# ONE-STEP METHODS FOR THE NUMERICAL SOLUTION OF LINEAR DIFFERENTIAL EQUATIONS BASED UPON LOBATTO QUADRATURE FORMULAE

K. D. SHARMA

(Received 8 April 1968; revised 26 June 1968)

## 1. Introduction

The necessity of accurate numerical approximations to the solutions of differential equations governing physical systems has always been an important problem with scientists and engineers. Hammer and Hollingsworth [11] have used Gaussian quadrature for solving the linear second order differential equations. This method has been further developed by Morrison and Stoller [3], Korganoff [1], Kuntzman [9], Henrici [12] and Day [7, 8]. Quadrature methods based upon Lobatto quadrature formulae have recently been considered by Day [6, 8] and Jain and Sharma [10] and seem to give better results.

The purpose of this paper is to investigate the use of Lobatto quadrature formulae for developing one-step methods, of sufficiently high orders, for the numerical integration of the differential equation

$$y''(x) = f(x)y(x)+g(x); \quad y(x_0) = y_0, \quad y'(x_0) = y'_0.$$

The numerical integration of this differential equation is often carried out by solving a system of two first order differential equations but this approach limits down the accuracy that can be achieved otherwise for its approximate solution. In order to deal with higher order differential equations the method has been extended for the linear system of differential equations $Y' = AY+B$; $Y(x_0) = Y_0$. A stability criterion is also discussed for a second order differential equation. Two sample examples are used to illustrate the comparison of this method with other well known methods and with the exact solution.

## 2. Second order differential equations

Let us consider the general second order linear differential equation in the normal form

https://doi.org/10.1017/S1446788700006054 Published online by Cambridge University Press

$$y''(x) = f(x)y(x)+g(x) \tag{2.1}$$

with the initial conditions

$$y(x_0) = y_0, \quad y'(x_0) = y'_0. \tag{2.2}$$

Throughout our discussion we shall assume that the initial-value problem (2.1, 2.2) has a unique solution in the interval $(a, b)$ and the functions $f(x)$ and $g(x)$ occurring in (2.1) are sufficiently differentiable. These assumptions ensure the validity of our subsequent derivations in any context in which they are used.

After integrating (2.1) from $x_0$ to $x_0+h$ $(h > 0)$ we obtain the system of integral equations

$$y(x_0+h) = y(x_0)+hy'(x_0)+\int_{x_0}^{x_0+h}(x_0+h-\tau). \quad [f(\tau)y(\tau)+g(\tau)]d\tau. \tag{2.3}$$

$$y'(x_0+h) = y'(x_0)+\int_{x_0}^{x_0+h}[f(\tau)y(\tau)+g(\tau)]d\tau. \tag{2.4}$$

Although the integrals in (2.3) and (2.4) may now be approximated by any quadrature formula but to obtain high accuracy integration formulae with few function evaluations we shall make use of a class of quadrature formulae (known as Lobatto quadrature formulae ref. [4]) which utilize the values of the integrand at some irregularly spaced interior points in addition to the values of the integrand at the end points of the interval of integration, which in the present case are known to us.

The $n$-point Lobatto quadrature formula can be expressed in the form

$$\int_{x_0}^{x_0+h} F(x)dx = \frac{h}{n(n-1)}[F(x_0)+F(x_0+h)]+h\sum_{k=1}^{n-2} W_k F(x_0+t_k h)+R \tag{2.5}$$

where

$$R = -\frac{n(n-1)^3[(n-2)!]^4 h^{2n-1}}{(2n-1)[(2n-2)!]^3} F^{(2n-2)}(\xi)$$

$$0 < t_k < 1, \quad x_0 < \xi < x_0+h.$$

In order to shorten the succeeding equations and calculations we denote

$$\begin{array}{llll} \tau_0 = x_0; & \tau_{n-1} = x_1 = x_0+h; & \tau_k = x_0+t_k h & 0 < t_k < 1 \\ y(x_0) = y_0; & y(x_1) = y_{n-1}; & y(\tau_k) = y_k & k = 1(1)n-1 \\ f(x_0) = f_0; & f(x_1) = f_{n-1}; & f(\tau_k) = f_k & k = 1(1)n-1 \quad \text{etc.} \end{array}$$

Approximation of the integrals in (2.3) and (2.4) by (2.5) leads us to the equations

https://doi.org/10.1017/S1446788700006054 Published online by Cambridge University Press

$$
\begin{aligned}
(2.6)\qquad y_{n-1} &= y_0+hy_0'+\frac{h^2}{n(n-1)}(f_0y_0+g_0)\\
&\qquad+h^2\sum_{k=1}^{n-2}W_k(1-t_k)(f_ky_k+g_k)+E
\end{aligned}
$$

$$
\begin{aligned}
(2.7)\qquad y_{n-1}' &= y_0'+\frac{h}{n(n-1)}(f_0y_0+g_0+f_{n-1}y_{n-1}+g_{n-1})\\
&\qquad+h\sum_{k=1}^{n-2}W_k(f_ky_k+g_k)+E^*
\end{aligned}
$$

where

$$(2.8)\qquad E = -\frac{n(n-1)^3[(n-2)!]^4h^{2n-1}}{(2n-1)[(2n-2)!]^3}\,[(x_1-\tau)y''(\tau)]_{\tau=\xi_n}^{(2n-2)}$$

$$(2.9)\qquad E^* = -\frac{n(n-1)^3[(n-2)!]^4h^{2n-1}}{(2n-1)[(2n-2)!]^3}\,y^{(2n)}(\xi_n')$$

$$x_0 < \xi_n,\quad \xi_n' < x_1.$$

We note that we do not know $y(\tau_k)$ which occur in (2.6) and (2.7) and thus if such an algorithm is to be of computational value we must obtain accurate approximate values for $y(\tau_k)$ $(k = 1, 2, \cdots, n-2)$. For this we construct a Hermite interpolating polynomial $Y(x)$, (approximating $y(x)$) which coincides with $y_0$, $y_0'$ and $y''(\tau_k)$ for $k = 0, (1), n-1$. These $n+2$ conditions define the polynomial $Y(x)$ of $(n+1)$st degree uniquely and can be written in the form

$$
(2.10)\qquad Y(x) = \begin{vmatrix}
y(x) & 1 & x & x^2 & x^2 & \cdot\;\cdot & x^{n+1}\\
y(\tau_0) & 1 & \tau_0 & \tau_0^2 & \tau_0^3 & \cdot\;\cdot & \tau_0^{n+1}\\
y'(\tau_0) & 0 & 1 & 2\tau_0 & 3\tau_0^2 & \cdot\;\cdot & (n+1)\tau_0^n\\
y''(\tau_0) & 0 & 0 & 2 & 6\tau_0 & \cdot\;\cdot & n(n+1)\tau_0^{n-1}\\
y''(\tau_1) & 0 & 0 & 2 & 6\tau_1 & \cdot\;\cdot & n(n+1)\tau_1^{n-1}\\
 & \cdot & \cdot & \cdot & \cdot & \cdot\;\cdot & \cdot\\
y''(\tau_{n-1}) & 0 & 0 & 2 & 6\tau_{n-1} & \cdot\;\cdot & n(n+1)\tau_{n-1}^{n-1}
\end{vmatrix}
$$

If we make use of the given differential equation (2.1) to replace $y''(\tau_k)$ by $(f_ky_k+g_k)$, we can express $Y(x)$ in terms of $y_0'$, $f_k$, $g_k$ and $y(\tau_k)$ $k = 1, 2, \cdots, n-1$. The resulting polynomial $Y(x)$ can now be written in the form

$$(2.11)\qquad Y(x) = y_0+(x-x_0)y_0'+\tfrac{1}{2}(x-x_0)^2y_0''+\sum_{k=2}^{n-1}C_k(x, f, g)y_k.$$

In this equation we substitute $x = \tau_k$ $(k = 1\ (1)\ n-1)$ and obtain a system of

https://doi.org/10.1017/S1446788700006054 Published online by Cambridge University Press

$n-1$ linear equations in $n-1$ unknowns $Y(\tau_k)$ $(k = 1\ (1)\ n-1)$. These equations can either be solved by the matrix inversion method (the matrix of the coefficients being nonsingular) or the equivalent Cramér's Rule. Since $x_0 < \tau_k < x_0+h$ for all values of $k$, it follows from the theory of polynomial interpolation that the error involved in obtaining the approximate value of $Y_k$ from (2.11) is $O(h^{n+2})$. Substitution of the values in (2.6) and (2.7) gives the approximate values of $y_{n-1}$ and $y'_{n-1}$ with local truncation errors of the order of $h^{n+4}$ and $h^{n+3}$ respectively.

The algorithm to determine the approximate values of $y_{r+1}$ is as follows. Replace $y_0$, $y'_0$, $y''_0$, $\tau_k$ occurring in (2.6, 2.7) by $y_r$, $y'_r$, $y''_r$ and $x_r+t_kh$ respectively. Calculate $Y(x_r+t_kh)$, $(k = 1\ (1)\ n-1)$ from (2.11) according to the new equations. The required approximate values of $y(x_{r+1})$ and $y'(x_{r+1})$ are then given by the equations,

$$\begin{aligned}y_{r+1} = y_r+hy'_r+ \frac{h^2}{n(n-1)}\,&[f(x_r)y(x_r)+g(x_r)]\\ &+h^2\sum_{k=1}^{n-2} W_k(1-t_k)[f(x_r+t_kh)Y(x_r+t_kh)+g(x_r+t_kh)],\end{aligned} \tag{2.12}$$

$$\begin{aligned}y'_{r+1} = y'_r+ \frac{h}{n(n-1)}\,&[f(x_r)y(x_r)+f(x_{r+1})y(x_{r+1})+g(x_r)+g(x_{r+1})]\\ &+h\sum_{k=1}^{n-2} W_k[f(x_r+t_kh)Y(x_r+t_kh)+g(x_r+t_kh)].\end{aligned} \tag{2.13}$$

The method suggested here gives rise to quadrature methods of arbitrary orders and corresponding to $n$ evaluations of the functions $f(x)$ and $g(x)$ one can always construct a one-step method of order $(n+4)$ for the differential equation (2.1). From application point of view we discuss two particular cases corresponding to $n = 4, 5$.

## 3. Two particular cases

Here we shall discuss in brief the methods corresponding to $n = 4, 5$ and derive explicit formulae to obtain the solution of (2.1) with local truncatio errors of $h^7$ and $h^9$ respectively.

### 3.1 *Seventh order method.*

In this case the integrals in (2.3) and (2.4) are approximated by the lobatto four-point rule in which case

$$\int_{x_0}^{x_0+h} F(x)dx = h\sum_{k=1}^{4} W_k F(\tau_k)+R_4 \tag{3.1.1}$$

where $W_1 = W_4 = \frac{1}{12}$; $W_2 = W_3 = \frac{1}{12}$

https://doi.org/10.1017/S1446788700006054 Published online by Cambridge University Press

$$\begin{aligned} \tau_1 &= x_0 \\ \tau_2 &= x_0+ph \qquad & p &= (5-\sqrt{5})/10 \\ \tau_3 &= x_0+qh \qquad & q &= (5+\sqrt{5})/10 \\ \tau_4 &= x_0+h \end{aligned}$$

and

$$R_4 = -\frac{4h^7 F^{(6)}(\xi)}{3\cdot 2^7\cdot 15750} \qquad x_0 < \xi < x_0+h.$$

The equations (2.6) and (2.7) take up the form

$$\text{(3.1.2)}\qquad y_1 = y_0+hy_0'+\frac{h^2}{12}[f_0y_0+g_0+5(f_py_p+g_p)q+5(f_qy_q+g_q)p]+E_7,$$

$$\text{(3.1.3)}\qquad y_1' = y_0'+\frac{h}{12}[f_0y_0+g_0+f_1y_1+g_1+5(f_py_p+f_qy_q+g_p+g_q)]+E_7^*.$$

where $x_p = \tau_2$, $x_q = \tau_3$ and $f_p = f(x_p)$ etc. and $E_7$ and $E_7^*$ are given by (2.8) and (2.9) with $n = 4$ and $x_0 < \xi_4,\ \xi_4' < x_0+h$.

The Hermite interpolating polynomial $Y(x)$ satisfying the conditions $Y(x_0) = y_0$; $Y'(x_0) = y_0'$; $Y''(\tau_k) = y''(\tau_k)$ $k = 1, 2, 3, 4$ is given by

$$\text{(3.1.4)}\qquad \begin{aligned} Y(x_0+th) = {} & y_0+thy_0'+h^2y_0''(6t^2-12t^3+10t^4-3t^5)/12 \\ & +h^2y''(x_p)(10qt^3-5(q+1)t^4+3t^5)/[12(q-p)] \\ & +h^2y''(x_q)(10pt^3-5(p+1)t^4+3t^5)/[12(p-q)] \\ & +h^2y''(x_1)(2t^3-5t^4+3t^5)/12 \\ & +h^6y_0^{(6)}(t^3-3t^4+3t^5)/6!+O(h^7). \end{aligned}$$

We now neglect the terms $O(h^6)$ in (3.1.4) and make use of (2.1) to express $y_p''$, $y_q''$, $y_1''$ in terms of $y_p$, $y_q$ and $y_1$. If we now take $t = p, q, 1$ we obtain three linear equations in the three required unknowns. The solution of these equations gives the values of $Y_p$ and $Y_q$ as follows (here $u$ denotes $\sqrt{5}$)

(3.1.5)

$$\begin{aligned} Y_p = {} & 2[120u+h^2f_q(25-13u)][\{(12+h^2f_0)h^2uf_1-6h^2f_0(125+27u)-18000\}y_0 \\ & +12hy_0'\{h^2uf_1-1500q\}]/(5\Delta)+10u[7200+h^2f_q\{h^2f_1(u-1)-144\}] \\ & [(24+h^2f_0)y_0+12hy_0']/\Delta+1296\cdot 10^6\cdot G/(h^2f_1\Delta) \end{aligned}$$

(3.1.6)

$$\begin{aligned} Y_q = {} & -10h^2f_pu[h^2f_1(u+1)-12(53+25u)][(24+h^2f_0)y_0+12hy_0']/\Delta \\ & +2[h^2f_p(25+13u)-120u][\{(12+h^2f_0)h^2f_1u-6h^2f_0(125+27u)-18000\}y_0 \\ & +12h\{h^2uf_1-1500q\}y_0']/(5\Delta)+129\cdot 10^6\cdot G'/(h^2f_1\Delta) \end{aligned}$$

where

https://doi.org/10.1017/S1446788700006054 Published online by Cambridge University Press

$$\begin{aligned}\Delta = {} & [120u - h^2 f_p(25+13u)][7200 + h^2 f_q(h^2 f_1(u-1)-144)] \\ & - h^2 f_p[120u + h^2 f_q(25-13u)][h^2 f_1(u+1) - 12(53+25u)]\end{aligned}$$

and $G$ and $G'$ are functions of $g$, given by the matrix equation

$$\begin{pmatrix} G \\ G' \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} \begin{pmatrix} G_1 \\ G_2 \end{pmatrix}$$

with

$$\begin{aligned} C_{11} &= -(7200 + h^2 f_q(h^2 f_1(u-1) - 144))/7200 \\ C_{12} &= h^2 f_1[120u + h^2 f_q(25-13u)]/180000 \\ C_{21} &= h^2 f_p[h^2 f_1(u+1) - 300u - 636]/7200 \\ C_{22} &= -h^2 f_1[120u - h^2 f_p(25+13u)]/180000 \\ G_1 &= -h^4 u f_1[10g_0 + (13+5u)g_p + (13-5u)g_q]/180000 \\ G_2 &= h^2[(h^2 u f_1 - 162u - 750)g_0 + (5h^2 f_1 qu - 750u - 1590)g_p \\ &\quad + (5h^2 f_1 pu - 360)g_q - 12g_1 u]/180000. \end{aligned}$$

Substitution of (3.1.5) and (3.1.6) in (3.1.2) and (3.1.3) gives us the values of $y(x_0+h)$ and $y'(x_0+h)$ with local truncation error $O(h^7)$.

The algorithm to determine $y_{n+1}$ is as follows. Replace $y(x_0)$, $y'(x_0)$, $y''(x_0)$, $x_p$, $x_q$ occurring in (3.1.2) and (3.1.3) by $y_n$, $y'_n$, $y''_n$, $x_{n+p}$ $(= x_n+ph)$ and $x_{n+q}$ $(= x_n+qh)$ respectively and calculate $Y_p$ and $Y_q$ according to the new equations. Denoting these values by $Y_{n+p}$ and $Y_{n+q}$, the required approximate values of $y_{n+1}$ and $y'_{n+1}$ are given by .

(3.1.7)
$$\begin{aligned} y_{n+1} = {} & y_n + hy'_n + \frac{h^2}{12}[f(x_n)y_n + g(x_n) + 5\{q(f(x_{n+p})Y_{n+p} \\ & + g(x_{n+p})) + p(f(x_{n+q})Y_{n+q} + g(x_{n+q}))\}], \end{aligned}$$

(3.1.8)
$$\begin{aligned} y'_{n+1} = {} & y'_n + \frac{h}{12}[f(x_n)y_n + g(x_n) + f(x_{n+1})y_{n+1} + g(x_{n+1}) \\ & + 5\{f(x_{n+p})Y_{n+p} + f(x_{n+q})Y_{n+q} + g(x_{n+p}) + g(x_{n+q})\}]. \end{aligned}$$

### 3.2 *Ninth-order method*

The derivation of this ninth-order method is similar to the previous method but to achieve higher accuracy by suitably minimizing the round-off error at every step, it is suggestive to make use of the results of the following algorithm.

In this case we shall use the lobatto five point rule, for which the values of the weights and abscissas are given by

https://doi.org/10.1017/S1446788700006054 Published online by Cambridge University Press

$$\begin{aligned}
&W_1 = W_5 = \tfrac{1}{20}; \qquad W_2 = W_4 = \tfrac{49}{180}; \qquad W_3 = \tfrac{16}{45}\\
&\tau_1 = x_0\\
&\tau_2 = x_0+rh \qquad\qquad r = (7-\sqrt{21})/14\\
&\tau_3 = x_0+mh \qquad\qquad m = \tfrac{1}{2}\\
&\tau_4 = x_0+sh \qquad\qquad s = (7+\sqrt{21})/14\\
&\tau_5 = x_0+h\\
&R = -\frac{h^9F^{(8)}(\xi)}{1432729600}
\end{aligned} \tag{3.2.1}$$

The equations (2.6) and (2.7) take the form

$$\begin{aligned}
y_1 = y_0+hy_0'+\frac{h^2}{180}&[9(f_0y_0+g_0)+49\{s(f_ry_r+g_r)\\
&+r(f_sy_s+g_s)\}+32(f_my_m+g_m)]+E_9,
\end{aligned} \tag{3.2.2}$$

$$\begin{aligned}
y_1' = y_0'+\frac{h}{180}&[9(f_0y_0+g_0+f_1y_1+g_1)+49(f_ry_r+g_r+f_sy_s+g_s)\\
&+64(f_my_m+g_m)+E_9^*
\end{aligned} \tag{3.2.3}$$

where $E_9$ & $E_9^*$ are given by (2.8) and (2.9) with $n = 5$ and $x_0 < \xi_5,\ \xi_5' < x_0+h$. The Hermite interpolating polynomial satisfying the conditions $Y(x_0) = y_0$; $Y'(x_0) = y_0'$; $Y''(\tau_k) = y''(\tau_k)$ for $k = 1\ (1)\ 5$ is given by

$$\begin{aligned}
Y(x_0+th) = y_0+hy_0'+h^2y_0''&(30t^2-100t^3+150t^4-105t^5+28t^6)/60\\
&+49h^2y''(x_r)(120st^3-60(3s+1)t^4\\
&\qquad\qquad+36(3+2s)t^5-48t^6)/2160\\
&+8h^2y''(x_m)(-5t^3+20t^4-21t^5+7t^6)/45\\
&+49h^2y''(x_s)(120rt^3-60(3r+1)t^4\\
&\qquad\qquad+36(3+2r)t^5-48t^6)/2160\\
&+h^2y''(x_1)(-10t^3+45t^4-63t^5+28t^6)/60\\
&+h^7y_0^{(7)}(-t^3+5t^4-9t^5+7t^6)/10080+O(h^8).
\end{aligned} \tag{3.2.4}$$

We neglect the error terms in (3.2.4) and make use of (2.1) to replace $y''(\tau_k)$ in terms of $y(\tau_k)$, $f(\tau_k)$, $g(\tau_k)$. We now substitute $t = r, m, s, 1$ in this equation and obtain four linear equations in four unknowns $y_r$, $y_m$, $y_s$, $y_1$. The values of these quantities are given by the matrix equation (here $v$ denotes $\sqrt{21}$)

https://doi.org/10.1017/S1446788700006054 Published online by Cambridge University Press

$$(3.2.5)\qquad \begin{bmatrix} Y_r \\ Y_m \\ Y_s \\ Y_1 \end{bmatrix} = -P^{-1} \begin{bmatrix} y_0 + rhy_0' + y_0''(145-21v)/5880 + h^2G_1 \\ y_0 + \frac{1}{2}hy_0' + 49y_0''/1920 + h^2G_2 \\ y_0 + shy_0' + y_0''(145+21v)/5880 + h^2G_3 \\ y_0 + hy_0' + h^2y_0''/20 + h^2G_4 \end{bmatrix}$$

where

$$(3.2.6)\qquad \begin{aligned} G_1 &= [420g_r + 24(124-28v)g_m + 21(227-49v)g_s - 18g_1]/52920 \\ G_2 &= [(245+56v)g_r + 80g_m + (245-56v)g_s + 3g_1]/5760 \\ G_3 &= [21(227+49v)g_r + 24(124+28v)g_m + 420g_s - 18g_1]/52920 \\ G_4 &= [49sg_r + 32g_m + 49rg_s]/180 \end{aligned}$$

and the matrix $P$ is given by

$$(3.2.7)\qquad P = \frac{1}{282240}$$

$$\begin{bmatrix} 2240(h^2f_r - 126) & 128(124-28v)h^2f_m & 112(227-49v)h^2f_s & -96h^2f_1 \\ 49(245+56v)h^2f_r & 3920(h^2f_m - 72) & 49(245-56v)h^2f_s & 147h^2f_1 \\ 112(227+49v)h^2f_r & 128(124+28v)h^2f_m & 2240(h^2f_s - 126) & -96h^2f_1 \\ 76832sh^2f_r & 50176h^2f_m & 76832h^2rf_s & -282240 \end{bmatrix}$$

It can be easily seen that the matrix $P$ (for small values of $h$) is non-singular and therefore the system (3.2.5) can be solved for the required unknowns. The algorithm to determine $y_{n+1}$ is as follows.

Replace $y_0$, $y_0'$, $y_0''$, $x_r$, $x_m$, $x_s$ occurring in (3.2.2) and (3.2.3) by $y_n$, $y_n'$, $y_n''$, $x_{n+r}$, $x_{n+m}$, $x_{n+s}$ respectively and calculate $Y_r$, $Y_m$, $Y_s$ according to the new equations. Denoting these quantities by $Y_{n+r}$, $Y_{n+m}$, $Y_{n+s}$ the required approximate values of $y_{n+1}$, $y_{n+1}'$ are given by the equations.

$$(3.2.8)\qquad \begin{aligned} y_{n+1} = y_n + hy_n' + \frac{h^2}{180} & [9(f(x_n)y_n + g(x_n)) \\ & + 32(f(x_{n+m})Y_{n+m} + g(x_{n+m})) \\ & + 49\{s(f(x_{n+r})Y_{n+r} + g(x_{n+r})) + r(f(x_{n+s})Y_{n+s} + g(x_{n+s}))\}], \end{aligned}$$

$$(3.2.9)\qquad \begin{aligned} y_{n+1}' = y_n' + \frac{h}{180} & [9(f(x_n)y_n + g(x_n) + f(x_{n+1})Y_{n+1} + g(x_{n+1})) \\ & + 49(f(x_{n+r})Y_{n+r} + g(x_{n+r}) + f(x_{n+s})Y_{n+s} + g(x_{n+s})) \\ & + 64(f(x_{n+m})Y_{n+m} + g(x_{n+m}))]. \end{aligned}$$

## 4. Stability considerations

It is of course difficult to discuss the stability of the method for the most general case but for a particular case it can be studied in a manner similar to the one adopted by Jain and Sharma [10]. We shall here consider

https://doi.org/10.1017/S1446788700006054 Published online by Cambridge University Press

the stability of the Seventh order method for the differential equation $y'' = \alpha y$, $\alpha$ being a real number. We shall discuss three cases $\alpha = k^2$, $0$, $-k^2$ respectively. If we insert the values of $Y(x_p)$, $Y(x_q)$ from (3.1.5), (3.1.6) in (3.1.2) and (3.1.3) we obtain

$$\begin{bmatrix} y_{n+1} \\ y'_{n+1} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} y_n \\ y'_n \end{bmatrix} \tag{4.1}$$

where

$$a_{11} = 1 + \frac{\alpha h^2}{12} + \frac{\alpha h^2(u-4)}{\Delta'} + \frac{3\alpha^2 h^4}{10\,\Delta'} + \frac{\alpha^3 h^6}{600\,\Delta'} + \frac{\alpha^4 h^8}{360000\,\Delta'}$$

$$a_{12} = h + \frac{(5+u)\alpha h^3}{25\,\Delta'} + \frac{58(u+1)\alpha^2 h^5}{675\,\Delta'} + \frac{31\alpha^3 h^7}{108000\,\Delta'}$$

$$a_{21} = \frac{\alpha h}{12} + \frac{9u\alpha h}{59\,\Delta'} + \frac{11\alpha^2 h^3}{120\,\Delta'} + \frac{\alpha^3 h^5}{1440\,\Delta'}$$

$$a_{22} = 1 + \frac{\alpha h^2}{24\,\Delta'} + \frac{(77+25u)\alpha^2 h^4}{2880\,\Delta'}$$

and

$$\Delta' = 1 - \frac{\alpha h^2}{25} + \frac{\alpha^2 h^4}{1000} - \frac{\alpha^3 h^6}{36000}.$$

For $\alpha = 0$ we have

$$y_{n+1} = y_n + h y'_n$$
$$y'_{n+1} = y'_n.$$

The solution of this system can be written as

$$y'_n = y'_0$$
$$y_n = y_0 + nhy'$$

which is an expected result.

We now consider the case $\alpha = -k^2$; the solutions in this case are oscillating. We therefore consider the eigenvalues of the matrix (4.1) which are given by

$$\lambda_1, \lambda_2 = \tfrac{1}{2}(a_{11}+a_{22}) \pm \tfrac{1}{2}\sqrt{(a_{11}-a_{22})^2 + 4a_{12}a_{21}}.$$

If we substitute the values of the coefficients from the equation (4.1) and carry out the computation of the eigenvalues for a sufficiently wide range of $h^2k^2$, we find that the eigenvalues are real except in the range (approximately)

$$15.3 \leqq h^2k^2 \leqq 16.9 \tag{4.2}$$

in which case the roots are complex. If we further use the value of $\Delta'$

https://doi.org/10.1017/S1446788700006054 Published online by Cambridge University Press

as given by (3.1.10) we find a sub range of (4.2) in which the roots are complex and have a unit modulus. This however indicates that the range of stable operation of the method is very small and therefore the method in general cannot be suggested to be always trustworthy in automatic computation.

For the case $y'' = k^2y$, of which the solutions are exponential in nature, one can again proceed on the lines of Jain and Sharma [10] and discuss the stability of the method. But in this case the calculations seem to be too lengthy to be considered here. Nevertheless it can be seen that the series for $1/\Delta'$ converges if $(h^2k^2)/25 < 1$ and therefore for small values of $h$, stability in this case is also expected.

## 5. Higher-order differential equations

Let us consider the system of linear ordinary differential equations

$$Y' = AY+B, \quad Y(x_0) = Y_0 \tag{5.1}$$

where $A$ is a $m \times m$ matrix with elements as functions of $x$. Integrating (5.1) over the interval $[x_0, x_0+h]$ $(h > 0)$, we get

$$Y(x_0+h) = y(x_0)+\int_{x_0}^{x_0+h} [A(t)Y(t)+B(t)]dt. \tag{5.2}$$

Application of (2.5) to the integral in this equation gives

$$\begin{aligned} Y(x_0+h) = Y_0+ \frac{h}{n(n-1)} &[A(x_0)Y(x_0)+A(x_1)Y(x_1)+B(x_0)+B(x_1)] \\ &+\sum_{k=1}^{n-2} W_k[A(\tau_k)Y(\tau_k)+B(\tau_k)]+R \end{aligned} \tag{5.3}$$

where

$$R = -\frac{n(n-1)^3[(n-2)!]^4h^{2n-1}}{(2n-1)[(2n-2)!]^3} [Y(\tau)A(\tau)+B(\tau)]_{\tau=\xi}^{2n-2}$$

$$x_0 < \xi < x_0+h.$$

To calculate the approximate values of the unknown quantities $Y(\tau_k)$ for $k = 1\ (1)\ n-1$, we can construct the Hermite interpolating polynomial

$$Y(x_0+th) = a_0(t)Y(x_0)+ \sum_{k=1}^{n-1} a_k(t)Y'(\tau_k) \tag{5.4}$$

by requiring that the Taylor series expansion of both the sides about the point $x_0$ agrees with each other up to the terms of order $h^n$. We make use of (5.1) and rewrite (5.4) in the form

https://doi.org/10.1017/S1446788700006054 Published online by Cambridge University Press

$$Y(x_0+th) = a_0(t)Y_0+\sum_{k=1}^{n-1} a_k(t)[A(\tau_k)Y(\tau_k)+B(\tau_k)]. \tag{5.5}$$

Equation (5.5) is linear in the unknown vectors $Y(\tau_k)$ and if we substitute $t = \tau_k$, $k = 1\ (1)\ n-1$ we get a linear system of matrix equations to determine the $n-1$ unknown vectors $Y(\tau_k)$, $k = 1, 2, \cdots, n-1$. These values may now be substituted in (5.3) to obtain the required values of $Y(x_0+h)$ with a local truncation error of $O(h^{n+2})$. The stability of a particular method in this general case can be considered exactly in the same way as in the previous case.

## 6. Illustrations

For the purpose of computational comparison of this method with other methods, we consider the following two sample examples. We have written programs for the ATLAS COMPUTER in FORTRAN (in single precision) for the methods under consideration and the results have been compared with the exact results.

Example 1. Bessel differential equation

$$y''(x)+(100+1/(4x^2))y(x) = 0.$$

We take the initial conditions at $x = 1$ such that the exact solution is $\sqrt{x}J_0(10x)$. Starting values were taken from the tables of Bessel Functions in [2] to 10 decimal places. The Runge-kutta method used for comparison is the simple single-step method which as well employs four evaluations of the functions in each of the subintervals. The results obtained by using the algorithms of section 3 with $h = 0.02$ are given in table 1.

Example 2. The differential equation

$$y''(x)+(16\pi^2e^{-2x}-\tfrac{1}{4})y(x) = 0$$

with initial conditions $y(0) = 1$, $y'(0) = 0.5$ has the solution

$$y(x) = e^{\frac{1}{2}x}\cdot\cos\,(4\pi e^{-x}).$$

In this case also we took $h = 0.02$ and used the algorithms of section 3. The results obtained have been listed in Table 2.

## 7. Conclusions

We observe that the one-step Lobatto method developed here (which utilizes very few function evaluations) gives a good agreement of the computed values with the exact values in the case of differential equations and compares quite favourably with the other methods under consideration.

https://doi.org/10.1017/S1446788700006054 Published online by Cambridge University Press

TABLE I

Differential equation $y''+\left(100+1/(4x^2)\right)\,y = 0$

| $X$ | 2.0 | 3.0 | 4.0 | 5.0 | 6.0 |
|---|---|---|---|---|---|
| Runge-Kutta | .2362149 | —.1496406 | .0148322 | .1246736 | —.2239581 |
| Lobatto method [10] | .2362089 | —.1495948 | .0147351 | .1247993 | —.2240596 |
| Present Lobatto method $0(h^7)$ | .2362085458 | —.1495937365 | .0147337825 | .1248001570 | —.2240592448 |
| Present Lobatto method $0(h^9)$ | .2362085456 | —.1495937357 | .0147337811 | .1248001587 | —.2240592459 |
| Exact (10*d*) | .2362085456 | —.1495937357 | .0147337812 | .1248001586 | —.2240592458 |

TABLE II

Differential equation $y''+(16\pi^2 e^{-2x}-\frac{1}{4})\,y = 0$

| $X$ | 1.0 | 2.0 | 3.0 | 4.0 | 5.0 |
|---|---|---|---|---|---|
| Runge-Kutta | —.1471882215 | —.3522760330 | 3.632521597 | 7.193956367 | 12.13858732 |
| Runge-Kutta Lobatto [8] | —.1473306297 | —.3520501715 | 3.632801677 | 7.194209811 | 12.13885935 |
| Present Lobatto method $(h^7)$ | —.1473301023 | —.3520506024 | 3.632798358 | 7.194204132 | 12.13885024 |
| Present Lobatto method $0(h^9)$ | --.1473301030 | —.3520506023 | 3.632798356 | 7.194204131 | 12.13885024 |
| Exact (10*D*) | —.1473301031 | —.3520506031 | 3.632798356 | 7.194204131 | 12.13885025 |

https://doi.org/10.1017/S1446788700006054 Published online by Cambridge University Press

The method has been tried for many other problems and the results obtained were quite satisfactory in all these cases. The method suggests the construction of one-step methods of arbitrary order and it is expected that in the case of differential equation (4.1) one should always get the value of $Y_{r+1}$ with a local truncation error $O(h^{n+2})$ for $n$ evaluations of $A$ and $B$ for arbitrary $n$. However the present method does not find extension for non-linear differential equations, but in this case the reader is suggested to ref. [5].

Although the general second-order differential equation

$$Y'' = P(x)Y' + f(x)Y + g(x),$$

can be considered by the method of section 5, but to make use of the algorithms listed in section 3 it is suggestive to either eliminate the term $Y'$ from the above differential equation or treat the term $P(x)Y'$ by integration by parts, depending upon whether or not $P(x)$ is explicitly integrable.

## Acknowledgements

I would like to express my thanks to Prof. F. H. Sumner, Dept. of Computer Science, University of Manchester, for his constant encouragement during the preparation of this paper and to Prof. T. Kilburn for making me available to the computing facilities to obtain the numerical results of this paper. Thanks are also due to the referee for his valuable suggestions to improve the presentation of this paper.

## References

[1] A. Korganoff, 'Sur les formules d'intégration numérique des équations différentielles donnant une approximation d'ordre élève', *Chiffres*, 1 (1958), 171–180.

[2] British Association for the Advancement of Science, *Mathematical Tables*. Vol. 6, (Cambridge University Press, 1958).

[3] D. Morrison and L. Stoller, 'A method for the numerical integration of ordinary differential equations', *MTAC* 12, (1958), 269–272.

[4] F. B. Hildebrand, *Introduction to numerical analysis*, (Mc-Graw-Hill, New York, 1956).

[5] G. J. Cooper and E. Gal, 'Single-step methods for linear differential equations', *Numerische Mathematik*, 10, (1967), 307–315.

[6] J. T. Day, 'A one-step method for the numerical solution of second order linear ordinary differential equations', *MTAC*, 18, (1964), 664–668.

[7] J. T. Day, 'A one-step method for the numerical integration of the differential equation $y'' = f(x)y+g(x)$', *Computer Journal*, 7, (1964), 314–317.

[8] J. T. Day, 'A Runge-kutta method for the numerical integration of the differential equation $y'' = f(x, y)$' *ZAMM H*. 5. S, (1965), 354–356.

[9] J. Kuntzman, 'Neuere Entwicklungen der Methode von Runge und Kutta' *ZAMM*, 41, (1961), T28–T31.

[10] M. K. Jain and K. D. Sharma, 'Numerical solution of linear differential equations and Volterra's integral equation using Lobatto quadrature formula', *Computer Journal*, 10, (1967), 101–106.

https://doi.org/10.1017/S1446788700006054 Published online by Cambridge University Press

[11] P. C. Hammer and J. W. Hollingsworth, 'Trapezoidal methods of approximating solutions of differential equations', *MTAC*, 9, (1955).

[12] P. Henrici, *Discrete variable methods in ordinary differential equations* (Wiley & Sons, 1962).

Department of Computer Science
University of Manchester
Manchester

Department of Mathematics
Indian Institute of Technology
Hauz Khas, New Delhi-29, India

https://doi.org/10.1017/S1446788700006054 Published online by Cambridge University Press