

# Accurate Piecewise Linear Continuous Approximations to One-Dimensional Curves: Error Estimates and Algorithms

Hiroaki Nishikawa\*

January 1998

## Abstract

Local and global asymptotic  $L_2$  error estimates are derived for piecewise linear continuous approximations to smooth one-dimensional curves in  $\mathbf{R}^n$  ( $n \geq 1$ ). Based on the estimates and an equidistribution strategy, an algorithm to construct a highly accurate piecewise linear approximation to a one-dimensional curve is devised with a special feature of achieving a desired  $L_2$  error. By its generality, the algorithm is applicable to virtually any problems which can be interpreted as numerically approximating a one-dimensional curve. Detailed algorithms are presented and numerical results are shown in four different application areas: piecewise linear approximations to functions in one dimension or curves in higher dimensions; numerical integration; initial and boundary value problems for ordinary differential equations with a particular focus on stiff problems.

## 1 Introduction

In many branches of numerical analysis, piecewise linear continuous functions are often employed to approximate one-dimensional curves, whether or not one intends to do so, such as discretization of boundaries for computational grids in two dimension, interpolation of a function in one dimension, numerical solutions of ordinary differential equations, and so on. In a terminology of modern mathematics, these problems may be categorized as a problem of numerically approximating one-dimensional curves (curves), typically by piecewise linear elements. In any problems of this kind, it is well known that adjusting the node distribution is a very effective way to improve the approximation. A popular strategy is to place nodes so as to equally distribute the arc length of the curve. In White[1], such an approach is studied in detail to improve the numerical solutions of two-point boundary value problems. Another area of the same kind is  $L_2$  fits to functions with adjustable nodes. In this area, since the function is known, algorithms have been developed based on a direct minimization of the  $L_2$  error such as the one in Baines[2]. Numerical integration is yet another area in which the trapezoidal rule can be considered as approximating a function by a piecewise linear continuous function. On the other hand, particularly in the area of numerical initial value problems for ordinary differential equations, a different approach is usually taken that focuses on the error at grid points. Adaptive step-size control techniques have been developed based on an error estimate given by the difference of numerical solutions obtained by two schemes with two consecutive orders of accuracy such as embedded Runge-Kutta methods[6, 8, 9].

Here, we approach these problems from a unified viewpoint that all these problems can be thought of as a numerical approximation to one-dimensional curves by piecewise linear continuous elements. Choosing the  $L_2$  error as a measure of the error of the approximation, we consider the problem of constructing geometry(or solution)-adaptive node distribution in the parameter space of the curve to reduce the error. A key idea to the determination of the node distribution derives from the fact that the  $L_2$  error does not vanish even with the exact nodal values. This simple fact implies that there exist error components that

---

\*Department of Aerospace Engineering, University of Michigan, Ann Arbor, Michigan 48109. (hiroakin@engin.umich.edu).

are essentially independent of the nodal approximation. Hence no numerical schemes, which play a role of computing the nodal values, can control these errors, thus further implying that they can be controlled only by changing the node distribution in the parameter space. In other words, the development of algorithms for adaptive node distribution should be guided by these error components. As a consequence, such algorithms, unlike those which compute the nodal values, should be equally applicable to all the areas of application whose objective is to construct an accurate piecewise linear continuous approximation of a one-dimensional curve.

The error components that are incurable by numerical schemes can be derived by directly computing the  $L_2$  error for a smooth curve as shown in Section 2. It is shown that the leading term of these error components estimates the  $L_2$  error with second-order accuracy provided the nodal approximation is more than third order accurate. Therefore it plays an important role not only in guiding the node distribution but also in estimating the error. Exploiting this fact, an algorithm that constructs a piecewise linear continuous approximation for is devised in Section 3 based on an equidistribution strategy, that ensures a specified global  $L_2$  error or equivalently a specified local average error. The algorithm generates nodes successively, advancing from one end of the curve to the other rather than adjusting an existing nodes, which makes the computation very quick. A difficulty arises by a singular behaviour of the generating equation which is solved iteratively: a curvature term in the denominator causes a trouble when it is very small. A modification of the equation is proposed that damps a large number created by extremely small curvature.

Applications of the algorithm are described in Section 4. The first example is the approximation of a curve in  $\mathbf{R}^n$ . Results demonstrate that the method be capable of highly accurate approximations to known functions in one dimension that ensures a specified  $L_2$  error. The second is numerical integration in which case the upper bound of the integration error can be specified. The third is initial-value problems for ordinary differential equations in which the algorithm is combined with a fourth-order numerical scheme that computes nodal solutions. Finally, the method is applied to boundary-value problems for ordinary differential equations via the shooting technique, which requires a slight modification of the algorithm for initial-value problems.

## 2 Error estimates

In this section, we derive an  $L_2$  error estimate for a piecewise linear continuous approximation to a smooth one-dimensional curve. For this purpose, we begin by studying the simple case that the curve is merely a function in one dimension. The error will be calculated directly assuming that the curve we wish to approximate is smooth. From this result, the leading terms of the error components independent of nodal errors will be identified. The analysis will then be extended to one-dimensional curves in higher dimensions to derive a local and global  $L_2$  error estimates. Some numerical results are shown that verify the accuracy of the error estimates.

### 2.1 Local error analysis

Suppose we have a piecewise linear continuous approximation of unknown accuracy,  $u(t)$ , to a function  $x(t)$  in the interval  $I = [0, 1]$  which is subdivided into a set of elements  $\{E\}$  (See Figure 1). Within each element  $E = [t_l, t_r] \in \{E\}$ , we have

$$u_E(t) = \frac{\Delta u_E}{\Delta t_E} t + \frac{u_l t_r - u_r t_l}{\Delta t_E} \quad (1)$$

where  $\Delta u_E = u_r - u_l$ ,  $\Delta t_E = t_r - t_l$ , and  $u_l$  and  $u_r$  are some approximations to  $x(t_l)$  and  $x(t_r)$  respectively. Assume that  $x(t)$  is smooth within the element  $E$  and therefore expressible as a Taylor series around the midpoint of  $E$ ,

$$x(t) = x_m + \sum_{n=1}^{\infty} \frac{x_m^{(n)}}{n!} (t - t_m)^n \quad (2)$$

where  $x_m = x(t_m = (t_l + t_r)/2)$ , and  $x_m^{(n)}$  is the  $n$ th derivative of  $x(t)$  evaluated at  $t_m$ . Assuming  $\Delta t_E$  is small, we are going to compute the square of the local  $L_2$  error,

$$\|x - u_E\|_{L_2(E)}^2 = \int_E (x(t) - u_E(t))^2 dt \quad (3)$$

using (1) and (2) up to some significant order. Hence we begin with

$$\|x - u_E\|_{L_2(E)}^2 = \int_E \left( x_m - u_E(t) + \sum_{n=1}^{\infty} \frac{x_m^{(n)}}{n!} (t - t_m)^n \right)^2 dt. \quad (4)$$

Using the following relation, which is derived by approximating  $x_m$  by  $\frac{x(t_l) + x(t_r)}{2}$ ,

$$x_m - u_E(t) = \bar{e}_E - \frac{\Delta u_E}{\Delta t_E} (t - t_m) - \sum_{n=2,4,6} \frac{x_m^{(n)}}{2^n \cdot n!} \Delta t_E^n + \mathcal{O}(\Delta t_E^8) \quad (5)$$

where  $\bar{e}_E = \frac{1}{2}(e_l + e_r) = \frac{1}{2} \{ (x(t_l) - u_l) + (x(t_r) - u_r) \}$ , and neglecting high order terms, we obtain

$$\begin{aligned} \|x - u_E\|_{L_2(E)}^2 &= \int_E \left( \bar{e}_E - \Psi_E(t - t_m) - \sum_{n=2,4,6} \frac{x_m^{(n)}}{2^n \cdot n!} \Delta t_E^n + \sum_{n=2}^6 \frac{x_m^{(n)}}{n!} (t - t_m)^n \right)^2 dt \\ &\quad + \mathcal{O}(\Delta t_E^9) \end{aligned} \quad (6)$$

where  $\Psi_E = \frac{\Delta u_E}{\Delta t_E} - x_m^{(1)}$ . Note that the integral of  $(t - t_m)^n$  vanishes when  $n$  is odd, and results in  $\frac{\Delta t_E^{(n+1)}}{2^n(n+1)}$  when  $n$  is even. Consequently, the resulting expression will contain only odd order terms. The result is

$$\begin{aligned} \|x - u_E\|_{L_2(E)}^2 &= \bar{e}_E^2 \Delta t_E - \frac{1}{12} \left[ 2\bar{e}_E x_m^{(2)} - \Psi_E^2 \right] \Delta t_E^3 \\ &\quad - \frac{1}{240} \left[ \bar{e}_E x_m^{(4)} + \Psi_E x_m^{(3)} - 2 \left\{ x_m^{(2)} \right\}^2 \right] \Delta t_E^5 \\ &\quad - \frac{1}{26880} \left[ \bar{e}_E x_m^{(6)} + \Psi_E x_m^{(5)} - \frac{20}{3} x_m^{(2)} x_m^{(4)} - \frac{5}{3} \left\{ x_m^{(3)} \right\}^2 \right] \Delta t_E^7 \\ &\quad + \mathcal{O}(\Delta t_E^9). \end{aligned} \quad (7)$$

Although looks a little complicated, this contains valuable pieces of information about the error.

## 2.2 Leading terms

The simplest case to begin with would be a piecewise linear continuous approximation of a function  $x(t)$  with the exact nodal values. In this case (7) simplifies to

$$\|x - u_E\|_{L_2(E)}^2 = \frac{\left\{ x_m^{(2)} \right\}^2}{120} \Delta t_E^5 + \frac{\left\{ x_m^{(3)} \right\}^2}{30240} \Delta t_E^7 + \frac{x_m^{(2)} x_m^{(4)}}{4032} \Delta t_E^7 + \mathcal{O}(\Delta t_E^9) \quad (8)$$

where we have used the fact that

$$\Psi_E = \frac{1}{2} x_m^{(3)} \Delta t_E^2 + \mathcal{O}(\Delta t_E^4) \quad (9)$$

when the nodal values are exact. This shows that the second derivative of  $x(t)$  is the main contribution to the error, which clearly agrees to our intuition. We notice also that the leading term estimates the

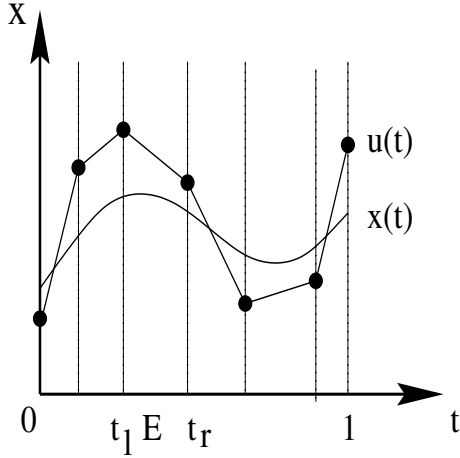


Figure 1: A piecewise linear approximation to a function in one dimension.

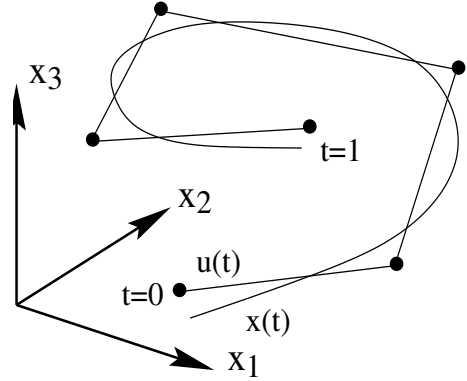


Figure 2: A piecewise linear approximation to a curve in three dimension.

error with second-order accuracy. It is, of course, exact for quadratic functions. It is then tempting to define it as a second-order error estimator neglecting the rest of the terms. If this is to be done, we can replace  $x_m^{(2)}$  by a second-order difference approximation, without altering the accuracy of the estimate. Therefore we can write

$$\|x - u_E\|_{L_2(E)}^2 = \frac{\Delta f_E^2 \Delta t_E^3}{120} + \mathcal{O}(\Delta t_E^7) \quad (10)$$

where  $f$  denotes the first derivative of  $x(t)$ . Now, we remark that the leading term fails to estimate the error accurately in the neighborhood of an inflection point where  $x_m^{(2)}$  would be extremely small no matter what difference approximation is employed. Clearly, in this case, the error must be estimated by a quantity which carries the information of  $x^{(3)}$ . Since the error terms of second-order difference approximations to  $x_m^{(2)}$  generally do not contain  $x^{(3)}$ , it follows that the largest term of such is only the second term in (8). However this means also that the error will be  $\mathcal{O}(\Delta t_E^2)$  smaller than that in the other region. Besides, since inflection points will exist usually at a finite number of discrete points, the failure might be tolerable if we are interested only in relatively large errors. But if that is not the case, we must retain the second term. Then, it may be convenient to replace  $x_m^{(3)}$  by  $\Psi_E$  by using (9). We thus obtain

$$\|x - u_E\|_{L_2(E)}^2 = \frac{\Delta f_E^2 + \frac{16}{7} \Psi_E^2}{120} \Delta t_E^3 + \mathcal{O}(\Delta t_E^7). \quad (11)$$

Note that we have included only a part of the next largest term which is relevant to  $x_m^{(3)}$ . Therefore the accuracy is improved near inflection points only, and so the overall accuracy of the estimate remains second order.

Next we consider the solution of ordinary differential equations where the nodal values are computed by a numerical scheme. In this case, we can deduce from (7) that

$$\|x - u_E\|_{L_2(E)}^2 = \frac{\Delta f_E^2 \Delta t_E^3}{120} + \mathcal{O}(\Delta t_E^6) \quad (12)$$

provided the nodal approximation is of  $\mathcal{O}(\Delta t_E^3)$ , and that we recover (10) with the nodal approximations higher than third order. Therefore we must use, at lowest, third-order methods to compute the nodal values in order for the leading term to estimate the error. This is possible in most cases, for example using the fourth-order Runge-Kutta method for initial-value problems or boundary-value problems via shooting technique. Then the leading term can be used to estimate  $\|x - u_E\|_{L_2(E)}^2$  with second-order accuracy, and (11) can also be used to improve it near inflection points. Yet, even if the leading term

is disqualified for the error estimator, it will still be a useful quantity. Suppose that a perfect scheme were devised which finds the exact nodal solution on a grid with a finite number of elements. Then, it is seen from (8) that even with such a scheme, the error, expressed by  $\|x - u_E\|_{L_2(E)}^2$ , cannot be driven to zero, and that it will be dominated by the leading term or the one in (12). Hence the leading term dominates the error components that are incurable by numerical schemes. And we see that it is possible to control this error component only through the adjustment of the grid. In other words, minimization of this quantity can be considered as a theme of the selection or the adjustment of the grid.

### 2.3 Error estimates

We now consider  $L_2$  error of piecewise linear approximations to one-dimensional curves in  $\mathbf{R}^n$ . The extension of the previous analysis to one-dimensional curves in higher dimensions is in fact straightforward. Suppose we have a piecewise linear continuous approximation in  $\mathbf{R}^n$  (See Figure 2)

$$\mathbf{u} = (u_1, \dots, u_n) = (u_1(t), \dots, u_n(t)) \quad (13)$$

where  $t \in I = [0, 1]$  which is again divided into a set of elements  $\{E\}$ , to a smooth one-dimensional curve parametrized by

$$\mathbf{x} = (x_1, \dots, x_n) = (x_1(t), \dots, x_n(t)). \quad (14)$$

Here  $n$  denotes the dimension of the ambient space, and thus  $n = 1$  corresponds to functions in one dimension. Within each element, we have a local linear function  $u_{iE}$  similar to (1) for each coordinate. In  $n$  dimensions, it would be natural to define the local  $L_2$  error by

$$\|\mathbf{x} - \mathbf{u}\|_{L_2(E)}^2 = \int_E \sum_{i=1}^n (x_i(t) - u_{iE}(t))^2 dt. \quad (15)$$

Geometrically, the quantity thus defined is the measure of the square of the area of the surface formed by joining the two curves by line segments at points with the same  $t$ . Then the previous analysis applies to each  $i$ , and introducing the notation  $f_i = dx_i/dt$ , we have the result

$$\|\mathbf{x} - \mathbf{u}\|_{L_2(E)}^2 = \frac{\{\sum_{i=1}^n \Delta f_{iE}^2\} \Delta t_E^3}{120} + \mathcal{O}(\Delta t_E^7) \quad (16)$$

for a smooth one-dimensional curves with the exact or fourth-order accurate nodal values. Note that the leading term approximates

$$\frac{\left| \frac{d^2 \mathbf{x}}{dt^2} \right|^2 \Delta t_E^5}{120}. \quad (17)$$

Hence it contains the magnitude of the acceleration vector,

$$\frac{d^2 \mathbf{x}}{dt^2} = \left( \frac{ds}{dt} \right)^2 \frac{d^2 \mathbf{x}}{ds^2} + \frac{d^2 s}{dt^2} \frac{d\mathbf{x}}{ds} \quad (18)$$

where  $s$  denotes the arc length. This means that the leading term contains the information not only about the curvature  $\frac{d^2 \mathbf{x}}{ds^2}$  but also about the acceleration along the curve  $\frac{d^2 s}{dt^2}$ . In particular, if the curve is parametrized by its arc length, it becomes precisely the measure of the magnitude of the curvature.

Now, taking the square root of (16) and expanding, we obtain the following local  $L_2$  error estimate

$$\|\mathbf{x} - \mathbf{u}\|_{L_2(E)} = \mathcal{E}_{2(E)} [1 + \mathcal{O}(\Delta t_E^2)] \quad (19)$$

where

$$\mathcal{E}_{2(E)} = \frac{C_E \sqrt{\Delta t_E}}{\sqrt{120}}, \quad C_E = \Delta t_E \sqrt{\sum_{i=1}^n \Delta f_{iE}^2}. \quad (20)$$

Note that  $C_E$  is related to the local average error,

$$\sqrt{\frac{1}{\Delta t_E} \int_E \sum_{i=1}^n (x_i(t) - u_{iE}(t))^2 dt} = \frac{C_E}{\sqrt{120}} [1 + \mathcal{O}(\Delta t_E^2)]. \quad (21)$$

In a similar manner, we can estimate the global  $L_2$  error defined by

$$\|\mathbf{x} - \mathbf{u}\|_{L_2(I)} = \sqrt{\int_I \sum_{i=1}^n (x_i(t) - u_i(t))^2 dt}. \quad (22)$$

Clearly we have

$$\|\mathbf{x} - \mathbf{u}\|_{L_2(I)}^2 = \sum_{\{E\}} \|\mathbf{x} - \mathbf{u}\|_{L_2}^2. \quad (23)$$

Then, inserting (16), taking the square root and expanding, we obtain

$$\|\mathbf{x} - \mathbf{u}\|_{L_2(I)} = \mathcal{E}_{2(I)} [1 + \mathcal{O}(\Delta t_E^2)] \quad (24)$$

where

$$\mathcal{E}_{2(I)} = \sqrt{\sum_{\{E\}} \mathcal{E}_{2(E)}^2} = \sqrt{\sum_{\{E\}} \frac{C_E^2 \Delta t_E}{120}} \quad (25)$$

The accuracy of  $\mathcal{E}_{2(E)}$  and  $\mathcal{E}_{2(I)}$  as error estimators, as mentioned earlier, will be second-order with the nodal approximation better than third-order, and first-order with third-order nodal approximations, except in the neighborhood of inflection points. To improve it near inflection points as discussed earlier, it can be shown, extending (11), that we need to modify  $C_E$  as follows.

$$C_E = \Delta t_E \sqrt{\sum_{i=1}^n \left\{ \Delta f_{iE}^2 + \frac{16}{7} \Psi_{iE}^2 \right\}} \quad (26)$$

where  $\Psi_{iE} = \frac{\Delta u_{iE}}{\Delta t_E} - x_{im}^{(1)}$ . Although the error estimators thus defined require the knowledge of the first derivatives at nodes, they do not have to be exact. The estimates will be second order accurate as long as the nodal values approximating the first derivatives  $f_i$  are second order accurate at least. If the parametrization  $x(t)$  is available, we may apply a finite-difference approximation to compute  $f(t)$  using a suitable choice of a small interval or apply the second order central difference formula directly to the second derivatives. In the case of initial-value problems for ordinary differential equations,  $f$  is always available as the right hand side of the equations. We emphasize again that it is also important to remember that the quantity  $\mathcal{E}_{2(E)}$  (or  $C_E$ ) will have a role as a guide to a good node distribution even if it is disqualified for an error estimator.

## 2.4 Numerical tests for accuracy

This section gives the results that verify the accuracy of the error estimate. First we consider the case where the nodal values are exact, i.e. linear interpolation of a function. Then we consider the solution of an initial-value problem for ordinary differential equations. In the latter, we discuss how numerical schemes affect the accuracy of the error estimator.

We consider approximating the following function

$$x(t) = e^{-3t} \sin(4\pi t) \quad (27)$$

by a piecewise linear function on a uniform grid with  $N_E$  elements. The function and the approximation with  $N_E = 5$  are shown in Figure 3. The computed errors are given in Table 1, where the actual  $L_2$

$N_E$	$\ x - u\ _{L_2(I)}$	$\mathcal{E}_{2(I)}$	Relative Error
5	1.947E-01	1.722E-01	1.152E-01
10	4.657E-02	4.513E-02	3.100E-02
20	1.173E-02	1.164E-02	7.740E-03
40	2.941E-03	2.935E-03	1.933E-03
80	7.359E-04	7.356E-04	4.833E-04

Table 1: The results for (27)

$N_E$	$\ x - u\ _{L_2(I)}$	$\mathcal{E}_{2(I)}$	Relative Error
20	1.07705E-02	1.43262E-02	3.30132E-01
40	3.13290E-03	3.69313E-03	1.78821E-01
80	8.65301E-04	9.30703E-04	7.55830E-02
160	2.25419E-04	2.33147E-04	3.42850E-02
320	5.73809E-05	5.83164E-05	1.63025E-02

Table 2: 3rd order Runge-Kutta Method.

$N_E$	$\ x - u\ _{L_2(I)}$	$\mathcal{E}_{2(I)}$	Relative Error
20	1.42137E-02	1.43262E-02	7.92006E-03
40	3.68583E-03	3.69313E-03	1.98266E-03
80	9.30242E-04	9.30703E-04	4.95826E-04
160	2.33118E-04	2.33147E-04	1.23966E-04
320	5.83146E-05	5.83164E-05	3.09921E-05

Table 3: 4th order Runge-Kutta Method.

errors were computed by using, within each element, the five-point Gaussian quadrature formula which is exact for 9th order polynomials and therefore is sufficiently accurate for our comparison purpose. Note that all the computations in this paper were performed with double precision, but the results have been truncated for brevity. The table shows a very good agreement between the actual  $L_2$  error and  $\mathcal{E}_{2(I)}$  for which we used  $C_E$  defined by (20) and  $f(t)$  is computed by using the central difference formula with a step size 1.0E-05 which was found to give the identical result as that obtained by using the exact derivative of (27). In the last column of the table, the relative errors, defined by  $\left|1 - \frac{\mathcal{E}_{2(I)}}{\|x-u\|_{L_2(I)}}\right|$ , are shown. Starting with 1.15% error for  $N_E = 5$ , the relative error goes down as the grid is refined. The rate of convergence is found to be 1.98 which verifies the result in the last section. As mentioned earlier, the local  $L_2$  error cannot be predicted accurately near inflection points. To see this, we computed the relative error locally for each element with  $N_E = 160$ . The plots are shown in Figure 4. Note that the ordinates of the lower three plots are logarithmically scaled. On the top, the first derivative  $f(t)$  is plotted so that the inflection points can be easily located, and the upper-middle one is the plot of the elementwise relative error  $\left|1 - \frac{\mathcal{E}_{2(E)}}{\|x-u\|_{L_2(E)}}\right|$  plotted in the middle of each element. It is seen that the relative error is significantly larger near inflection points than in the other region. However, as seen in the plot in the lower-middle, the actual local  $L_2$  errors  $\|x - u\|_{L_2(E)}$  are significantly smaller near inflection points than in the other region. The relative error obtained with the modification (26) is in the plot on the bottom. It is seen that the errors have been reduced nearly by an order of magnitude at inflection points.

Next we consider numerical solutions of the following linear initial-value problem.

$$\frac{dx(t)}{dt} = -c_s x + c_s \sin(at) + \cos(at) \quad t \in [0, 1] \quad (28)$$

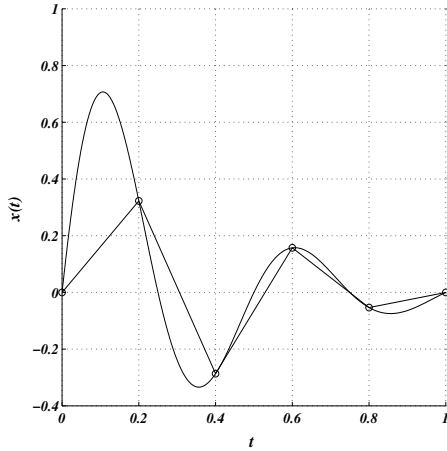


Figure 3: Function (27) and a piecewise linear representation on a uniform grid with  $N_E = 5$ .

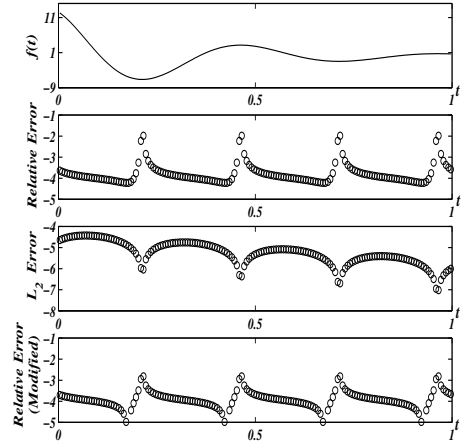


Figure 4: Top: The first derivative of (27). Upper-Middle: Elementwise relative error. Lower-Middle: Actual local  $L_2$  errors. Bottom: Elementwise relative error with the modification (26). The last three are plotted in the middle of each element with  $N_E = 160$ .

with  $c_s = 20$  and  $a = 6$ . The exact solution is given by

$$x(t) = e^{-c_s t} + \sin(at) \quad (29)$$

which is shown in Figure 5 with a numerical solution. The equation was integrated by a third-order explicit Runge-Kutta method[6] and the classical 4th order Runge-Kutta method. Table 2 shows the result for the third-order method. The actual  $L_2$  errors were computed in the same way as in the previous section, and  $\mathcal{L}_2$  was computed without the modification (26). We see that  $\mathcal{E}_2$  estimates the actual error fairly well in the sense that their first digits agree except for  $N = 80$ . The rate of convergence of the relative error is 1.11 as we expected. For the fourth-order method, it is seen in Table 3 that the errors are estimated more accurately, for example the relative error is 8% for  $N_E = 20$  while it is 33% for the third-order method. The rate of convergence of the relative error is 2.00 as we expected. We also applied the modified Euler's method which is only second-order accurate. Including this, Figure 6 shows the plot of the convergence histories. As we expected, using a second-order method deteriorates the accuracy of the error estimate, and the relative error remains large for any grids.

### 3 Algorithm

In this section, we give a basic idea of the algorithm that generates a node distribution to produce a smooth piecewise linear approximation. It is based on an equidistribution property which is discussed below. The algorithm is described for a simple example.

#### 3.1 Equidistribution

It is desired that the node distribution is determined such that the quantity  $\mathcal{E}_{2(E)}$  is minimized. But since a direct minimization seems rather difficult, we propose to place the nodes so as to equally distribute  $C_E$  over the elements, thus concentrating the nodes in the region of large  $\left| \frac{d^2 \mathbf{x}}{dt^2} \right|$  or equivalently equidistributing

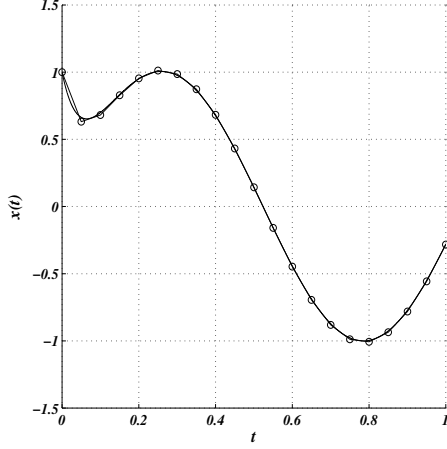


Figure 5: The exact solution (29) with  $c_s = 1.0$ , and the numerical solution by a third-order Runge-Kutta Method with  $N_E = 20$ .

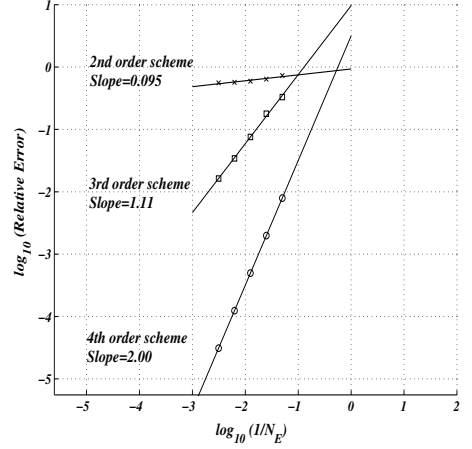


Figure 6:  $-\log_{10}(N_E)$  vs.  $\log_{10}(\text{Relative Error})$  and the linear least-squares fits.

the local average error. We advocate this strategy also because of the following useful property. Suppose that we generated nodes such that  $C_E$  is equidistributed. Then we find

$$\mathcal{E}_{(I)}^2 = \sum_{\{E\}} \frac{C_E^2 \Delta t_E}{120} = \sum_{\{E\}} \frac{C^2 \Delta t_E}{120} = \frac{C^2}{120} \quad (30)$$

where  $C = C_E$  for all  $E \in \{E\}$ . This shows that the global  $L_2$  error becomes identical to the local average error on equidistribution grids. Also, more importantly, we notice that it provides us with a means to determine  $C$  for a desired  $L_2$  or the local average error. This will be useful when we seek an approximation that gives a desired error. Thus this leads us to consider an algorithm that generates nodes such that  $C_E = C$  for every element to be generated. Note that the equidistribution implies a uniform spacing for quadratic functions whose curvature is constant.

At this point, it is not clear yet if the grid thus generated gives a better approximation than that with the uniform grid of the same size. To investigate this, we ask how the equidistribution affects the first variation of  $\mathcal{E}_{(I)}^2$ . For this purpose, without loss of generality, we may consider the case  $n = 1$  with the exact nodal values. Suppose that we have successfully equidistributed  $C_E = \Delta t_E |\Delta f_E|$  over  $\{E\}$ . We give a small perturbation  $dt$  to a node at  $t = t_j$ . The first variation of  $\mathcal{E}_{(I)}^2$  is then given by

$$\begin{aligned} d\mathcal{E}_{(I)}^2 = & \frac{1}{40} [\Delta t_L^2 \Delta f_L^2 - \Delta t_R^2 \Delta f_R^2] dt \\ & + \frac{1}{60} [\Delta t_L^3 \Delta f_L - \Delta t_R^3 \Delta f_R] f^{(1)}(t_j) dt \end{aligned} \quad (31)$$

where  $f^{(1)} = df/dt$ , and the subscripts, L and R, denote the elements that share the node  $j$ . Note that the both sides are  $\mathcal{O}(\Delta t^4)$ , meaning that  $\mathcal{E}_{(I)}^2$  and the gradient go down at the same rate. The first term vanishes by the assumption, and we have

$$\frac{\partial \mathcal{E}_{(I)}^2}{\partial t_j} = \frac{1}{60} [\Delta t_L^3 \Delta f_L - \Delta t_R^3 \Delta f_R] f^{(1)}(t_j). \quad (32)$$

Note that this shows that we achieve a minimum for a quadratic function because the equidistribution implies a uniform grid, and that it is a minimum of the true  $L_2$  error because  $\mathcal{E}_{(I)}^2$  becomes identical

to the  $L_2$  error for quadratic functions. Yet another interesting observation is that if  $\Delta f_R \Delta f_L < 0$ , i.e. nonconvex, the only way to achieve a minimum is to place  $t_j$  at the inflection point of  $x(t)$  where  $f^{(1)}$  vanishes. Next we assume that  $\Delta f_R \Delta f_L > 0$  and that  $\Delta t_L \neq \Delta t_R$ , and write, using the assumption of the equidistribution,

$$\frac{\partial \mathcal{E}_{(I)}^2}{\partial t_j} = \frac{C}{60} [\Delta t_L^2 - \Delta t_R^2] f^{(1)}(t_j) \quad (33)$$

where  $C = \Delta t_R \Delta f_R = \Delta t_L \Delta f_L$ . It is clear that we cannot have a minimum in this case. However, if we replace  $f^{(1)}(t_j)$  by a first-order finite difference approximation in each element, we obtain

$$\begin{aligned} \frac{\partial \mathcal{E}_{(I)}^2}{\partial t_j} = & \frac{C}{60} \left[ \Delta t_L^2 \left\{ \frac{\Delta f_L}{\Delta t_L} + \frac{f^{(2)}(t_j)}{2} \Delta t_L + \mathcal{O}(\Delta t_L^2) \right\} \right. \\ & \left. - \Delta t_R^2 \left\{ \frac{\Delta f_R}{\Delta t_R} - \frac{f^{(2)}(t_j)}{2} \Delta t_R + \mathcal{O}(\Delta t_R^2) \right\} \right]. \end{aligned} \quad (34)$$

By the assumption and also noting that  $C$  is  $\mathcal{O}(\Delta t^2)$ , we finally get

$$\frac{\partial \mathcal{E}_{(I)}^2}{\partial t_j} = \frac{C}{120} [\Delta t_L^3 + \Delta t_R^3] f^{(2)}(t_j) + \mathcal{O}(\Delta t^6). \quad (35)$$

Hence the right hand side is now  $\mathcal{O}(\Delta t^5)$ , implying that a smaller gradient can be achieved by equidistribution and that the gradient goes down faster than  $\mathcal{E}_{(I)}^2$  does, implying that the error is closer to a minimum. Therefore, we conclude that the equidistribution grid give, if not minimum, a smaller error than the uniform grid of the same size.

## 3.2 Algorithm

Given a function  $x(t)$  in  $I$  together with its first derivative, we consider constructing a piecewise linear continuous approximation for a given error  $C$  where  $C$  is a desired local average error multiplied by  $\sqrt{120}$ . The nodal values are assigned simply by  $x(t)$ , and therefore the error estimates are second order accurate. Now the idea is to generate nodes successively starting from the initial point  $(t_0, x(t_0)) = (0, x(0))$  such that  $C_E = C$  over all the elements to be constructed. To generate the next node  $t_{j+1}$  from  $t_j$  which together define the element  $E \in \{E\}$ , we need to iterate for  $t_{j+1}$  until we have

$$(t_{j+1} - t_j) |\Delta f_E| - C = 0. \quad (36)$$

where  $\Delta f_E = f(t_{j+1}) - f(t_j)$ . A good iteration formula is

$$t_{j+1}^{(k+1)} = t_j + \left[ \frac{C(t_{j+1}^{(k)} - t_j)^{p-1}}{|\Delta f_E^{(k)}|} \right]^{\frac{1}{p}} \quad (37)$$

where the superscript  $k(\geq 1)$  indicates the number of iterations,  $p$  is a positive real number. It is easy to show that (36) is satisfied at convergence. The role of  $p$  is to damp an excessively large change, i.e. an extremely small  $\Delta f_E^k$  which will occur when the iteration enters a region that is near an inflection point or where the curve is locally linear. Also the modification (26) will be effective to prevent the zero division. But if the curve is truly linear, i.e.  $|\Delta f_E^k|$  as well as  $\Psi_E$  are identically zero, any choice of  $p$  or the modification (26) will not work. In this case, a practical tip is to add a small constant in the denominator, thus adding a small curvature  $\delta$ . Yet, a more sophisticated way to add the artificial curvature is to add the term in the following form.

$$\delta = \frac{C}{\tilde{h}} \exp\left(-\lambda |\Delta f_E^{(k)}|\right) \quad (38)$$

where  $\tilde{h}$  is the desired spacing in the region where the curve is linear and  $\lambda$  is a constant that controls the effect of the artificial curvature. This will, of course, destroy the error equidistribution property, but will not increase the error because its effect is to increase the nodes.

### 3.3 Properties of the iteration formula

For simplicity, we assume that the function in consideration is quadratic, which would be approximately valid locally in a small subdomain of nonquadratic functions unless the curvature is zero. We write (37) as

$$\Delta t^{(k+1)} = Q^{1/p} (\Delta t^{(k)})^{1-\frac{2}{p}} \quad (39)$$

where

$$\Delta t^{(k+1)} = t_{j+1}^{(k+1)} - t_j, \quad Q = \frac{C}{|\Delta f_E^{(k)}|/\Delta t^{(k)}}. \quad (40)$$

For quadratic functions,  $\Delta f/\Delta t$  represents the constant curvature exactly, and therefore  $Q$  is constant. Observe that for the choice  $p = 2$ , (39) becomes

$$\Delta t^{(k+1)} = \sqrt{Q}. \quad (41)$$

That is to say, the iteration is independent of  $k$  as well as the initial guess, and will converge at the first try for any initial values, producing uniform node distribution in  $t$ . Of course, we then achieve the equidistribution.

$$C_E^{(k+1)} = \Delta t^{(k+1)} |\Delta f_E^{(k+1)}| = \left\{ \Delta t^{(k+1)} \right\}^2 \frac{C}{Q} = C. \quad (42)$$

In general, we have, for quadratic functions,

$$\Delta t^{(k+1)} = \sqrt{Q} \left[ \frac{\Delta t^{(1)}}{\sqrt{Q}} \right]^{(1-\frac{2}{p})^k}. \quad (43)$$

It can be easily proved by induction that this satisfies (39). In terms of  $C_E$ , we have

$$C_E^{(k+1)} = \Delta t^{(k+1)} |\Delta f_E^{(k+1)}| = C \left[ \frac{\{\Delta t^{(1)}\}^2}{Q} \right]^{(1-\frac{2}{p})^k}. \quad (44)$$

It is straightforward to show that for  $p > 1$  we have

$$\left| 1 - \frac{2}{p} \right| < 1 \quad (45)$$

and  $C_E^{(k+1)}$  will converge to the limit  $C$ , and that for  $p \geq 1$  we find

$$1 - \frac{2}{p} \leq -1, \quad (46)$$

and hence it will not converge. Therefore, the value of  $p$  must be greater than 1 for the iteration to converge. It is also important to note that it takes longer to converge with a larger  $p$  since  $\left| 1 - \frac{2}{p} \right| \rightarrow 1$  as  $p \rightarrow \infty$ . For nonquadratic functions with nonvanishing curvature, we therefore conclude that  $p = 2$  give the fastest convergence which takes only one iteration provided the spacing  $\Delta t$  that achieves  $C_E = C$  is so small that the curvature is locally constant. For functions with inflection points, however, a large value must be assigned to  $p$  to make the iteration proceed as mentioned in the last subsection although this will make the iteration take longer in convex part of the function. Note also that the iteration will converge faster for smaller  $C$ , i.e. smaller  $\Delta t$ , since the function will behave more and more quadratically in the neighborhood of such a small subdomain.

## 4 Applications

### 4.1 Approximation of function and curves

In this section, we describe an algorithm to construct an accurate piecewise linear continuous approximation to a given curve in  $\mathbf{R}^n$ , with the equidistribution property. The curve is assumed to be parametrized analytically or by some approximation such as cubic splines. The main objective is to determine the distribution of the nodes in the parameter space such that the equidistribution is achieved. The results are however shown for the case  $n = 1$  only.

#### 4.1.1 Algorithm

Given an  $L_2$  error, we compute  $C$  such that  $\mathcal{E}_2$  is equal to the specified error, and generate nodes successively starting from an initial point  $(t_0, x_1(t_0), \dots, x_n(t_0)) = (0, x_1(0), \dots, x_n(0))$  such that  $C_E = C$  for all the elements to be generated. The iteration formula for  $t_{j+1}$ , a generalization of (37) to  $n$  dimensions, is

$$t_{j+1}^{k+1} = t_j + \left[ \frac{C}{C_E} \right]^{\frac{1}{p}} (t_{j+1}^k - t_j). \quad (47)$$

where  $k$  indicates the number of iterations,  $p$  is a positive real number and

$$C_E = (t_{j+1}^k - t_j) \left\{ \sqrt{\sum_{i=1}^n \left\{ \Delta f_{iE}^2 + \frac{16}{7} \Psi_{iE}^2 \right\}} + \delta \right\} \quad (48)$$

The second term in the square root may be removed for strictly convex functions. If it is desired to have a uniform grid spacing  $\tilde{h}$  in the region where the curve is linear, we may set

$$\delta = \frac{C}{\tilde{h}} \exp \left( -\lambda \sqrt{\sum_{i=1}^n \Delta f_{iE}^2 + \frac{16}{7} \Psi_{iE}^2} \right) \quad (49)$$

where  $\lambda$  is a user-specified constant that limits the influence of this artificial curvature in the region where such correction is not desired.. The first derivative  $f'(t)$  will be computed by a simple central difference formula,

$$f'(t_j) = \frac{x(t_j + h) - x(t_j - h)}{2h} \quad (50)$$

with  $h = 1.0\text{E-}5$ . The initial guess is always set to be  $t_j + (t_j - t_{j-1})$  assuming the curve is locally quadratic for  $j > 0$ , and to be 0.001 as a sheer guess for  $j = 0$ . The iteration is taken to be converged when

$$\left| \frac{C_E}{C} - 1 \right| < \text{TOL} \quad (51)$$

where TOL is set to be 1.0E-03, thus allowing 0.1 % error. The process will be terminated when a new node exceeds  $t = 1$ . The node distribution in  $I$  will be determined independently, and therefore the nodal values can be assigned by the given parametrization at the end of the node generation process. The algorithm is summarized as follows.

1. compute  $C$  by  $C = \sqrt{120} \mathcal{E}_{2(I)}$  for a desired error  $\mathcal{E}_{2(I)}$ ;
2. set  $t_{j+1} = t_j + (t_j - t_{j-1})$  ( $t_{j+1} = 0.001$  for  $j = 0$ );
3. compute a new location  $t_{j+1}$  by (47);
4. if (51) is not satisfied, go to 3;

$\mathcal{E}_{2(I)}$	N	noi	$\ x - u\ _{L_2(I)}$	$L_\infty$	$L_{2U}$
1.0E-02	6	7.4	4.711E-03	9.932E-03	6.053E-02
1.0E-04	40	2.5	6.069E-05	9.999E-05	2.073E-03
1.0E-06	384	1.2	7.409E-07	1.000E-06	2.199E-05

Table 4: Example (a):  $p = 2$ .

$\mathcal{E}_{2(I)}$	N	noi	$\ x - u\ _{L_2(I)}$	$L_\infty$	$L_{2U}$
1.0E-02	15	5.2	1.018E-02	1.260E-02	1.144E-01
1.0E-04	135	2.8	9.979E-05	1.086E-04	3.723E-03
1.0E-06	1337	1.7	1.000E-06	1.048E-06	3.863E-05

Table 5: Example (b):  $p = 3$ .

5. if  $t_{j+1} < 1$ , go to 2 (Next node).
6. choose the node at  $t = 1$ : end of the node-generation process.
7. assign  $x(t)$  at all the nodes generated.

In the results that follow, the endpoint has been chosen as follows. The last node generated, the one that exceeds  $t = 1$ , will be moved to  $t = 1$  if the size of the element defined by  $t = 1$  and the previous node is greater than 20 % of its previous element size. Otherwise, the previous node will be moved to  $t = 1$ .

#### 4.1.2 Results

We consider the case  $n = 1$ , i.e. approximation of functions in one dimension. Numerical tests were performed for the following four different functions.

- (a)  $at + (1 - a) \{1 - e^{(-t/\epsilon)}\} / \{1 - e^{(-1/\epsilon)}\}$  with  $\epsilon = 0.04$  and  $a = 0.6$
- (b)  $a(t - 0.5)^2 - \frac{a}{4}(1 + 8\epsilon t) + (1 + 2\epsilon a) \{1 - e^{(-t/\epsilon)}\} / \{1 - e^{(-1/\epsilon)}\}$  with  $\epsilon = 0.01$  and  $a = 3.5$
- (c)  $\tanh\{20(t - 0.5)\}$
- (d)  $10e^{(-10t)} + 20/\{1 + 400(t - 0.7)^2\}$

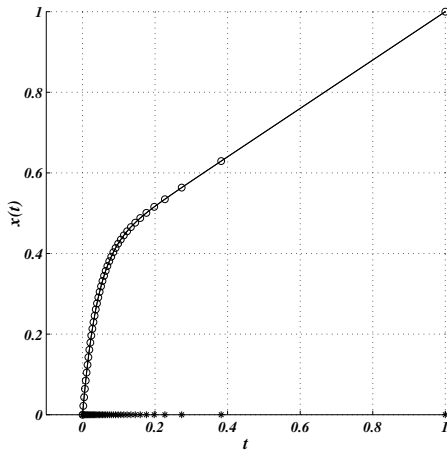
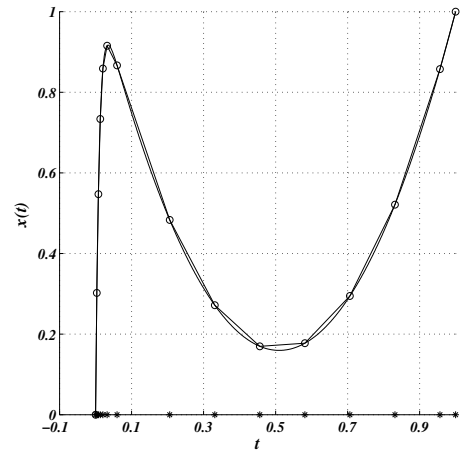
(a) is a strictly convex function, and (b) has a single inflection point and also exhibits a large variation near  $t = 0$ . (c) is symmetric with a single inflection point in the middle, and (d) has two inflection points. Examples (c) and (d) are taken from [2]. We set  $\delta = 0$  for all the cases.

The results are summarized in Table 4 to 7 where  $\mathcal{E}_{2(I)}$  is the specified  $L_2$  error,  $N$  is the total number of nodes,  $noi$  is the arithmetic average of the numbers of iterations to find the next node,  $\|x - u\|_{L_2(I)}$  is the actual  $L_2$  error computed by the five point Gaussian quadrature formula,  $L_\infty$  is the maximum of the local average error, i.e.  $L_\infty(\sqrt{\|x - u\|_{L_2(I)}/\Delta t_E})$ , and finally  $L_{2U}$  is  $L_2$  error on a uniform grid of the same size. The approximations constructed are shown in Figures 7 to 10. The results confirm that the specified  $L_2$  errors are accurately achieved except for Examples (a) and (b) where the actual  $L_2$  errors are smaller than expected. This happens because of the violation of the equidistribution in the last element. Since the functions are almost linear near  $t = 1$  in both cases, the placement of the last node is very likely to give a smaller error for the last element. Indeed, we found that if we excluded the last element when computing the error, the actual  $L_2$  errors were very close to the specified values.  $L_\infty$  norm of the local average errors are compared with  $\mathcal{E}_{2(I)}$ . A good agreement can be seen for every case. Although the only  $L_\infty$  norm is given here, the values do not vary significantly over the elements, again except for the last element, due to the equidistribution.  $L_2$  errors on the uniform grid of the same sizes are given

$\mathcal{E}_{2(I)}$	N	noi	$\ x - u\ _{L_2(I)}$	$L_\infty$	$L_{2U}$
1.0E-02	12	33.4	5.870E-03	1.042E-02	3.311E-02
1.0E-04	104	16.2	7.507E-05	1.001E-04	7.930E-04
1.0E-06	1026	8.7	8.899E-07	1.001E-06	8.026E-06

Table 6: Example (c):  $p = 8$ .

$\mathcal{E}_{2(I)}$	N	noi	$\ x - u\ _{L_2(I)}$	$L_\infty$	$L_{2U}$
1.0E-01	25	10.5	9.483E-02	1.084E-01	3.685E-01
1.0E-03	234	6.4	9.978E-04	1.033E-03	4.627E-03
1.0E-05	2324	3.4	9.993E-06	1.003E-05	4.660E-05

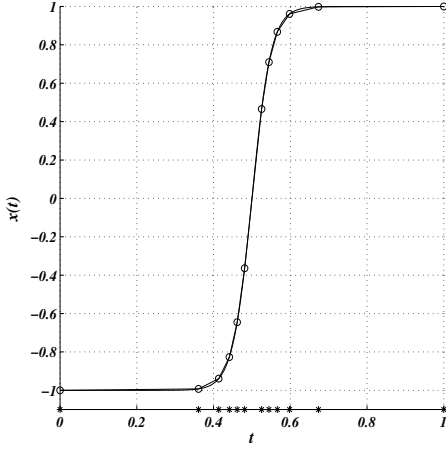
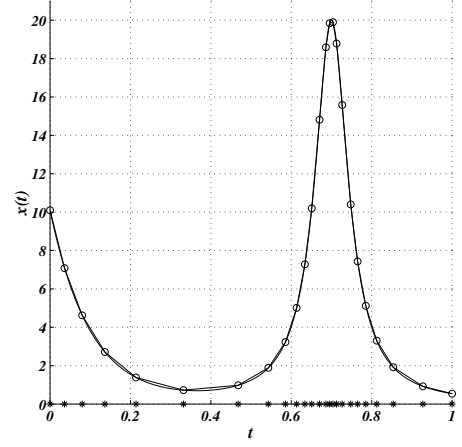
Table 7: Example (d):  $p = 4$ .Figure 7: Example (a).  $\mathcal{E}_{2(I)} = 1.0\text{E-}04$ .Figure 8: Example (b).  $\mathcal{E}_{2(I)} = 1.0\text{E-}02$ 

for comparison purposes. In all the cases, it is seen that extra orders of magnitude reduction has been achieved. The number of iterations to find the next node is usually a few in locally convex regions, and more in the neighborhood of inflection points. As seen in Tables, the number of iterations decreases as the number of nodes increases as we expected. Note also that the number of iterations is large for a large  $p$  which is consistent with our observation in the last section. We remark that the algorithm is so fast that it takes not more than two seconds to produce every result given here.

In some applications, it may be desired to capture the end point accurately. For instance, as shown in Figure 9, the function is symmetric with respect to  $x = 0.5$  but the grid is not quite symmetric, and one might wish to have a symmetric grid. One way to do this is to apply a shooting technique such as the one proposed by Ketzscher and Forth[5] in a related context.

## 4.2 Numerical integration

The algorithm described in the last section was shown to be capable of producing a highly accurate piecewise linear approximation, with a specified  $L_2$  error, to a known function. A byproduct is an accurate estimation of the integral of the function. But accurate prediction of the error cannot be made in this case, and only the upper bound of the error can be specified. The results show that the method gives errors lower than, but sufficiently close to, the specified upper bound for practical purposes.

Figure 9: Example (c).  $\mathcal{E}_{2(I)} = 1.0\text{E-}02$ Figure 10: Example (d).  $\mathcal{E}_{2(I)} = 1.0\text{E-}01$ 

#### 4.2.1 Algorithm

Consider the Cauchy-Schwarz inequality[4],

$$\left| \int_I g(t)h(t)dt \right| \leq \|g(t)\|_{L_2(I)} \|h(t)\|_{L_2(I)} \quad (52)$$

where  $g(t)$  and  $h(t)$  are continuous functions in  $I$ . Let  $g(t) = x(t) - u(t)$  and  $h(t) = 1$ , where  $x(t)$  is a known function we wish to integrate and  $u(t)$  is a piecewise linear approximation to  $x(t)$ , then we have

$$\left| \int_I x(t) - u(t) dt \right| \leq \|x(t) - u(t)\|_{L_2(I)} \quad (53)$$

which becomes, by (24),

$$\left| \int_I x(t) dt - \int_I u(t) dt \right| \leq \mathcal{E}_{2(I)} [1 + \mathcal{O}(\Delta t_E^2)]. \quad (54)$$

The integral of  $u(t)$ , that is easily computable by the trapezoidal rule, being our numerical approximation to the exact integral, the left hand side represents precisely the error of the integration. Then, the above inequality shows that the integration error is bounded by the  $L_2$  error estimate,  $\mathcal{E}_{2(I)}$ , with a possible second-order error. Therefore, with the algorithm that can construct  $u(t)$  for a given  $\mathcal{E}_{2(I)}$ , we can specify the upper bound of the integration error within the second-order truncation error. The resulting algorithm is an adaptive step-size control technique for the trapezoidal rule.

The algorithm is exactly the same as before except that the last node, i.e. the one that exceeds  $t = 1$ , is always pulled down to  $t = 1$  to avoid an increase in the upper bound  $\mathcal{E}_{2(I)}$ , and that the integration of the resulting piecewise linear function must be performed at the end of the process.

1. compute  $C$  by  $C = \sqrt{120} \mathcal{E}_{2(I)}$  for a desired error bound  $\mathcal{E}_{2(I)}$ ;
2. set  $t_{j+1} = t_j + (t_j - t_{j-1})$  ( $t_{j+1} = 0.001$  for  $j = 0$ );
3. compute a new location  $t_{j+1}$  by (47);
4. if (51) is not satisfied, go to 3;
5. if  $t_{j+1} < 1$ , go to 2 (Next node).
6. set  $t_{j+1} = 1$ : end of the node-generation process.
7. assign  $x(t)$  at all the nodes generated, and compute the integral  $\int_I u(t) dt$ .

The Integration Error	The Upper Bound( $\mathcal{E}_{2(I)}$ )	N
2.50E-03	1.0E-02	6
4.14E-05	1.0E-04	40
5.52E-07	1.0E-06	384

Table 8: Example (a):  $p = 2$ .

The Integration Error	The Upper Bound( $\mathcal{E}_{2(I)}$ )	N
8.07E-3	1.0E-02	15
7.82E-5	1.0E-04	135
7.82E-7	1.0E-06	1337

Table 9: Example (b):  $p = 3$ .

The Integration Error	The Upper Bound( $\mathcal{E}_{2(I)}$ )	N
4.09E-4	1.0E-02	6
5.11E-6	1.0E-04	104
4.61E-8	1.0E-06	1026

Table 10: Example (c):  $p = 8$ .

The Integration Error	The Upper Bound( $\mathcal{E}_{2(I)}$ )	N
7.47E-02	1.0E-01	25
8.05E-04	1.0E-03	234
8.07E-06	1.0E-05	2324

Table 11: Example (d):  $p = 4$ .

#### 4.2.2 Results

We consider numerically integrating over  $I$  the four functions used in the last section. These functions can be integrated exactly over  $I$ , and therefore we can compute the actual errors of the numerical integration. The results are summarized in Tables 8, 9, 10 and 11 in which the actual integration error, i.e. the left hand side of (54), the upper bound we specified, and the number of nodes generated are shown from the left. For functions (a), (b), and (d), we see that the integrals are very accurately estimated with the errors that are smaller than but very close to the specified upper bounds, which demonstrate the effectiveness of the method. In the example (c), on the other hand, the errors were found to be always nearly two orders of magnitude smaller than the upper bounds, but this does not mean that we generated more nodes than necessary. This is a special case: by the symmetry of the function, a large part of the quadrature error cancel out. It is easy to show that a perfectly symmetric piecewise linear approximation gives the exact value of the integration (zero) for any number of nodes. The approximations constructed by the algorithm are, however, not perfectly symmetric (see Figure 9). Therefore, they do not give the exact value, but very small errors due to a large cancellation that still can occur.

### 4.3 Initial-value problems for ordinary differential equations

It is well-known that the solution of an  $n \times n$  system of ordinary differential equations is a one-dimensional curve in  $\mathbf{R}^n$  (or the so-called phase space) with its independent variable as a parameter. We consider approximating this curve by a piecewise linear continuous function. Although the methodology is basically the same as the previous, we now need to consider numerical schemes that produce the nodal approximation since the parametrization of the solution curve is not known.

### 4.3.1 Algorithm

We consider approximating the solution curve of an  $n \times n$  system of ordinary differential equations,

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}(\mathbf{x}, t) \quad (55)$$

where  $\mathbf{x}(t) = (x_1(t), \dots, x_n(t))$  and  $\mathbf{f}(\mathbf{x}, t) = (f_1(\mathbf{x}, t), \dots, f_n(\mathbf{x}, t))$  with suitable initial conditions, by a piecewise linear continuous function,

$$\mathbf{u}(t) = (u_1(t), \dots, u_n(t)). \quad (56)$$

Given a desired  $L_2$  error, starting from the initial point  $(x_1(0), \dots, x_n(0))$ , we are going to find the grid points successively by the node generation algorithm using the right hand side  $\mathbf{f}$  to compute the first derivatives. The algorithm is exactly the same as before except that we need to compute the nodal values by a numerical scheme this time at every iteration for  $t_{j+1}$ . The generation process will be terminated when a new node exceeds  $t = 1.0$  and the node closest to the end point will be moved to the end point. The first choice of the scheme would be the classical 4th order Runge-Kutta method, which makes the error estimate second-order accurate. For smooth solutions, the method is expected to produce nodal values accurate enough for our purpose. However, for stiff problems, the scheme is not recommended because of its poor stability property. Our choice here is a two-stage 4th order implicit Runge-Kutta scheme of Gauss-Legendre type which is A-stable[8]. Note that A-stable methods does not necessarily imply high accuracy. Its accuracy depends on the step size; the node generation algorithm will play an important role in determining it. The algorithm is summarized as follows.

1. compute  $C$  by  $C = \sqrt{120} \mathcal{E}_{2(I)}$  for a desired error  $\mathcal{E}_{2(I)}$ ;
2. set  $t_{j+1} = t_j + (t_j - t_{j-1})$  ( $t_{j+1} = 0.001$  for  $j = 0$ );
3. compute  $\mathbf{u}(t_{j+1})$  by a numerical scheme;
4. compute a new location  $t_{j+1}$  by (47);
5. compute new solutions  $\mathbf{u}(t_{j+1})$  by a numerical scheme;
6. if (51) is not satisfied, go to 4;
7. if  $t_{j+1} < 1$ , go to 2 (Next node).

If it is known that the solution tends to become linear, including  $\delta$  will be effective as described earlier, producing a uniform grid in such a region.

### 4.3.2 Results

We first consider a scalar problem,

$$\frac{dx(t)}{dt} = -c_s x + [0.3c_s - a\pi e^{-c_s t}] \sin(a\pi t) + 0.3a\pi \cos(a\pi t) \quad t \in [0, 1] \quad (57)$$

with  $c_s = a = 5$  and  $x_1(0) = 1$ . The exact solution is given by

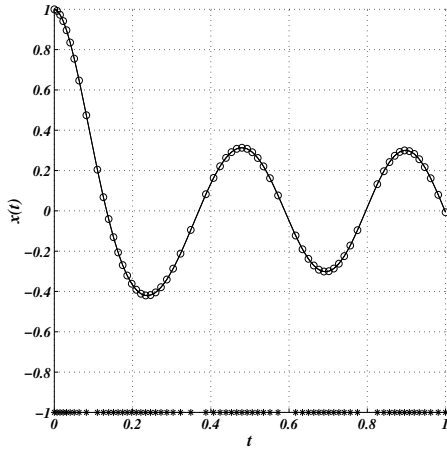
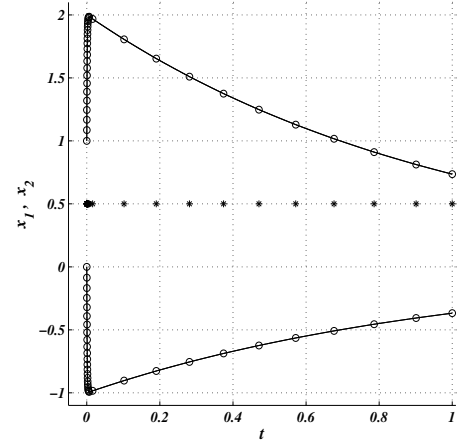
$$x(t) = e^{-c_s t} \cos(a\pi t) + 0.3 \sin(a\pi t). \quad (58)$$

Since this solution is smooth, we use the classical 4th order Runge-Kutta method for this problem. The modification (26) is not used here for simplicity, and  $\delta$  is also set zero. The results are summarized in Table 12, and a plot is shown in Figure 11. Although the improvement of the error over the uniform grids is not impressive, the desired errors have been achieved quite accurately for all the cases. As seen in Figure 11, the nodes are concentrated in the regions where the curvature is relatively large.

$\mathcal{E}_{2(I)}$	N	noi	$\ x - u\ _{L_2(I)}$	$L_{2U}$
1.0E-02	21	12.5	1.084E-02	1.37416E-02
1.0E-03	65	10.2	1.017E-03	1.36935E-03
1.0E-04	205	8.5	1.005E-04	1.35032E-04

Table 12: The results for the scalar case.  $p = 5$ .

$\mathcal{E}_{2(I)}$	N	noi	$\ \mathbf{x} - \mathbf{u}\ _{L_2(I)}$	$L_{2U}$
1.0E-01	4	8.3	9.362E-02	1.328
1.0E-02	11	4.6	1.158E-02	8.701E-01
1.0E-03	34	3.5	1.014E-03	2.747E-01

Table 13: The results for the stiff system.  $p = 2$ .Figure 11: Numerical and exact solutions for the scalar equation.  $\mathcal{E}_{2(I)} = 1.0\text{E-}03$ Figure 12: Numerical and exact solutions for the stiff system.  $\mathcal{E}_{2(I)} = 1.0\text{E-}03$ 

The second test case is a moderately stiff  $2 \times 2$  system of equations taken from [6].

$$\frac{dx_1(t)}{dt} = 998x_1 + 1998x_2 \quad (59)$$

$$\frac{dx_2(t)}{dt} = -999x_1 - 1999x_2 \quad (60)$$

with the initial conditions  $x_1(0) = 1$  and  $x_2(0) = 0$ . The exact solutions are given by

$$x_1 = 2e^{-t} - e^{-1000t}, \quad x_2 = -e^{-t} + e^{-1000t}. \quad (61)$$

We will compute the solution up to  $t = 1$ . The nodal values are computed by the two-stage 4th order implicit Runge-Kutta scheme. As can be seen in Table 13, the desired errors have been achieved accurately for all the cases, and also nearly two orders of magnitude reduction of the error over the uniform grid can be seen. The numerical solutions and the exact solutions for  $\mathcal{E}_{2(I)} = 1.0\text{E-}03$  are shown in Figures 12 and 13. We see that the generated grid is very smooth inside as well as outside the transient region. Figure 14 shows the exact solution (solid line) and the numerical solution (circles) in the phase space which we actually intended to approximate. It is clearly seen, as we expected, that more nodes are placed between

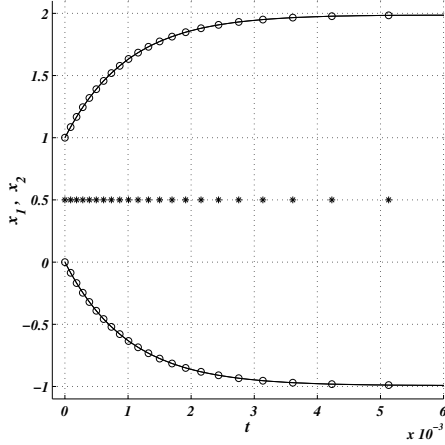


Figure 13: The transient region of the solutions shown in Figure 12.

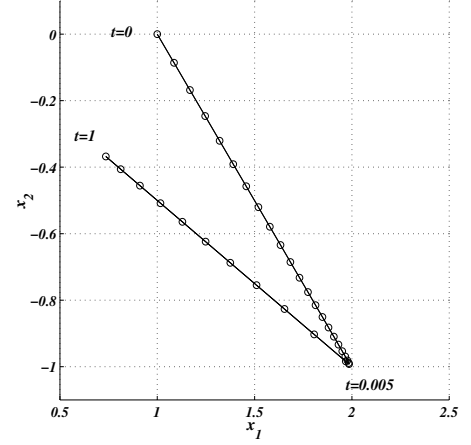


Figure 14: The numerical and exact solution curves in the phase space.

$t=0$  and  $t=0.005$  where the arc length changes very rapidly with respect to  $t$  than those in the rest of the curve, and that the nodes are concentrated in the portion with large curvature near  $t=0.005$ .

A final remark is on extremely stiff problems. The algorithm was tested for the scalar case with  $c_s = 1.0\text{E}+10$  and  $\mathcal{E}_{2(I)} = 1.0\text{E}-01$ . We found that the method generated a few nodes in the extremely narrow transient region and about 700 nodes in the smooth region which are too many, but that the equidistribution was accurately achieved on that grid. It seems that this was caused by the large error in  $f(u, t)$  produced by the product of  $c_s$  and a small error in the solution  $u$ . In other words, the method seems to have achieved the equidistribution by adjusting not the step size but the solution. Further study is necessary on this problem.

## 4.4 Boundary-value problems for ordinary differential equations

In this section, we describe the application of the node generation algorithm to boundary-value problems for ordinary differential equations. It is well known that the problem can be transformed into two initial value problems via the shooting technique. Therefore our algorithm can be applied just as described in the previous section, but a slight modification is necessary.

### 4.4.1 Algorithm

We consider approximating the solution of the linear boundary-value problem,

$$\frac{d^2x}{dt^2} - p(t)\frac{dx}{dt} - q(t)x = r(t) \quad t \in [0, 1] \quad (62)$$

with  $x(0) = \alpha$  and  $x(1) = \beta$ , by a piecewise linear continuous function  $u(t)$ . We assume that  $p(t)$ ,  $q(t)$  and  $r(t)$  are continuous in  $I$  and  $q(t) > 0$  in  $I$ . Then the problem has a unique solution[7]. It is well known that this unique solution can be written as

$$x(t) = x_1(t) + \frac{\beta - x_1(1)}{x_2(1)}x_2(t) \quad (63)$$

where  $x_1$  and  $x_2$  are the solutions of the following *initial value problems* [7].

$$\frac{d^2x_1}{dt^2} = p(t)\frac{dx_1}{dt} + q(t)x_1 + r(t), \quad x_1(0) = \alpha, \quad \frac{dx_1}{dt}(0) = 0 \quad (64)$$

$$\frac{d^2x_2}{dt^2} = p(t)\frac{dx_2}{dt} + q(t)x_2, \quad x_2(0) = 0, \quad \frac{dx_2}{dt}(0) = 1. \quad (65)$$

$\mathcal{E}_{2(I)}$	N	$noi_1$	$noi_2$	$\ x - u\ _{L_2(I)}$	$L_{2U}$
1.0E-01	10	10.6	22.0	1.394E-01	5.043E-01
1.0E-02	17	10.1	15.8	1.105E-02	3.855E-01
1.0E-03	43	9.8	11.1	9.914E-04	1.518E-01

Table 14: The results for (67).  $p = 5$ .

Therefore, each problem can be solved, rewritten as a first order system, by the algorithm described in the previous subsection. It is however important to notice that we are interested not in the solution curve of these systems but the solution of (62). That is to say, we want to use  $f = dx/dt$  to compute  $C_E$ . It appears possible to compute  $f$  during the integration of the initial value problems since we have from (63)

$$f(t) = f_1(t) + \frac{\beta - x_1(1)}{x_2(1)} f_2(t) \quad (66)$$

where  $f_1 = dx_1/dt$  and  $f_2 = dx_2/dt$  which are to be computed together with  $x_1$  and  $x_2$ . However, it requires the knowledge of  $x_1(1)$  and  $x_2(1)$  which are available only after the integration. This leads us to a two-step method; first solve the initial value problems with  $C_E = \Delta t_E \sqrt{\Delta f_{1E}^2 + \Delta f_{2E}^2}$ , and second repeat the computation with  $C_E = \Delta t_E \sqrt{\Delta f_E^2}$ . In the first step, the focus is on to approximate the solution curve in  $(x_1, x_2)$  space accurately because we seek accurate values of  $x_1(1)$  and  $x_2(1)$ . In the second step, we focus on approximating the solution of (62) as  $f$  is now computable by (66). In general, the grids generated in the two steps can be quite different. The algorithm is summarized as follows.

1. compute  $C$  by  $C = \sqrt{120} \mathcal{E}_{2(I)}$  for a desired error  $\mathcal{E}_{2(I)}$ ;
2. Apply the node generation algorithm to the two initial value problems (64) and (65) with  $C_E = \Delta t_E \sqrt{\Delta f_{1E}^2 + \Delta f_{2E}^2}$ ;
3. Repeat the step 2 with  $C_E = \Delta t_E \sqrt{\Delta f_E^2}$ .

#### 4.4.2 Results

We consider the following linear problem.

$$\frac{d^2 x}{dt^2} + \frac{1}{\varepsilon} \frac{dx}{dt} = \frac{n\pi}{\varepsilon} [\varepsilon n\pi \sin(n\pi t) - \cos(n\pi t)] \quad t \in [0, 1] \quad (67)$$

with the boundary conditions  $x(0) = 0$  and  $x(1) = 1$ , and  $\varepsilon = 0.001$  and  $n = 1$ . Although  $q(t) = 0$ , there exists a unique solution given by

$$x(t) = \frac{1 - e^{(-t/\varepsilon)}}{1 - e^{(-1/\varepsilon)}} - \sin(n\pi t). \quad (68)$$

We use the two-stage 4th order implicit Runge-Kutta scheme again to integrate the equivalent initial value problems. As can be seen in Table 14, in which  $noi_1$  is the average of the numbers of iterations at the first step and  $noi_2$  at the second step, the specified errors are achieved accurately for each case. On the other hand, the errors on the uniform grids are significantly larger for each case and decrease very slowly as the grid is refined. The reason for the latter is that no grid points are placed inside the narrow region near  $t = 0$  even with  $N = 44$ . Figures 15 and 16 show the exact solution and the numerical solutions obtained at the first and the second steps respectively, for  $\mathcal{E}_{2(I)} = 1.0E-03$ , where grid points are indicated by stars as before. It can be seen clearly that the two grids differ significantly. As mentioned earlier, this happens because in the first step we seek a good approximation to the solution curve in  $x_1 - x_2$  space, not  $x(t)$ . Although  $x_2$  does have a narrow region of nearly the same width near  $t = 0$ , its

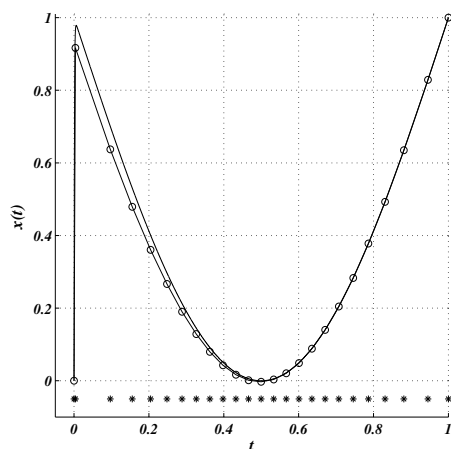


Figure 15: The exact solution and the numerical solution obtained after the first step.

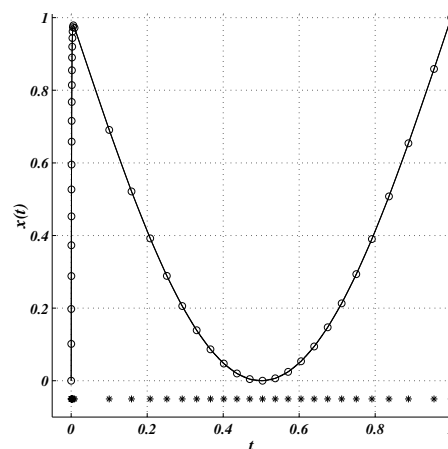


Figure 16: The exact solution and the numerical solution after the second step.

magnitude is of the same order as that width. Therefore the method does not place many points in this region. At this stage, we found that  $\|x - u\|_{L_2(I)} = 3.310\text{E-}02$ ; the prescribed error has not been achieved. In the second step, this solution is magnified by the constant  $\frac{\beta - x_1(1)}{x_2(1)}$  which was found to be 996.8 after the first step, and therefore the narrow region becomes very important, thus making the method place many nodes within it. The iteration to generate each node took nearly 10 on average with  $p = 5$ . The choice of  $p = 5$  was necessary for the iteration to converge near the inflection point. In other regions,  $p = 2$  is in fact sufficient and the iteration takes only one or two to converge.

Finally we remark that the method works for extremely stiff boundary value problems. We applied the algorithm to the case  $\varepsilon = 1.0\text{E-}10$  with  $\mathcal{E}_{2(I)} = 1.0\text{E-}01$  and  $p = 5$ . The algorithm generated 721 interior nodes, giving the error  $\|x - u\|_{L_2(I)} = 1.045\text{E-}01$ . The resulting grid looks almost identical to the one in Figure 16 outside the narrow region, but has many more nodes inside. This result is a total contrast to the one for a stiff initial value problem mentioned in the previous section. The reason for this success lies in the fact that the equivalent initial value problems are not really stiff and that the effect of the stiffness appears only in the constant  $\frac{\beta - x_1(1)}{x_2(1)}$  and does not affect the numerical solutions.

## 5 Conclusions and future work

A local and global  $L_2$  error estimates have been derived for piecewise linear continuous approximations to one-dimensional curves in  $\mathbf{R}^n$ . In fact, the local  $L_2$  error can be computed easily for quadratic functions, but a new fact is that it estimates the error accurately for nonquadratic functions. The accuracy of the estimate has been verified: first- and second-order accurate with third- and fourth-order accurate nodal approximations respectively. Algorithms to construct smooth piecewise linear continuous approximations to one-dimensional curves have been devised based on the error estimates. The algorithm that generates nodes successively has been applied successfully to the approximation of functions, numerical integration, and both initial and boundary value problems for ordinary differential equations.

In this study, the value of  $p$  in the algorithm was fixed in the entire domain. Although very effective, it would be more efficient if it could be changed in such a way that a large value is assigned near inflection points or in a region where the curve is linear, and a small value is assigned elsewhere. This is because the larger the  $p$  is the more iterations it takes in convex regions. This would become an important issue when the method is applied to nonlinear stiff ordinary differential equations where the iteration is required for nodal values as well, and also for large scale problems for which the evaluation of the right hand side

is very expensive. Further study is necessary also for extremely stiff initial-value problems, concerning accurate evaluations of the first derivatives of the solutions, i.e. the right hand sides.

## References

- [1] A. B. WHITE, *On the Selection of Equidistributing Meshes for Two-Point Boundary Value Problems*, SIAM J. Numer. Anal., 16 (1979), pp. 473–502.
- [2] M. J. BAINES, *Algorithms for Optimal Discontinuous Piecewise Linear and Constant  $L_2$  Fits to Continuous Functions with Adjustable Nodes in One and Two Dimensions*, Math. Comp., 62 (1994), pp. 645–669.
- [3] M. W. LIPSCHUTZ, *Differential Geometry*, Schaum's Outline Series, McGraw-Hill, 1969.
- [4] M. J. CLOUD AND B. C. DRACHMAN, *Inequalities with Applications to Engineering*, Springer-Verlag New York, 1998.
- [5] R. KETZSCHER AND S. FORTH. *A Shooting Method for the Generation of the best  $L_1$  piecewise linear interpolation*, AMOR 99/3, Cranfield University.
- [6] C. W. GEAR, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, 1981.
- [7] H. B. KELLER, *Numerical Methods for Two-Point Boundary-Value Problems*, Dover, 1992.
- [8] J. D. LAMBERT, *Numerical Methods for Ordinary differential Equations; The Initial Value Problems*, John Wiley & Sons, Chichester, England 1991.
- [9] J. R. DORMAND, *Numerical Methods for Differential Equations : A Computational Approach*, CRC Press, LLC, 1996.