A Shooting Method for A Node Generation Algorithm

Hiroaki Nishikawa *

W.M.Keck Foundation Laboratory for Computational Fluid Dynamics
Department of Aerospace Engineering, University of Michigan, Ann Arbor, Michigan 48109

January, 2000

Abstract

In this paper, a shooting method is proposed for a node generation algorithm. The node generation algorithm generates nodes succssesively equidistributing a local average error, but cannot terminate at a specified point. A shooting method is developed for this purpose which adjusts the equidistribution quantity by the Secant method. Numerical results are given that shows the effectiveness of the method.

1 Introduction

In this study, a shooting method is studied to capture the endpoint by adjusting C. The idea is taken from [2] where a shooting method for the construction of an L_1 optimum node distribution is studied. In [2], the nodes are generated successively starting from a pair of nodes to minimize L_1 error, and therefore the position of the last node depends upon the choice of the second node. The second node is then be found by the minimization technique of Brent's, which guarantees that the last node is placed at the specified endpoint. In this study, we need to find the value of C such that

2 Node Generation Algorithm

We consider generating nodes for interpolation of a function $x(t)$ in $t \in I = [0, 1]$ that will equidistribute the leading term of the local average error C_E [1]

$$\frac{C_E}{\sqrt{120}} = \frac{1}{\sqrt{120}} \Delta t_E \sqrt{{\Delta f_E}^2 + \frac{16}{7} \Psi_E^2}. \tag{2.1}$$

where $f(t) = dx/dt$, $\Delta t_E = t_{j+1} - t_j$, $\Delta f_E = f(t_{j+1}) - f(t_j)$,

$$\Psi_E = \frac{\Delta x_E}{\Delta t_E} - f(t_m), \quad t_m = \frac{t_{j+1} + t_j}{2}, \tag{2.2}$$

and E denotes the element defined by two consecutive nodes, t_j and t_{j+1}. The nodes can be generated successively starting from the initial point $(t, x(t)) = (0, x(0))$ once we specify the value of C_E. Let C denote this desired value, then given a node t_j, we must find t_{j+1} such that

$$C_E(t_{j+1}) - C = 0. \tag{2.3}$$

*Doctoral Candidate, Aerospace Engineering and Scientific Computing

The iteration formula proposed in [1] is

$$t_{j+1}^{k+1} = t_j + \left[\frac{C}{C_E}\right]^{\frac{1}{p}} (t_{j+1}^k - t_j). \tag{2.4}$$

the superscript k indicates the number of iterations, p is a positive real number. The role of p is to damp an excessivelly large change, i.e. an extremely small Δf_E^k which will occur when the iteration enters a region that is near an inflection point or where the curve is locally linear. We terminate the iteration when the equidistribution is achieved withm 3% error. The node generation 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 $t_{j+1} = t_j + (t_j - t_{j-1})$ $\quad (t_{j+1} = 0.001$ for $j = 0)$;
3. compute a new location t_{j+1} by (2.4);
4. if $|C_E/C - 1| > 1.0$E-03, go to 3;
5. if $t_{j+1} < 1.0$, go to 2 (Next node).

Let N denote the total number of nodes generated by the above algorithm, including the initial node. Then note that we always have $t_N > 1.0$.

3 Shooting Method

Given a set of N nodes generated by the node generation algorithm, we consider solving the following nonlinear equation for C.

$$t_N(C) - 1.0 = 0. \tag{3.1}$$

Note that this has many solutions unless N is fixed. For a convex function, this equation has a unique solution for a fixed N since t_N is a monotone function of C [2]. However it is not guaranteed for nonconvex functions, and therefore we allow N to change, relaxing the requirement for the solution. But the solution we seek is the one with the total number of nodes close to the original number N. Since the explicit functional relationship between t_N and C is not known, we use the Secant method to solve the problem. Given another set of nodes with $C^{(2)} = 1.0001C$ and the initial one $C^{(1)} = C$, we therefore compute C by

$$C^{(n)} = C^{(n-1)} - \frac{t_N(C^{(n-1)}) - 1}{\left[t_N(C^{(n-1)}) - t_N(C^{(n-2)})\right] / \left[C^{(n-1)} - C^{(n-2)}\right]} \tag{3.2}$$

where n is the iteration index greater than two, and when generate another set of nodes and also at each iteration, we modify the step 5 such that the node generation terminates if the number of nodes reaches $N + 1$. The iteration will be terminated when the error in capturing the endpoint becomes less than 5% of the size of the last element,

$$\frac{t_N(C^{(n)}) - 1.0}{t_N(C^{(n)}) - t_{N-1}(C^{(n)})} < 0.05 \tag{3.3}$$

The algorithm is summarized as follows.

1. compute t_{N_1} for $C^{(1)} = C$;
2. compute t_{N_2} for $C^{(2)} = 1.0001C$ with $N_2 \leq N_1 + 1$, set $n = 3$;
3. compute $C^{(n)}$ by (3.2);
4. compute t_{N_n} for $C^{(n)}$ with $N_n \leq N_1 + 1$;
5. if (3.3) is not satisfied, set $n = n + 1$ and go to 3;
6. set $t_{N_n} = 1.0$.

$\mathcal{E}_{2(I)}$	N_1	Iteration	N	$\|x-u\|_{L_2(I)}$
1.0E-02	6	9	6	3.6504E-03
1.0E-04	40	3	40	6.2812E-05
1.0E-06	383	4	383	7.6735E-07

Table 3.1: Example (a): $p = 2$.

$\mathcal{E}_{2(I)}$	N_1	Iteration	N	$\|x-u\|_{L_2(I)}$
1.0E-02	15	5	16	8.0120E-03
1.0E-04	135	2	135	9.9683E-05
1.0E-06	1337	1	1337	1.0001E-06

Table 3.2: Example (b): $p = 3$.

$\mathcal{E}_{2(I)}$	N_1	Iteration	N	$\|x-u\|_{L_2(I)}$
1.0E-02	12	8	13	5.4500E-03
1.0E-04	104	5	105	7.6247E-05
1.0E-06	1026	6	1026	1.0001E-06

Table 3.3: Example (c): $p = 8$.

$\mathcal{E}_{2(I)}$	N_1	Iteration	N	$\|x-u\|_{L_2(I)}$
1.0E-01	25	3	26	8.5046E-02
1.0E-03	234	417	235	9.7102E-04
1.0E-05	2324	1	2325	9.9872E-06

Table 3.4: Example (d): $p = 5$.

4 Results

Numerical tests were performed for the following four different functions which are taken from [1].

(a) $at + (1-a)\left\{1 - e^{(-t/\epsilon)}\right\} / \left\{1 - e^{(-1/\epsilon)}\right\}$ with $\epsilon = 0.04$ and $a = 0.6$

(b) $a(t-0.5)^2 - \frac{a}{4}(1+8\epsilon t) + (1+2\epsilon a)\left\{1 - e^{(-t/\epsilon)}\right\} / \left\{1 - e^{(-1/\epsilon)}\right\}$ 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, (b) and (c) have a single inflection point, and (d) has two inflection points.

All the computations were done with double precision. The value of p was chosen such that the node generation algorithm converges for all the nodes. The results are summarized in Table 3.1 to 3.4 where $\mathcal{E}_{2(I)}$ is the specified L_2 error, N_1 is the initial number of nodes, N is the final number of nodes, and $\|x-u\|_{L_2(I)}$ is the actual L_2 error computed by the five point Gaussian quadrature formula. Some plots are given in Figures 4.1 to 4.4. In amost all cases, the shooting method converges quickly at less than 10 iterations. For nonconvex functions, this is due to the option that it is allowed to increase the total number of nodes by 1. Without it, it would take extremely long or even fail to converge. However there still can be seen a pathological case. In the last example, the method took very long to converge for $\mathcal{E}_{2(I)} = 1.0E-03$. One way to make it converge faster is to change the input value C by a very small amount. We found that $\mathcal{E}_{2(I)} = 1.01E-03$ instead of $1.0E-03$ made it to converge at the first iteration.

However it should be remembered that there can be in principle two bad senarios in the method. First, $C^{(n)}$ can go negative during the iteration. Second the final number of nodes can be significantly less than the starting value. These problems would arise when the method is applied to functions with more inflection points. This robustness issue remains as a future study.

5 Concluding Remarks

A shooting method was proposed to make the node generation algorithm to end at a specified point. It was shown that the method is very effective although not perfectly robust, and that the method is much faster than the relaxation algorithm proposed in [1]. However future work must be done on the robustness of the method. Also, it would be necessary to investigate the possibility that the method is used to generate an equidistribution grid for a given number of nodes which seems to be a more practical problem. This is currently possible only by the relaxation algorithm.

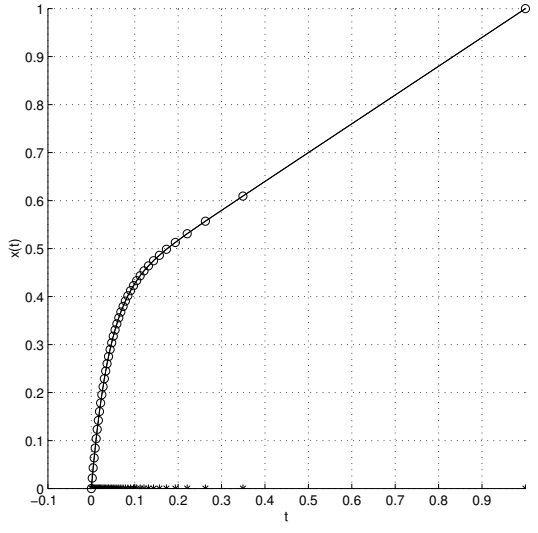

Fig. 4.1: Example (a). $\mathcal{E}_{2(I)}$ =1.0E-04. Circles indicate nodes, and stars are their projection onto t-axis.

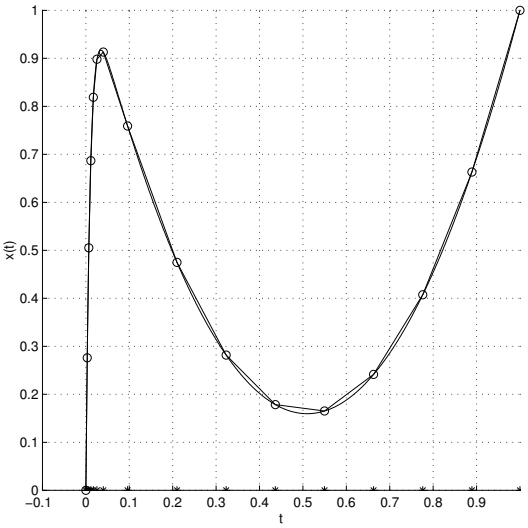

Fig. 4.2: Example (b). $\mathcal{E}_{2(I)}$ =1.0E-02

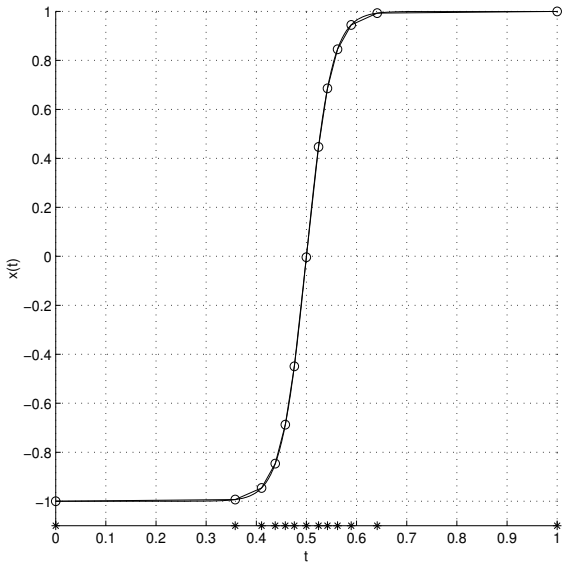

Fig. 4.3: Example (c). $\mathcal{E}_{2(I)}$ =1.0E-02

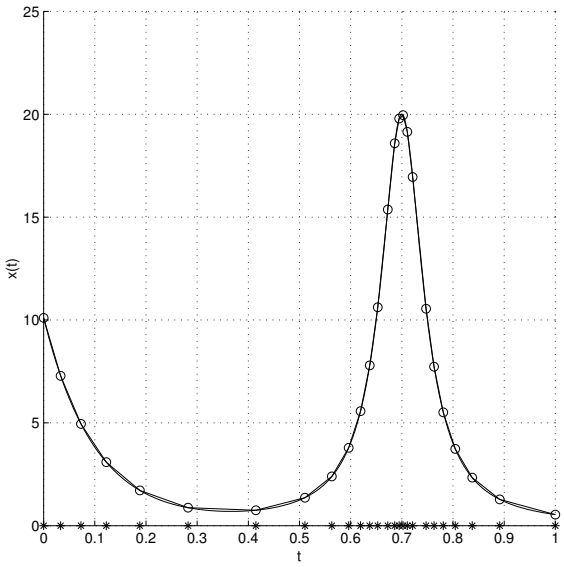

Fig. 4.4: Example (d). $\mathcal{E}_{2(I)}$ =1.0E-01

References

[1] Nishikawa, H., Accurate Piecewise Linear Approximations to 1D Submanifolds: Error Estimates and Algorithms.

[2] Ketzscher, R. and Forth, S., A Shooting Method for the Generation of the best L_1 piecewise linear interpolation, AMOR 99/3, Cranfield University

