points; the nails can then hold a plastic ruler that is bent to touch all the nails. Between the nails, the ruler acts as the interpolating function. From mechanics the equation governing

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-71123-4 - Fundamentals of Engineering Numerical Analysis, Second Edition
Parviz Moin
Excerpt
More information

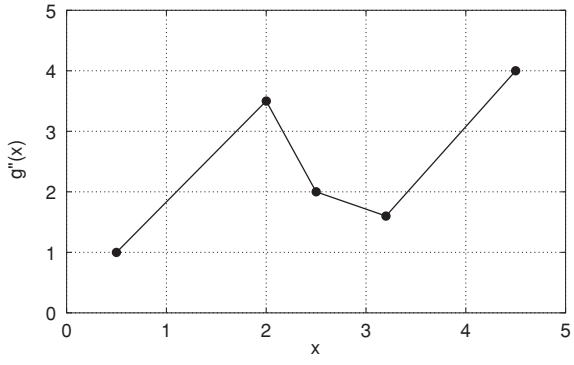

Figure 1.3 A schematic showing the linearity of g'' in between the data points. Also note that with such a construction, g'' is continuous at the data points.

the position of the curve $y(x)$ traced by the ruler is

$$Cy^{(iv)} = G,$$

where C depends on the material properties and G represents the applied force necessary to pass the spline through the data. The force is applied only at the data points; between the data points the force is zero. Therefore, the spline is piecewise cubic between the data. As will be shown below, the spline interpolant and its *first two derivatives are continuous at the data points.*

Let $g_i(x)$ be the cubic in the interval $x_i \le x \le x_{i+1}$ and let $g(x)$ denote the collection of all the cubics for the entire range of x. Since g is piecewise cubic its second derivative, g'', is piecewise linear. For the interval $x_i \le x \le x_{i+1}$, we can write the equation for the corresponding straight line as

$$g_i''(x) = g''(x_i)\frac{x - x_{i+1}}{x_i - x_{i+1}} + g''(x_{i+1})\frac{x - x_i}{x_{i+1} - x_i}. \tag{1.3}$$

Note that by construction, in (1.3) we have enforced the continuity of the second derivative at the data points. That is, as shown in Figure 1.3, straight lines from the adjoining intervals meet at the data points.

Integrating (1.3) twice we obtain

$$g_i'(x) = \frac{g''(x_i)}{x_i - x_{i+1}}\frac{(x - x_{i+1})^2}{2} + \frac{g''(x_{i+1})}{x_{i+1} - x_i}\frac{(x - x_i)^2}{2} + C_1 \tag{1.4}$$

and

$$g_i(x) = \frac{g''(x_i)}{x_i - x_{i+1}}\frac{(x - x_{i+1})^3}{6} + \frac{g''(x_{i+1})}{x_{i+1} - x_i}\frac{(x - x_i)^3}{6} + C_1 x + C_2. \tag{1.5}$$

The undetermined constants C_1 and C_2 are obtained by matching the functional values at the end points:

$$g_i(x_i) = f(x_i) \equiv y_i \qquad g_i(x_{i+1}) = f(x_{i+1}) \equiv y_{i+1},$$

which give two equations for the two unknowns, C_1 and C_2. Substituting for C_1

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-71123-4 - Fundamentals of Engineering Numerical Analysis, Second Edition
Parviz Moin
Excerpt
More information

and C_2 in (1.5) leads to the spline equation used for interpolation:

$$g_i(x) = \frac{g''(x_i)}{6}\left[\frac{(x_{i+1}-x)^3}{\Delta_i} - \Delta_i(x_{i+1}-x)\right] + \frac{g''(x_{i+1})}{6}\left[\frac{(x-x_i)^3}{\Delta_i} - \Delta_i(x-x_i)\right] + f(x_i)\frac{x_{i+1}-x}{\Delta_i} + f(x_{i+1})\frac{x-x_i}{\Delta_i}, \tag{1.6}$$

where $x_i \le x \le x_{i+1}$ and $\Delta_i = x_{i+1} - x_i$. In (1.6) $g''(x_i)$ and $g''(x_{i+1})$ are still unknowns. To obtain $g''(x_i)$, we use the remaining matching condition, which is the continuity of the first derivatives:

$$g_i'(x_i) = g_{i-1}'(x_i).$$

The desired system of equations for $g''(x_i)$ is then obtained by differentiating $g_i(x)$ and $g_{i-1}(x)$ from (1.6) and equating the two derivatives at $x = x_i$. This leads to

$$\frac{\Delta_{i-1}}{6}g''(x_{i-1}) + \frac{\Delta_{i-1}+\Delta_i}{3}g''(x_i) + \frac{\Delta_i}{6}g''(x_{i+1}) = \frac{f(x_{i+1})-f(x_i)}{\Delta_i} - \frac{f(x_i)-f(x_{i-1})}{\Delta_{i-1}} \qquad i = 1, 2, 3, \ldots, N-1. \tag{1.7}$$

These are $N-1$ equations for the $N+1$ unknowns $g''(x_0), g''(x_1), \ldots, g''(x_N)$. The equations are in tridiagonal form and diagonally dominant, and therefore they can be solved very efficiently. The remaining equations are obtained from the prescription of some "end conditions." Typical conditions are:

a) Free run-out (natural spline):

$$g''(x_0) = g''(x_N) = 0.$$

This is the most commonly used condition. It can be shown that with this condition, the spline is the smoothest interpolant in the sense that the integral of g''^2 over the whole interval is smaller than any other function interpolating the data.

b) Parabolic run-out:

$$g''(x_0) = g''(x_1)$$
$$g''(x_{N-1}) = g''(x_N).$$

In this case, the interpolating polynomials in the first and last intervals are parabolas rather than cubics (see Exercise 3).

c) Combination of (a) and (b):

$$g''(x_0) = \alpha g''(x_1)$$
$$g''(x_{N-1}) = \beta g''(x_N),$$

where α and β are constants chosen by the user.

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-71123-4 - Fundamentals of Engineering Numerical Analysis, Second Edition
Parviz Moin
Excerpt
More information

d) Periodic:

$$g''(x_0) = g''(x_{N-1})$$
$$g''(x_1) = g''(x_N).$$

This condition is suitable for interpolating in one period of a known periodic signal.

The general procedure for spline interpolation is first to solve the system of equations (1.7) with the appropriate end conditions for $g''(x_i)$. The result is then used in (1.6), providing the interpolating function $g_i(x)$ for the interval $x_i \leq x \leq x_{i+1}$. In general, spline interpolation is preferred over Lagrange polynomial interpolation; it is easy to implement and usually leads to smooth curves.

EXAMPLE 1.3 Cubic Spline Interpolation

We will now interpolate the data in Example 1.1 with a cubic spline. We solve the tridiagonal system derived in (1.7). Since the data are uniformly spaced, this equation takes a particularly simple form for $g''(x_i)$:

$$\frac{1}{6}g''(x_{i-1}) + \frac{2}{3}g''(x_i) + \frac{1}{6}g''(x_{i+1}) = \frac{y_{i+1} - 2y_i + y_{i-1}}{\Delta^2} \qquad i = 1, 2, \ldots, n-1.$$

For this example, we will use the free run-out condition $g''(x_0) = g''(x_n) = 0$. The cubic spline is evaluated at several x points using (1.6) and the $g''(x_i)$ values obtained from the solution of this tridiagonal system. The subroutine `spline` in *Numerical Recipes* has been used in the calculation. The equivalent function in MATLAB is also called `spline`. The result is presented in Figure 1.4. Spline representation appears to be very smooth and is virtually indistinguishable from Runge's function.

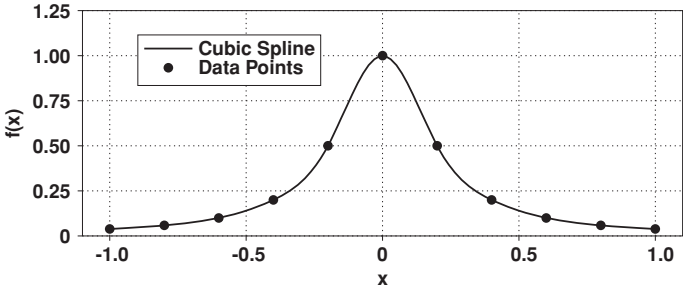

Figure 1.4 Cubic spline interpolation of Runge's function using the equally spaced data of Example 1.1.

Clearly spline interpolation is much more accurate than Lagrange interpolation. Of course, the computer program for spline is longer and a bit more complicated than that for Lagrange interpolation. However, once such programs are written for general use, then the time taken to develop the program, or the "human cost," no longer enters into consideration.

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-71123-4 - Fundamentals of Engineering Numerical Analysis, Second Edition
Parviz Moin
Excerpt
More information

An interesting version of spline interpolation, called tension spline, can be used if the spline fit wiggles too much. The idea is to apply some tension or pull from both ends of the flexible ruler discussed at the beginning of this section. Mathematically, this also leads to a tridiagonal system of equations for g_i'', but the coefficients are more complicated. In the limit of very large tension, all the wiggles are removed, but the spline is reduced to a simple straight line interpolation (see Exercise 6).

EXERCISES

1. Write a computer program for Lagrange interpolation (you may want to use the *Numerical Recipes* subroutine `polint` or `interp1` of MATLAB). Test your program by verifying that $P(0.7) = -0.226$ in Example 1.1.
   (a) Using the data of Example 1.1, find the interpolated value at $x = 0.9$.
   (b) Use Runge's function to generate a table of 21 equally spaced data points. Interpolate these data using a Lagrange polynomial of order 20. Plot this polynomial and comment on the comparison between your result and the plot of Example 1.1.
2. Derive an expression for the derivative of a Lagrange polynomial of order n at a point x between the data points.
3. Show that if parabolic run-out conditions are used for cubic spline interpolation, then the interpolating polynomials in the first and last intervals are indeed parabolas.
4. An operationally simpler spline is the so-called quadratic spline. Interpolation is carried out by piecewise quadratics.
   (a) What are the suitable joint conditions for quadratic spline?
   (b) Show how the coefficients of the spline are obtained. What are suitable end conditions?
   (c) Compare the required computational efforts for quadratic and cubic splines.
5. Consider a set of $n + 1$ data points $(x_0, f_0), \ldots, (x_n, f_n)$, equally spaced with $x_{i+1} - x_i = h$. Discuss how cubic splines can be used to obtain a numerical approximation for the first derivative f' at these data points. Give a detailed account of the required steps. You should obtain formulas for the numerical derivative at the data points $x_0, \ldots, x_n$ and explain how to calculate the terms in the formulas.
6. Tension splines can be used if the interpolating spline wiggles too much. In this case, the equation governing the position of the plastic ruler in between the data points is

$$y^{(iv)} - \sigma^2 y'' = 0$$

   where σ is the tension parameter. If we denote $g_i(x)$ as the interpolating tension spline in the interval $x_i \le x \le x_{i+1}$, then $g_i''(x) - \sigma^2 g_i(x)$ is a straight line in

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-71123-4 - Fundamentals of Engineering Numerical Analysis, Second Edition
Parviz Moin
Excerpt
More information

this interval, which can be written in the following convenient forms:

$$g_i''(x) - \sigma^2 g_i(x) = [g''(x_i) - \sigma^2 f(x_i)]\frac{x - x_{i+1}}{x_i - x_{i+1}} + [g''(x_{i+1}) - \sigma^2 f(x_{i+1})]\frac{x - x_i}{x_{i+1} - x_i}.$$

(a) Verify that for $\sigma = 0$, the cubic spline is recovered, and $\sigma \to \infty$ leads to linear interpolation.

(b) Derive the equation for tension spline interpolation, i.e., the expression for $g_i(x)$.

7. The tuition for 12 units at *St. Anford* University has been increasing from 1998 to 2008 as shown in the table below:

Year	Tuition per year
1998	$21,300
1999	$23,057
2000	$24,441
2001	$25,917
2002	$27,204
2003	$28,564
2004	$29,847
2005	$31,200
2006	$32,994
2007	$34,800
2008	$36,030

(a) Plot the given data points and intuitively interpolate (draw) a smooth curve through them.

(b) Interpolate the data with the Lagrange polynomial. Plot the polynomial and the data points. Use the polynomial to predict the tuition in 2010. This is an extrapolation problem; discuss the utility of Lagrange polynomials for extrapolation.

(c) Repeat (b) with a cubic spline interpolation and compare your results.

8. The concentration of a certain toxin in a system of lakes downwind of an industrial area has been monitored very accurately at intervals from 1993 to 2007 as shown in the table below. It is believed that the concentration has varied smoothly between these data points.

Year	Toxin Concentration
1993	12.0
1995	12.7
1997	13.0
1999	15.2
2001	18.2
2003	19.8
2005	24.1
2007	28.1
2009	???

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-71123-4 - Fundamentals of Engineering Numerical Analysis, Second Edition
Parviz Moin
Excerpt
More information

(a) Interpolate the data with the Lagrange polynomial. Plot the polynomial and the data points. Use the polynomial to predict the condition of the lakes in 2009. Discuss this prediction.
(b) Interpolation may also be used to fill "holes" in the data. Say the data from 1997 and 1999 disappeared. Predict these values using the Lagrange polynomial fitted through the other known data points.
(c) Repeat (b) with a cubic spline interpolation. Compare and discuss your results.

9. Consider a piecewise Lagrange polynomial that interpolates between three points at a time. Let a typical set of consecutive three points be x_{i-1}, x_i, and x_{i+1}. Derive differentiation formulas for the first and second derivatives at x_i. Simplify these expressions for uniformly spaced data with $\Delta = x_{i+1} - x_i$. You have just derived finite difference formulas for discrete data, which are discussed in the next chapter.

10. Consider a function f defined on a set of $N+1$ discrete points

$$x_0 < x_1 < \cdots < x_N.$$

We want to derive an $(N+1) \times (N+1)$ matrix, D (with elements d_{ij}), which when multiplied by the vector of the values of f on the grid results in the derivative of f' at the grid points. Consider the Lagrange polynomial interpolation of f in (1.2):

$$P(x) = \sum_{j=0}^{N} y_j L_j(x).$$

We can differentiate this expression to obtain P'. We seek a matrix D such that

$$D\boldsymbol{f} = \boldsymbol{P}'_N$$

where, $\boldsymbol{P}'_N$ is a vector whose elements are the derivative of $P(x)$ at the data points. Note that the derivative approximation given by $D\boldsymbol{f}$ is exact for all polynomials of degree N or less. We define D such that it gives the exact derivatives for all such polynomials at the $N+1$ grid points. That is, we want

$$D \underbrace{L_k(x_j)}_{\delta_{kj}} = L'_k(x_j) \qquad j,\ k = 0,\ 1,\ 2, \ldots,\ N$$

where δ_{kj} is Kronecker delta which is equal to one for $k = j$ and zero for $k \neq j$. Show that this implies that

$$d_{jk} = \frac{d}{dx} L_k \bigg|_{x=x_j}, \tag{1}$$

where d_{jk} are the elements of D. Evaluate the right-hand side of (1) and show that

$$d_{jk} = L'_k(x_j) = \alpha_k \prod_{\substack{l=0 \\ l \neq j,k}}^{N} (x_j - x_l) = \frac{\alpha_k}{\alpha_j (x_j - x_k)} \qquad \text{for } j \neq k, \tag{2}$$

© in this web service Cambridge University Press www.cambridge.org