Internat. J. Math. & Math. Sci.
VOL. 20 NO. 3 (1997) 577-584

ONE-DIMENSIONAL ADAPTIVE GRID GENERATION

AHMED M. M. KHODIER and ADEL Y. HASSAN
Department of Mathematics
Faculty of Education
Ain Shams University
Roxy, Cairo, Egypt.

(Received November 1, 1995 and in revised form May 8, 1996)

ABSTRACT. In this work, we give an adaptive grid generation method which allows a single point to be added in the regions of large variation. This method uses a quadrature rule as a weight function. Our weight function measures the variation of the solution function on each subinterval of the solution domain. The method is applied to obtain the numerical solutions of some differential equations. A comparison of the numerical solution obtained by this method and other methods is given.

KEY WORDS AND PHRASES : Finite differences, error estimation, weight function, adaptive grid generation.
1991 AMS SUBJECT CLASSIFICATION CODES : 65L50.

1. INTRODUCTION

It is well known that the choice of the mesh points plays an important role in the accuracy of the numerical solution of differential equations. For example, when the solution has large variations like large gradients, peaks, or boundary layers in some parts of the solution domain, we need more grid points in such regions than in regions where the solution changes smoothly. This type of mesh generation is known as adaptive grid generation.

During the last two decades, a significant interest in generating and applying adaptive grids to the numerical solutions of differential equations is surfaced (see for example Denny and Landis [1]; Eiseman [2]; Lentini and Pereyra [3]; Matsuno and Dwyer [4]; and Thompson [5]). The common property of most adaptive grid methods is that they divide the solution domain into subintervals such that some positive weight function has roughly equal value over each subinterval. Most adaptive grid generation methods differ from each other in the choice of the weight function and agree in calculating the value of the given weight function at a single point. Most forms of the weight functions used in literature depend on the first derivative, or the second derivative, or the truncation error of the solution.

Equidistribution schemes in literature usually use a curvilinear coordinate transformation to transform the physical domain into a computational domain where the mesh points are equally spaced and then construct the adaptive mesh by using the differential equation

$$x_{\xi\xi} + x_{\xi}\frac{w_{\xi}}{w} = 0$$

where w is the given weight function. In the next section, we show that the use of such techniques introduces a numerical diffusion to the solution of the problem under consideration.

In this paper, we give an adaptive grid generation method based on using a quadrature rule as a weight function . If the variation of the solution is large on a subinterval, then the quadrature will be large. The advantage of our weight function is that it is calculated on a subinterval of the solution domain, not at a single point of that domain. To avoid the numerical diffusion introduced in most adaptive methods, we use finite differences on irregular grid to solve the given problems in the physical domain without the use of any transformations.

2. FINITE DIFFERENCES AND TRUNCATION ERRORS

Adaptive methods in literature usually use a curvilinear coordinate transformation to transform the physical domain into a computational domain where the mesh points are equally spaced. A one dimensional transformation can be written as

$$x(\xi) = p\left(\frac{\xi}{N}\right) \qquad 0 \le \xi \le N \tag{2.1}$$

where N is the number of subintervals in the solution domain, i.e., h=1/N, h is the step size in the uniform mesh. The first and second derivatives of the solution in physical domain take the form

$$u_x = \frac{u_{\xi}}{x_{\xi}}\ , \tag{2.2}$$

and

$$u_{xx} = \frac{u_{\xi\xi}}{x_{\xi}^2} - \frac{u_{\xi}x_{\xi\xi}}{x_{\xi}^3} \tag{2.3}$$

in the computational domain. Hoffman [6] compared the truncation errors of the first derivatives of the solution function in the physical and computational domains. Thompson et al. [7] proved that if u_x is approximated in the physical domain by using a central difference for u_{ξ} and the exact value of x_{ξ}, then the truncation error in equation (2.2) is given by

$$T_x = \frac{-1}{x_{\xi}}\sum_{n=1}^{\infty}\frac{1}{(2n+1)!}u_{\xi}{}^{2n+1} = \frac{-1}{x_{\xi}}\sum_{n=1}^{\infty}\frac{1}{(2n+1)!}(D_{\xi} + x_{\xi}D_x)^{2n+1}u \tag{2.4}$$

The dominant part of T_x is

$$T_1 = -\frac{h^2 p_{\xi\xi\xi}}{6p_{\xi}}u_x - \frac{1}{2}h^2 p_{\xi\xi}u_{xx} - \frac{1}{6}h^2 p_{\xi}^2 u_{xxx} \tag{2.5}$$

The first term of T_1 contains a numerical viscosity u_x which usually causes troubles and may distroy the solution. This term can be eliminated if we use central difference approximation for x_{ξ}. In this case, we have

$$u_x = \frac{u_{\xi}}{x_{\xi}} = \frac{u(\xi_{i+1}) - u(\xi_{i-1})}{x_{i+1} - x_{i-1}} + T_x' \tag{2.6}$$

The dominant part of T_x' is

$$T_2 = -\frac{1}{2}(x_{i+1} - 2x_i + x_{i-1})u_{xx} - \frac{1}{6}\frac{(x_{i+1} - x_i)^3 - (x_{i-1} - x_i)^3}{x_{i+1} - x_{i-1}}u_{xxx} \tag{2.7}$$

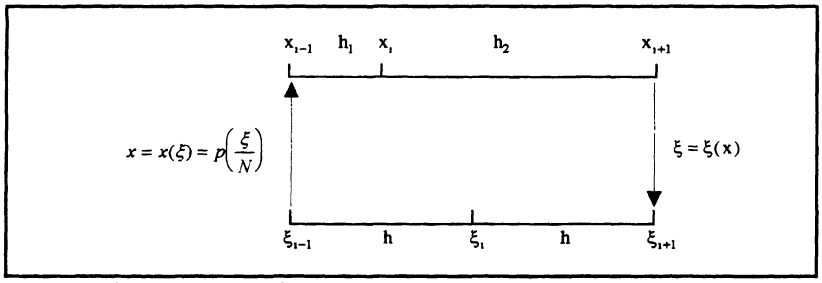

Fig. 2.1. A Mesh Distribution in Physical and Computational domain

In the physical domain, the first derivative u_x can be approximated by using the central difference as follows (see fig. 2.1)

$$u_x = \frac{u(x_{i+1}) - u(x_{i-1})}{h_1 + h_2} + T_x'' \tag{2.8}$$

where the dominant part of the truncation error T_x'' is given by

$$T_3 = -\frac{1}{2}(h_2 - h_1)u_{xx} - \frac{1}{6}(h_1^2 - h_1h_2 + h_2^2)u_{xxx} \tag{2.9}$$

The dominant part of the truncation error in equation (2.9) is the same as that given in equation (2.7). Hence, the central difference approximation of the first derivative u_x has the same truncation error in both physical and computational domain.

Now, we compare the truncation errors for the central difference approximation of the second derivative u_{xx} in the physical and computational domains. In the physical domain, we have

$$u_{xx} = \frac{2[h_1u(x_{i+1}) - (h_1 + h_2)u(x_i) + h_2u(x_{i-1})]}{h_1h_2(h_1 + h_2)} + T_{xx} \tag{2.10}$$

The main part of the truncation error T_{xx} is

$$T_4 = -\frac{1}{3}(h_2 - h_1)u_{xxx} - \frac{1}{12}(h_1^2 - h_1h_2 + h_2^2)u_{xxxx} \tag{2.11}$$

In the computational domain, we have

$$u_{xx} = \frac{u_{\xi\xi}}{x_\xi^2} - \frac{u_\xi x_{\xi\xi}}{x_\xi^3}$$

$$= \frac{u(\xi_{i+1}) - 2u(\xi_i) + u(\xi_{i-1})}{\left(\frac{x_{i+1} - x_{i-1}}{2}\right)^2} - \frac{(u(\xi_{i+1}) - u(\xi_{i-1}))(x_{i+1} - 2x_i + x_{i-1})}{2\left(\frac{x_{i+1} - x_{i-1}}{2}\right)^3} + T_{xx}' \tag{2.12}$$

where the dominant part of the truncation error T_{xx}' is given by

$$T_5 = \frac{(h_2 - h_1)^2}{(h_1 + h_2)^2} u_{xx} - \frac{1}{3}(h_2 - h_1)[1 - \frac{(h_2 - h_1)^2}{(h_1 + h_2)^2}]u_{xxx} \tag{2.13}$$

The first term in equation (2.13) is troublesome since it depends on the derivative u_{xx} which is being approximated. This term vanishes only when $h_1 = h_2$. This leads to uniform mesh in the physical domain which is not our interest in this work.

Hence, the use of central differences for the derivatives of the transformation which eliminate the term u_x in the truncation error of the first derivative now introduces an error that depend on u_{xx}. This error introduces a numerical diffusion.

From the above discussion, it is clear that the truncation errors of the first derivative u_x and the second derivative u_{xx} (as shown in equations (2.7) and (2.13)) contain a numerical diffusion term in the computational domain while in the physical domain, there is a numerical diffusion term only in the truncation error of the first derivative (see equation (2.9)). De Rivas [8] suggested the following approximation

$$u_x = \frac{h_1^2 u(x_{i+1}) - (h_1^2 - h_2^2)u(x_i) - h_2^2 u(x_{i-1})}{h_1 h_2 (h_1 + h_2)} \tag{2.14}$$

which has the main part of the truncation error

$$T_6 = \frac{1}{6} h_1 h_2 u_{xxx}$$

Therefore, it is preferable to solve a problem in the physical domain than to solve it in the computational domain. A comparison of the maximum error between the exact solution and the numerical solution obtained in physical and computational domains is given in example 4.1.

Manteuffel and White [9] showed that the difference scheme obtained by using equations (2.10) and (2.14) yields a second order accurate solution despite the fact that the truncation error is of lower order.

3. CONSTRUCTION OF ADAPTIVE MESH

In this section, we describe our procedure for constructing the adaptive mesh. To illustrate this, let us consider the differential equation

$$au_{xx} + bu_x + cu = g \quad , \quad 0 \le x \le 1 \tag{3.1}$$

where a, b, c, g are, in general, functions of x and the boundary values u(0) and u(1) are given.

Let $u(x) \in C^4[0,1]$ be an approximate solution of equation (3.1) obtained by some interpolation of the discrete numerical solution obtained on crude uniform mesh. For example, the function u(x) can be approximated by using spline or any piecewise polynomial approximations. In this work, we use quadratic spline polynomial to approximate the solution. Then by considering the midpoint $x_m = (x_i + x_{i+1})/2$. the error of Simpson's rule applied to the subinterval $[x_i, x_{i+1}]$ is given by

$$E_i^1(u) = \int_{x_i}^{x_{i+1}} u(x)dx - \frac{1}{6}(x_{i+1} - x_i)[u(x_i) + 4u(x_m) + u(x_{i+1})] = \frac{-1}{2880}(x_{i+1} - x_i)^5 u_{xxxx}(\zeta_i) \tag{3.2}$$

where $x_i < \zeta_i < x_{i+1}$. If we consider the two points $x_\ell = (3x_i + x_{i+1})/4$ and $x_r = (x_i + 3x_{i+1})/4$, then the error becomes

$$E_i^2(u) = \int_{x_i}^{x_m} u(x)dx + \int_{x_m}^{x_{i+1}} u(x)dx - \frac{1}{12}(x_{i+1} - x_i)[u(x_i) + 4u(x_\ell) + 2u(x_m) + 4u(x_r) + u(x_{i+1})]$$

$$= \frac{-1}{92160}(x_{i+1} - x_i)^5[u_{xxxx}(\zeta_i^1) + u_{xxxx}(\zeta_i^2)] \tag{3.3}$$

where $x_i < \zeta_i^1 < x_m$ and $x_m < \zeta_i^2 < x_{i+1}$. By subtracting equation (3.2) from equation (3.3), we get

$$E_i(u) = \left|E_i^2(u) - E_i^1(u)\right| = \frac{1}{12}(x_{i+1} - x_i)\left|u(x_l) - 4u(x_\ell) + 6u(x_m) - 4u(x_r) + u(x_{l+1})\right|$$

$$= \frac{1}{92160}(x_{i+1} - x_i)^5\left|u_{xxxx}(\zeta_i^1) + u_{xxxx}(\zeta_i^2) - 32u_{xxxx}(\zeta_i)\right| \tag{3.4}$$

Hence $$E_i(u) \le \frac{17}{46080}(x_{i+1} - x_i)^5\left|u_{xxxx}(\zeta_i^3)\right|$$

where $u_{xxxx}(\zeta_i^3)$ is the maximum value of $\{u_{xxxx}(\zeta_i), u_{xxxx}(\zeta_i^1), u_{xxxx}(\zeta_i^2)\}$. Equation (3.4) represent the error of Simpson rule on the subinterval $[x_i, x_{i+1}]$. If the fourth derivative u_{xxxx} has a large value on a subinterval, then we have a large value of the error given in equation (3.4). In this case, we add the midpoint x_m to the mesh points and repeat the procedure until some stopping criteria is satisfied. It is well known that the numerical results depend on the properties of the grid. If the mesh ratio is large, the convergence of the iterative solution method may be lost. In this work, we restrict the mesh ratio M_R to be $1/4 \le M_R \le 4$. In this work, the procedure of constructing adaptive mesh was to stop when the number of adaptive mesh points is the same as the uniform mesh . Our procedure can be explained in the following algorithm.

ALGORITHM

Given an approximate solution u of equation (3.1) obtained on a uniform mesh points $\{x_i\}$, i=0,1,2,...,N. The adaptive mesh $\{z_i\}$ can be constructed as in the following steps

1. Start with the uniform mesh points $z_i = x_i$, i=0,1,2,...,N.
2. Compute the error E_i in equation (3.4) on each subinterval $[z_{i-1}, z_i]$.
3. Determine the subinterval of maximum error, say m.
4. Update the mesh points $\{z_i\}$ to include the midpoint of $[z_{m-1}, z_m]$.
5. Repeat steps 2-4 until the stopping criteria is satisfied.

4. NUMERICAL RESULTS

In this section, we give the numerical solution of some examples on adaptive mesh which is generated by the above algorithm. A comparison of the numerical solution obtained on the adaptive mesh and on a uniform mesh is given.

EXAMPLE 4.1. $\frac{d^2u}{dx^2}+4u=20x^3+4x^5 \quad , \quad 0\le x\le 1$

In this example, we compare the numerical results obtained in the physical and computational domain. The boundary conditions are obtained from the exact solution $u(x)=x^5$. The results are given in table 4.1. Also, a comparison of the numerical results obtained by our method and some other methods is given in table 4.2. The adaptive mesh is shown in fig. 4.1.

EXAMPLE 4.2. $\frac{d^2u}{dx^2}-q\frac{du}{dx}=0 \quad , \quad 0\le x\le 1$

with the boundary conditions $u(0)=0$ and $u(1)=1$. This example is considered by Lick and Gaskins [10]. It has a boundary layer of thickness 1/q near the right boundary. Numerical results for q=20 are given in table 4.3, and the adaptive mesh is shown in figure 4.2.

EXAMPLE 4.3. $\frac{rx}{r-1}\frac{d^2u}{dx^2}+\frac{du}{dx}=0 \quad , \quad 0\le x\le 1$

where the boundary conditions are $u(0)=0$ and $u(1)=1$. The first and higher order derivatives of the solution function have singularities at x=0 for $r>1$. The numerical results obtained for r=1.25 are given in table 4.4, and the adaptive mesh distribution is shown in figure 4.3.

EXAMPLE 4.4 $\frac{d^2u}{dx^2}+\frac{du}{dx}+tu=0 \quad , \quad 0\le x\le 1 \quad , \quad t>1/4$

with boundary conditions $u(0)=0$ and $u(1)=\sin(\frac{1}{2}\sqrt{4t-1})$. Numerical results given in table 4.5 are due to t=10. The adaptive mesh is shown in figure 4.4.

REMARK. In all tables, adaptive[1], adaptive[2], and adaptive represent the numerical results on adaptive grid obtained by using the first derivative, second derivative, and our quadrature rule (equation (3.4)) as a weight function.

Transformation	No. of Subintervals	Abs. Max. Error in Phys. Domain	Abs. Max. Error in Comp. Domain
$x(\xi)=(e^{\xi}-1)/(e-1)$	32	23×10^{-5}	20×10^{-4}
	64	59×10^{-6}	50×10^{-5}
$x(\xi)=\xi^4$	32	18×10^{-6}	30×10^{-5}
	64	4×10^{-6}	75×10^{-6}

Table 4.1. Numerical Results of Example 4.1 in Physical and Computational domain

Mesh Type	Absolute Maximum Error		
	80 Subintervals	160 Subintervals	320 Subintervals
uniform	48×10^{-6}	25×10^{-6}	14×10^{-6}
adaptive[1]	29×10^{-6}	16×10^{-6}	7×10^{-6}
adaptive[2]	13×10^{-6}	8×10^{-6}	3×10^{-6}
adaptive	67×10^{-7}	39×10^{-7}	12×10^{-7}

Table 4.2. Numerical Results of Example 4.1

Mesh Type	Absolute Maximum Error		
	80 Subintervals	160 Subintervals	320 Subintervals
uniform	20×10^{-4}	49×10^{-5}	15×10^{-5}
adaptive[1]	14×10^{-4}	42×10^{-5}	13×10^{-5}
adaptive[2]	72×10^{-5}	30×10^{-5}	48×10^{-6}
adaptive	10×10^{-5}	29×10^{-6}	8×10^{-6}

Table 4.3. Numerical Results of Example 4.2

Mesh Type	Absolute Maximum Error		
	80 Subintervals	160 Subintervals	320 Subintervals
uniform	30×10^{-4}	18×10^{-4}	11×10^{-4}
adaptive[1]	26×10^{-4}	13×10^{-4}	97×10^{-5}
adaptive[2]	21×10^{-4}	98×10^{-5}	73×10^{-5}
adaptive	14×10^{-4}	53×10^{-5}	24×10^{-5}

Table 4.4. Numerical Results of Example 4.3

Mesh Type	Absolute Maximum Error		
	80 Subintervals	160 Subintervals	320 Subintervals
uniform	17×10^{-3}	72×10^{-4}	11×10^{-4}
adaptive[1]	40×10^{-4}	18×10^{-4}	71×10^{-5}
adaptive[2]	23×10^{-4}	6×10^{-4}	16×10^{-5}
adaptive	6×10^{-4}	20×10^{-5}	6×10^{-5}

Table 4.5. Numerical Results of Example 4.4

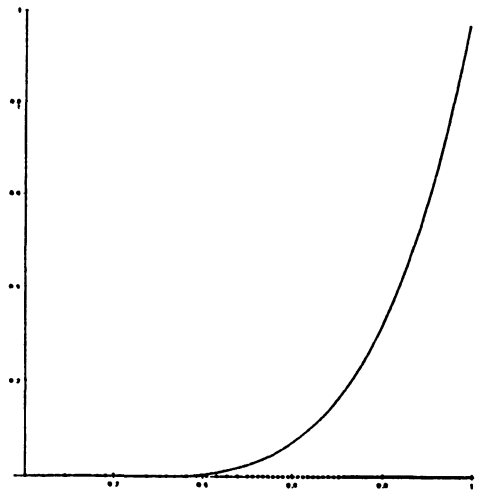

Fig. 4.1. Adaptive Mesh for Example 4.1 with 80 Subintervals

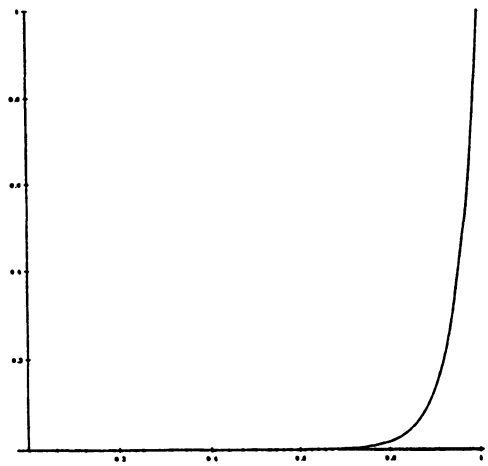

Fig. 4.2. Adaptive Mesh for Example 4.2 with 80 Subintervals

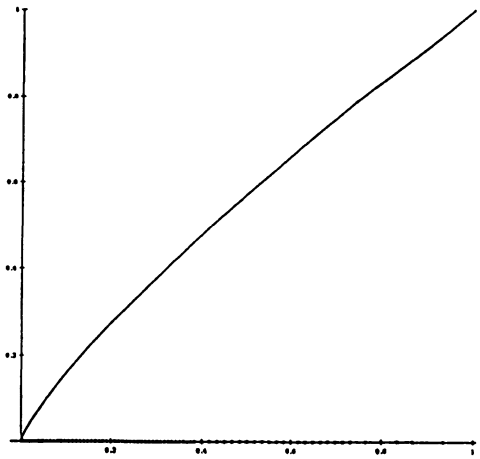

Fig. 4.3. Adaptive Mesh for Example 4.3 with 80 Subintervals

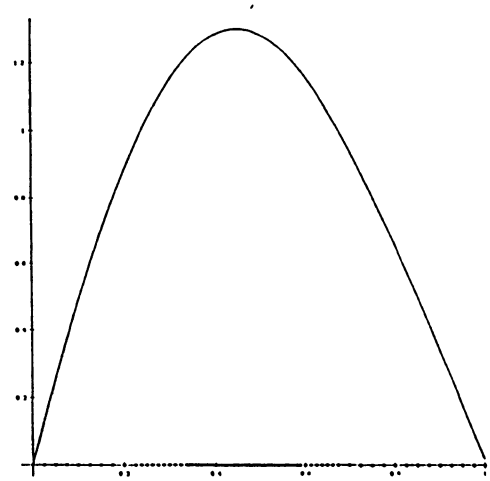

Fig. 4.4. Adaptive Mesh for Example 4.4 with 80 Subintervals

5. CONCLUSIONS

In this work, we have analyzed the numerical solution of differential equations in the physical and computational domains. The difference approximations of the first derivatives have the same truncation errors in both domains while the difference approximations of the second derivatives introduce artificial viscosity in the computational domain. The artificial viscosity can be avoided by solving the differential equations in the physical domain.

The adaptive grid generation methods generates a well suited mesh for the problems under consideration specially if the solution of these problems has a large variation in some parts of the solution domain. Also the use of adaptive methods does not require any priori knowledge about the solution or even the locations of its large variations.

The numerical results presented in this paper show that the error obtained by using our adaptive method is of almost 50% of the error obtained by using other methods.

REFERENCES

[1] DENNY, V. E. and LANDIS, R. B. , "A New Method for Solving Two Point Boundary Value Problems Using Optimal Node Distribution", J. of Comp. Phys. 9 (1972), 120-137.

[2] EISEMAN, P. R. , "Adaptive Grid Generation", Computer Methods in Applied Mechanics and Eng. 64 (1987), 321-376.

[3] LENTINI, M. and PEREYRA, V. , "An Adaptive Finite Difference Solver for Nonlinear Two Point Boundary Value Problems with Mild Boundary Layers", Siam J. Numer. Anal. 14, No. 1 (1977), 91-111.

[4] MATSUNO, K. and DWYER, H. A. , "Adaptive Methods for Elliptic Grid Generation", J. of Comp. Phys. 77 (1988), 40-52.

[5] THOMPSON, J. F. , "A Survey of Dynamically Adaptive Grid in the Numerical Solution of Partial Differential Equations", Appl. Numer. Math. 1 (1985), 3-27.

[6] HOFFMAN, J. D., "Relationship between the Truncation Errors of Centered Finite Difference Approximation on Uniform and Nonuniform Meshes", J. of Comp. Phys. 46 (1982), 469- 474.

[7] THOMPSON, J. F. , WARSI, Z. U. A. , and MASTIN, C. W. , "Order of Difference Expression on Curvilinear Coordinate Systems", Advances in Grid Generation, ASME (1983), 17-28.

[8] DE RIVAS, E. K. , "On the Use of Nonuniform Grids in Finite Difference Equations", J. of Comp. Phys. 10 (1972), 202-210.

[9] MANTEUFFEL, T. A. and WHITE, A. B. , "The Numerical Solution of Second-Order Boundary Value Problems on Nonuniform Meshes", Math. of Computation, 47, No. 176 (1986), 511-535.

[10] LICK, W. and GASKINS, T. , "A Consistent and Accurate Procedure for Obtaining Difference Equations from Differential Equations", Inter. J. for Numer. Math. in Eng. 20 (1984), 1433-1441.