Optimal Cubic Spline Collocation on Non-uniform Partitions

Christina C. Christara and Kit Sun Ng
Department of Computer Science
University of Toronto
Toronto, Ontario M5S 3G4, Canada
{ccc,ngkit}@cs.utoronto.ca

November 2004

Abstract

We develop optimal non-uniform Cubic Spline Collocation methods for second-order two-point Boundary Value Problems. The formulation and analysis of the methods are basedon a mapping function from uniform to non-uniform partitions. However, the cubic spline collocation method is turned into a mapping-free method. The non-uniform partition method is integrated with adaptive grid techniques, and grid size and error estimators. One adaptive grid technique is based on the construction of a mapping function. Another is mapping-free and resembles the technique used in COLSYS (COLNEW) [2, 4]. Numerical results on a variety of problems, including problems with boundary or interior layers, and singular perturbation problems indicate that, for most problems, the cubic spline collocation method requires less computational effort for the same error tolerance, and has equally reliable error estimators, when compared to Hermite piecewise cubic collocation. Comparison results with quadratic spline collocation are also presented.

AMS Subject Classification: 65L10, 65L20, 65L50, 65L60, 65L70, 65D05, 65D07.

Key words: spline collocation, second-order two-point boundary value problem, error bounds, optimal order of convergence, adaptive grid, grid size estimator, error estimator, spline interpolation.

1 Introduction

Consider a linear two-point Boundary Value Problem (BVP) described by the operator equation

$$\mathbf{L}u \equiv ru'' + pu' + qu = g \quad \text{in} \quad \Omega \equiv (0,1), \tag{1}$$

where r, p, q and g are given functions of x, and u is the unknown function of x, subject to boundary conditions on the boundary $\partial\Omega$ of Ω, described by

$$\mathbf{B}u \equiv \{\alpha_0 u(0) + \beta_0 u'(0) = \gamma_0, \alpha_1 u(1) + \beta_1 u'(1) = \gamma_1\}, \tag{2}$$

where α_i, β_i and γ_i, $i = 0, 1$ are given scalars.

Collocation is a simple-to-implement and quadrature-free method for solving differential equation problems. Piecewise polynomials of certain degree and continuity often serve as approximation spaces for collocation. The most well-known piecewise polynomial collocation method for the solution of (1)-(2) is Gaussian collocation introduced by De Boor and Swartz [9]. A commonly used Gaussian collocation determines a $\mathbf{C}^1$ piecewise cubic polynomial approximation u_Δ by collocating at the two Gaussian points in each partition interval, as well as the two boundary conditions. Under some conditions, the global error in u_Δ and the first derivative error at the nodes are shown to be $O(H^4)$, where H is the maximum partition length.

Cubic splines, i.e. cubic piecewise polynomials with $\mathbf{C}^2$ continuity, have always been a popular alternative for approximating functions. Reasons for this include the fact that certain applications (e.g. applications in automobile industry or computer graphics, where smooth, "pleasing to the eye" curves are desirable) require higher continuity of the solution, and the fact that cubic splines give rise to a smaller bandwidth and sparser linear system than piecewise cubic polynomials in $\mathbf{C}^1$ for the same number of grid points. However, the standard cubic spline collocation method applied to (1)-(2) produces $O(H^2)$ approximations even on uniform partitions. Moreover, cubic spline interpolation gives an $O(H^4)$ error bound.

To obtain the optimal fourth order convergence, several researchers proposed deferred-correction [10] or extrapolated cubic spline collocation methods [1] and [7]. Both types of methods are based on an appropriate perturbation of the differential equation. The extrapolated or one-step method applies the perturbation to the operator (or, in discrete form, to the matrix arising). The deferred-correction or two-step method, after computing a second order approximation by standard collocation, applies the perturbation to the right-hand side. These optimal cubic spline collocation methods were extended to two-dimensional second-order elliptic BVPs for rectangular domains in [11].

None of the above optimal spline collocation methods has been extended to non-uniform partitions. The applicability of a discretization method to non-uniform partitions is essential for problems with rough solution behaviour, layers, etc, which are usually solved by adaptive techniques, and for multi-dimensional problems in non-rectangular domains. In this paper, we develop optimal non-uniform partition cubic spline collocation (CSC) methods for linear one-dimensional second-order BVPs, and in [6] we do the same for quadratic spline collocation (QSC).

The outline of this paper is as follows. In Section 2, we derive and analyze an optimal two-step CSC method on non-uniform partitions. The derivation and analysis of the method assumes that the non-uniform partition is the image of a uniform partition under an appropriate mapping. However, the final formulation and implementation of the method is mapping-free. No mapping is assumed or computed, in contrast to the QSC method in [6]. We integrate our method with adaptive grid techniques in Section 3. One technique is similar to that for QSC in [6] and is based on a mapping, while the second technique is similar to that of COLSYS (COLNEW) [2, 4], with a few differences which we elaborate. In Section 4, we present numerical results that demonstrate the behaviour of the CSC method on a variety of problems, including problems with boundary or interior layers, and singular perturbation problems. We compare the results from CSC and from Hermite piecewise cubic collocation (which we refer to as HPCC) as implemented in COLSYS. We also present results from QSC for comparison.

2 Non-uniform Partition Cubic Spline Collocation

In this section, we develop an optimal CSC method for second-order two-point boundary value problems on non-uniform partitions. Our results reduce to those in [7], if the partition is uniform.

Consider the two-point BVP (1)-(2). Let $\Delta \equiv \{x_0 = 0 < x_1 < \ldots < x_N = 1\}$ be a uniform partition of $\overline{\Omega}$ with stepsize $h = 1/N$. For convenience, let $T \equiv \{\tau_i = (x_i + x_{i-1})/2; i = 1, \dots, N\}$ be the set of midpoints. Furthermore, define the Gaussian points $\delta_{ij} = x_i - \lambda_j h$; $j = 1, 2$, $i = 1, \dots, N$, where $\lambda_1 = (3 - \sqrt{3})/6$ and $\lambda_2 = (3 + \sqrt{3})/6$.

Let $w(x) : \overline{\Omega} \to \overline{\Omega}$ be a bijective function in $\mathbf{C}^3$, with $w'(x) > 0$, $\forall x \in \overline{\Omega}$. Let $\Delta_w \equiv \{s_i = w(x_i), i = 0, \dots, N\}$ be the partition of $\overline{\Omega}$ with respect to which the splines are defined. For CSC, the partition points (nodes) of Δ_w will also be the collocation points. That is, for the initial development of the method, we restrict the non-uniform grid (collocation) points to be images of uniform grid (collocation) points under w. This restriction is lifted in Section 3, where we allow any set of grid (collocation) points, and do not require any mapping between uniform and non-uniform points. For convenience, let $T_w \equiv \{w_i = w(\tau_i), i = 1, \dots, N\}$. Let also $\sigma_{ij} \equiv w(\delta_{ij})$, $j = 1, 2$, $i = 1, \dots, N$. In consistency with [6], let $H_i = s_{i+1} - s_i$, $i = 0, \dots, N-1$, and $H = \max_{i=0,\dots,N-1} H_i$.

For any given $w(x)$, we denote by $S^3_{\Delta_w}$ the cubic spline space with respect to the partition Δ_w, that is the space of piecewise cubic polynomials which are globally in $\mathbf{C}^2$. We choose as basis functions for $S^3_{\Delta_w}$ the functions $\psi_i(x)$, $i = 0, \dots, N+2$, where

$$
\psi_i(x) \equiv \begin{cases}
\frac{(x-s_{i-2})^3}{(s_{i+1}-s_{i-2})(s_i-s_{i-2})(s_{i-1}-s_{i-2})} & \text{for } s_{i-2} \le x \le s_{i-1} \\
\frac{(x-s_{i-2})}{(s_{i+1}-s_{i-2})}\left(\frac{(x-s_{i-2})(s_i-x)}{(s_i-s_{i-2})(s_i-s_{i-1})} + \frac{(s_{i+1}-x)(x-s_{i-1})}{(s_{i+1}-s_{i-1})(s_i-s_{i-1})}\right) & \text{for } s_{i-1} \le x \le s_i \\
+\frac{(s_{i+2}-x)(x-s_{i-1})^2}{(s_{i+2}-s_{i-1})(s_{i+1}-s_{i-1})(s_i-s_{i-1})} & \\
\frac{(x-s_{i-2})(s_{i+1}-x)^2}{(s_{i+1}-s_{i-2})(s_{i+1}-s_{i-1})(s_{i+1}-s_i)} & \text{for } s_i \le x \le s_{i+1} \\
+\frac{(s_{i+2}-x)}{(s_{i+2}-s_{i-1})}\left(\frac{(x-s_{i-1})(s_{i+1}-x)}{(s_{i+1}-s_{i-1})(s_{i+1}-s_i)} + \frac{(s_{i+2}-x)(x-s_i)}{(s_{i+2}-s_i)(s_{i+1}-s_i)}\right) & \\
\frac{(s_{i+2}-x)^3}{(s_{i+2}-s_{i-1})(s_{i+2}-s_i)(s_{i+2}-s_{i+1})} & \text{for } s_{i+1} \le x \le s_{i+2} \\
0 & \text{elsewhere.}
\end{cases} \tag{3}
$$

Let $S \in S^3_{\Delta_w}$ be the cubic spline interpolant of u such that

$$
\begin{aligned}
S''(s_0) &= u''(s_0) - \tfrac{1}{24} H_0(5H_0 - 4H_1 + H_2)u^{(4)}(s_0), \\
S(s_i) &= u(s_i), \quad i = 0, \dots, N, \\
S''(s_N) &= u''(s_N) - \tfrac{1}{24} H_{N-1}(5H_{N-1} - 4H_{N-2} + H_{N-3})u^{(4)}(s_N).
\end{aligned} \tag{4}
$$

For consistency with [6], adopt the following notation: $w_i \equiv w(\tau_i)$ and $e(x) \equiv u(x) - S(x)$. In this section only, α, β, γ, α_1, β_1, γ_1, α_2, β_2 and γ_2 denote scalars defined within the section.

We will develop expansions of the errors of the derivatives of S at the collocation points. We consider the second derivative error at the grid points. By setting up the equations,

$$
\alpha_2 S''(s_{i-1}) + \beta_2 S''(s_i) + \gamma_2 S''(s_{i+1}) = \alpha S(s_{i-1}) + \beta S(s_i) + \gamma S(s_{i+1}) \tag{5}
$$

for $i = 1, \dots, N-1$, and solving for α, β, γ, α_2, β_2 and γ_2 with Maple we have

$$
\begin{aligned}
\alpha &= \tfrac{6}{H_{i-1}}, & \alpha_2 &= H_{i-1}, \\
\beta &= -\tfrac{6(H_i + H_{i-1})}{H_{i-1}H_i}, & \beta_2 &= 2(H_i + H_{i-1}), \\
\gamma &= \tfrac{6}{H_i}, & \gamma_2 &= H_i.
\end{aligned} \tag{6}
$$

The proof of the following lemma is trivial and is therefore omitted.

LEMMA 1 *If $w(x) \in \mathbf{C}^3[\overline{\Omega}]$, then*

$$H_i = hw'(x_i) + \tfrac{h^2}{2}w''(x_i) + O(h^3), \quad H_{i-1} = hw'(x_i) - \tfrac{h^2}{2}w''(x_i) + O(h^3). \tag{7}$$

where the index i takes all possible values consistent with the definition of the stepsizes.

THEOREM 1 *If $u \in \mathbf{C}^6[\overline{\Omega}]$, $w(x) : \overline{\Omega} \to \overline{\Omega}$ is a bijective function in $\mathbf{C}^3$, with $w'(x) > 0$, $\forall x \in \overline{\Omega}$, $w^{-1} \in \mathbf{C}^1[\overline{\Omega}]$, and S is defined by (4), then*

$$S''(s_i) = u''(s_i) - \frac{1}{12}H_iH_{i-1}u^{(4)}(s_i) + O(h^4) \tag{8}$$

for $i = 1, \ldots, N-1$.

PROOF
Since $S(s_i) = u(s_i)$, $i = 0, \ldots, N$ and $u \in \mathbf{C}^6$, the right-hand side of (5) after applying Taylor expansion to $u(s_{i\pm1})$ is equivalent to

$$\begin{aligned} \alpha u(s_{i-1}) + \beta u(s_i) + \gamma u(s_{i+1}) = {} & 3(H_i + H_{i-1})[u''(s_i) + \frac{1}{3}(H_i - H_{i-1})u^{(3)}(s_i) \\ + {} & \frac{1}{12}H_iH_{i-1}u^{(4)}(s_i)] + O(H^4),\ i = 1, \ldots, N-1. \end{aligned} \tag{9}$$

Moreover, using Taylor expansion of a function $f \in \mathbf{C}^4$ we can show

$$\begin{aligned} \alpha_2 f(s_{i-1}) + \beta_2 f(s_i) + \gamma_2 f(s_{i+1}) = {} & 3(H_i + H_{i-1})[f(s_i) + \frac{1}{3}(H_i - H_{i-1})f'(s_i) \\ + {} & \frac{1}{6}H_iH_{i-1}f''(s_i)] + O(H^4),\ i = 1, \ldots, N-1. \end{aligned} \tag{10}$$

Let $\wedge u(s_i) \equiv \frac{1}{3(H_i+H_{i-1})}(\alpha_2 u(s_{i-1}) + \beta_2 u(s_i) + \gamma_2 u(s_{i+1}))$, $i = 1, \ldots, N-1$. Then by using Lemma 1 to expand the truncation errors of (9) and (10), we have

$$\wedge u(s_i) = u''(s_i) + \frac{1}{3}(H_i - H_{i-1})u^{(3)}(s_i) + \frac{1}{12}H_iH_{i-1}u^{(4)}(s_i) + O(h^4),\ i = 1, \ldots, N-1.$$

Using the definition of $\wedge$, relation (10) becomes

$$\begin{aligned} \wedge f(s_i) = {} & f(s_i) + \frac{1}{3}(H_i - H_{i-1})f'(s_i) \\ + {} & \frac{1}{6}(H_i^2 - H_iH_{i-1} + H_{i-1}^2)f''(s_i) + O(h^4),\ i = 1, \ldots, N-1. \end{aligned}$$

Let $v = h^2 w' \circ w^{-1}$. By using Lemma 1, we have

$$v(s_i) = H_iH_{i-1} + O(h^4),\ i = 1, \ldots, N-1, \tag{11}$$

$$v(s_i) = H_i(5H_i - 4H_{i+1} + H_{i+2})/2 + O(h^4),\ i = 0, \ldots, N-3, \tag{12}$$

$$v(s_i) = H_{i-1}(5H_{i-1} - 4H_{i-2} + H_{i-3})/2 + O(h^4),\ i = 3, \ldots, N. \tag{13}$$

Now, by taking $f = u'' - \frac{1}{12}vu^{(4)}$ and using (11), we have

$$\wedge(S''(s_i) - f(s_i)) = O(h^4),\ i = 1, \ldots, N-1. \tag{14}$$

Moreover, from the end conditions in (4) and relations (12) and (13), we have

$$S''(s_0) - f(s_0) = O(h^4) \tag{15}$$

and

$$S''(s_N) - f(s_N) = O(h^4). \tag{16}$$

Relations (15), (14) and (16) (in that order) form a system of equations that is strictly diagonally dominant. More specifically, $\frac{|\beta_2|-|\alpha_2|-|\gamma_2|}{3(H_i+H_{i-1})} = \frac{1}{3}$. Therefore, the infinity norm of the inverse of the matrix of equations (15), (14) and (16) is bounded by 3. This implies that

$$S''(s_i) - f(s_i) = O(h^4), \quad i = 0, \ldots, N.$$

$\diamondsuit$

The proof of the following theorem is similar to the proofs of Theorems 4 and 5 in [6], and is therefore omitted.

THEOREM 2 *Under the assumptions of Theorem 1,*

$$\begin{aligned} |e'(w_i)| &= O(h^4), & (17)\\ |e'(s_i)| &= O(h^4), & (18)\\ |e''(\sigma_{ij})| &= O(h^3), & (19)\\ |e^{(3)}(w_i)| &= O(h^2), \quad \textit{and} & (20)\\ \|e^{(k)}\|_\infty &= O(h^{4-k}), \;\; k = 0, \ldots, 3. & (21) \end{aligned}$$

The following theorem is the cubic spline counterpart of Theorem 6 in [6], and its proof is similar to that proof.

THEOREM 3 *Under the assumptions of Theorem 1,*

$$\begin{aligned} u^{(4)}(s_0) &= \frac{(H_0 + H_1) \sqcap S''(s_1) - H_0 \sqcap S''(s_2)}{H_1} + O(h^2), & (22)\\ u^{(4)}(s_i) &= \sqcap S''(s_i) + O(h^2), \quad i = 1, \ldots, N-1, & (23)\\ u^{(4)}(s_N) &= \frac{(H_{N-1} + H_{N-2}) \sqcap S''(s_{N-1}) - H_{N-1} \sqcap S''(s_{N-2})}{H_{N-2}} + O(h^2), & (24) \end{aligned}$$

where $\sqcap u(s_i) \equiv \frac{2H_i u(s_{i-1}) - 2(H_{i-1}+H_i)u(s_i) + 2H_{i-1}u(s_{i+1})}{H_{i-1}(H_{i-1}+H_i)H_i}$.

Consider solving the BVP (1)-(2). Based on the relations from Theorems 1-2, we observe that the interpolant S of u satisfies the relations

$$\begin{aligned} \mathbf{L}S(s_0) &= g(s_0) - \frac{r(s_0)}{24} H_0(5H_0 - 4H_1 + H_2)u^{(4)}(s_0) + O(h^4),\\ \mathbf{L}S(s_i) &= g(s_i) - \frac{r(s_i)}{12} H_i H_{i-1} u^{(4)}(s_i) + O(h^4), \;\; i = 1, \ldots, N-1, \\ \mathbf{L}S(s_N) &= g(s_N) - \frac{r(s_N)}{24} H_{N-1}(5H_{N-1} - 4H_{N-2} + H_{N-3})u^{(4)}(s_N) + O(h^4), \end{aligned} \tag{25}$$

and

$$\mathbf{B}S(s_i) = \gamma(s_i) + O(h^4), \;\; i = 0, N. \tag{26}$$

Notice that due to Theorem 3, relations (25) can be written as

$$\mathbf{L}S(s_i) = g(s_i) - P_{\mathbf{L}}S(s_i) + O(h^4), \;\; i = 0, \ldots, N, \tag{27}$$

where

$$P_{\mathbf{L}}S(s_0) = \frac{r(s_0)}{24} H_0(5H_0 - 4H_1 + H_2) \frac{(H_0 + H_1) \sqcap S''(s_1) - H_0 \sqcap S''(s_2)}{H_1}, \tag{28}$$

$$P_{\mathbf{L}}S(s_i) = \frac{r(s_i)}{12} H_i H_{i-1} \sqcap S''(s_i), \;\; i = 1, \ldots, N-1, \tag{29}$$

$$\begin{aligned} P_{\mathbf{L}}S(s_N) &= \frac{r(s_N)}{24} H_{N-1}(5H_{N-1} - 4H_{N-2} + H_{N-3}) \\ &\quad \frac{(H_{N-1} + H_{N-2}) \sqcap S''(s_{N-1}) - H_{N-1} \sqcap S''(s_{N-2})}{H_{N-2}}. \end{aligned} \tag{30}$$

We are now ready to present an *optimal two-step CSC method* to determine an approximation $u^3_\Delta \in S^3_{\Delta_w}$ of the solution of the BVP (1)-(2).

Step 1: Determine $u^3_{\Delta[1]} \in S^3_{\Delta_w}$ by forcing it to satisfy

$$\mathbf{L}u^3_{\Delta[1]} = g \;\text{ in }\; \Delta_w, \tag{31}$$

$$\mathbf{B}u^3_{\Delta[1]} = \gamma \;\text{ on }\; T_{w\mathbf{B}}. \tag{32}$$

Step 2: Determine $u^3_\Delta \in S^3_{\Delta_w}$ by forcing it to satisfy

$$\mathbf{L}u^3_\Delta = g - P_{\mathbf{L}}u^3_{\Delta[1]} \;\text{ in }\; \Delta_w, \tag{33}$$

$$\mathbf{B}u^3_\Delta = \gamma \;\text{ on }\; T_{w\mathbf{B}}. \tag{34}$$

Similarly as for QSC in [6], we can also define an *optimal one-step CSC method.* Using techniques similar to those in [6] we can show the following theorems.

THEOREM 4 *If*
(A1) the coefficients p and q, and the right-hand side g are in $\mathbf{C}[\Omega]$,
(A2) the BVP $\mathbf{L}u = g, \mathbf{B}u = 0$ has a unique solution,
(A3) the BVP $u'' = 0, \mathbf{B}u = 0$ has a unique solution,
(A4) the assumptions of Theorem 1 hold,
then $u^3_{\Delta[1]} \in S^3_{\Delta_w}$ defined by (31)-(32) exists, is unique, and satisfies the global error estimates

$$\|(u - u^3_{\Delta[1]})^{(k)}\|_\infty = O(h^2), \; k = 0, 1, 2, \;\; \textit{and} \;\; \|(u - u^3_{\Delta[1]})^{(3)}\|_\infty = O(h), \tag{35}$$

and the local error estimates

$$|(u - u^3_{\Delta[1]})^{(3)}(w_i)| = O(h^2), \; i = 1, \ldots, N. \tag{36}$$

THEOREM 5 *Under the assumptions of Theorem 4, and the assumption that $u'' - u^{3''}_{\Delta[1]}$ has a smooth expansion at the collocation points, $u^3_\Delta \in S^3_{\Delta_w}$ defined by (33)-(34) exists, is unique, and satisfies the global error estimates*

$$\|(u - u^3_\Delta)^{(k)}\|_\infty = O(h^{4-k}),\ k = 0, 1, 2, 3, \tag{37}$$

and the local error estimates

$$|(u - u^3_\Delta)'(x)| = O(h^4)\ \textit{for}\ x = s_i\ \textit{and}\ w_i,\ |(u - u^3_\Delta)''(\sigma_{ij})| = O(h^3),\ |(u - u^3_\Delta)'''(w_i)| = O(h^2). \tag{38}$$

3 Adaptive Mesh Generation

In [6], we presented a procedure (algorithm **PlaceMap**) that constructs an appropriate mapping function for an optimal QSC method, given a certain BVP and a certain grid size. Clearly, that procedure can be easily adjusted for the CSC method. It can also be combined with grid size and error estimators, obtained by extrapolation through two runs, in order to solve a problem within a given error tolerance. As mentioned in [6], for the grid size estimator to be reliable, a procedure is needed to gradually advance the grid size, until enough points are taken so that the the solution behaviour is properly captured.

Given the above, we have implemented an adaptive mesh generation technique similar to the one in COLSYS, with a few differences. This technique can be used with CSC or QSC. It is important to note, though, that the CSC method only requires the grid points (and not both the midpoints and grid points), therefore, it does not necessarily need the computation of a mapping function. We also note that the computation of the mapping function for large grid sizes increases the computational cost of the method. We present here an overview of the adaptive technique and elaborate on the differences from the technique in COLSYS.

We first select a grading function $\xi_*(x)$, which denotes the portion of the approximate error up to point x. The grading functions we considered for CSC are $\xi_{c7}(x) = \int_0^x (u^{(4)})^{2/7}dx / \int_0^1 (u^{(4)})^{2/7}dx$ and $\xi_{c9}(x) = \int_0^x (u^{(4)})^{2/9}dx / \int_0^1 (u^{(4)})^{2/9}dx$ (equidistributing the H^1 semi-norm and the H^0 norm of the error according to [5]), $\xi_{c8}(x) = \int_0^x (u^{(4)})^{2/8}dx / \int_0^1 (u^{(4)})^{2/8}dx$ (as [8] suggests) and $\xi_{c2}(x) = \int_0^x \sqrt{1 + (u')^2}dx / \int_0^1 \sqrt{1 + (u')^2}dx$ (same as $\xi_{q2}(x)$ in [6]). Note that a grading function $\xi_*(x)$ is related to a *monitor* function $\hat{u}_*$ (see for example [3]) by $\xi_*(x) = \int_0^x \hat{u}_*dx / \int_0^1 \hat{u}_*dx$. For the approximation of all integrals arising in the algorithm with CSC we use the trapezoidal rule. Note that the approximation to u' is given directly by the derivative of the CSC approximation, while the approximation to $u^{(4)}$ is computed as shown in Theorem 3.

For an mth order BVP, the grading function arising from the monitor function chosen in COLSYS is $\xi_G(x) = \int_0^x (u^{(k+m)})^{1/(k+m)}dx / \int_0^1 (u^{(k+m)})^{1/(k+m)}dx$, which, for Hermite piecewise cubic polynomials ($k = 2$) and second-order BVPs ($m = 2$), is the same as ξ_{c8}.

We now introduce grid size and error estimators. In the following discussion, for convenience, the subscripts $[N]$ and $[2N]$ denote number of subintervals used to compute piecewise polynomial or spline approximations (and *not* values of functions at the respective collocation points).

The grid size estimation in each iteration of the adaptive procedure in COLSYS is done according to a formula developed by theory. The error $\epsilon^G_{[N]}(x) \equiv u^G_{[N]}(x) - u(x)$ in the Gaussian spline collocation approximation $u^G_{[N]}$ with grid size N is proportional to $h^{k+m}u^{(k+m)}$ and to h^{2k} and satisfies $||\epsilon^G_{[N]}||_\infty \leq C(\frac{\theta}{N})^{k+m}(1 + O(h)) + O(h^{2k})$, where $\theta \approx \int_a^b |u^{(k+m)}|^{1/(k+m)}dx$, a and b are the boundary points, and C

is a constant that depends on m and k [3]. For Hermite piecewise cubic polynomials and second-order BVPs, $C = 1/384$. Given a tolerance TOL, COLSYS estimates that it will require a grid of size $NestC = \lceil \theta(\frac{C}{TOL})^{1/(k+m)} \rceil$ to obtain a spline approximation $u^G_{[N]}$ such that the error satisfies $||\epsilon^G_{[N]}(x)||_\infty \leq TOL$. However, for many problems the above grid size estimator often underestimates the required number of points, especially when θ is calculated by composite quadrature based on a small number of points. The problem is partly overcome by toughening the user chosen tolerance by a factor of 10, a fact that we observed in the code of COLSYS. However, even with the tougher tolerance, the number of points is usually underestimated, and it takes several iterations of the adaptive procedure until a reasonable grid size estimation is made. That is, COLSYS gradually advances the mesh, and recomputes θ and $NestC$. Each time the grid size changes and new points are selected, the new points are checked for sufficient error equidistribution. Each time the grid size increases and the equidistribution of error improves, the grid size estimator becomes more reliable. The error estimation and the decision for termination of computation in COLSYS are done by extrapolation through two runs, one of double grid size as the other. Once $u^G_{[N]}$ and $u^G_{[2N]}$ have been computed, the error for any M is estimated by $\epsilon^G_M \approx \frac{u^G_{[2N]} - u^G_{[N]}}{2^\rho - 1}(\frac{2N}{M})^\rho$, where $\rho = k + m$ is the expected order of convergence of $u^G_{[N]}$. Thus, $\epsilon^G_{[2N]} \approx \frac{u^G_{[2N]} - u^G_{[N]}}{2^\rho - 1}$.

We have incorporated similar techniques for grid size and error estimators in our code. Let $\epsilon^3_{[N]} \equiv u^3_\Delta(x) - u(x)$ in the CSC approximation u^3_Δ with grid size N. If we follow COLSYS, the grid size estimator predicts that

$$NestC = \lceil \theta(\frac{C}{TOL})^{1/4} \rceil \tag{39}$$

subintervals are needed, to have $||\epsilon^3_{[NestC]}||_\infty \leq TOL$, where $\theta \approx \int_a^b |u^{(4)}|^{1/4} dx$. Note that we do not have a theoretical derivation for the value of C for CSC, but we calculated it experimentally based on the following arguments. We assume that the error behaves as $\epsilon^3_{[N]} \approx Ch^4 u^{(4)} +$ higher order terms. Consider the simple BVP $u'' = 12x^2$, with $u(0) = 0$ and $u(1) = 1$. This has the solution $u = x^4$, with $u^{(4)} = 24$, which implies that the error behaves as $\epsilon^3_{[N]} \approx 24Ch^4$. By applying CSC on this problem we found that $C = 1/384$. (By similar tests, for QSC, the midpoint error was found to behave as $h^4 u^{(4)}/128$.) However, as in COLSYS, we found that, for many problems, $NestC$ often underestimates the required number of points. (We did not use a factor of 10 to toughen the tolerance.) For the error estimation we use extrapolation as in COLSYS.

The outline of the algorithm, which we refer to as **AdaptSolve1**, is as follows:

Pick$N$ (usually $N = 32$) and initial mesh s_i, $i = 0, \ldots, N$ (usually uniform)
Pick grading function $\xi(x)$ and tolerance TOL
For $itadpt = 1, \ldots, maxstep$ do
    Use CSC on mesh s_i, $i = 0, \ldots, N$, to compute $u_{[N]}$
    Approximate the required derivatives of u on s_i, $i = 0, \ldots, N$
    Calculate $drift = \max_i \{\int_{s_i}^{s_{i+1}} \hat{u} dx\} / (\int_0^1 \hat{u} dx / N)$
    If just_doubled, apply err_est, if $err\_est < TOL$, exit loop, end, end
    Decide whether to redistribute, double, half, or start over
End
If $TOL/4 < err\_est < TOL$ and $drift > 2$, redistribute once, end
Output N, s_i, $i = 0, \ldots, N$, $u_{[N]}$ and err_est

It is important to note that algorithm **AdaptSolve1** includes the construction of the adaptive mesh and the approximate solution to the problem within tolerance TOL. It does not compute a mapping function. (A mapping function may be implicitly defined by the computed location of the grid points, but it is never explicitly computed or used.) Notice also that the drift in this algorithm is calculated in a relative way, as in COLSYS, and not in an absolute way, as in [5] and [6].

The decision whether to redistribute, double, half, or start over with a new set of grid points is taken according to similar criteria as in COLSYS, with the following differences:
(a) COLSYS doubles the grid size if the same N has been used 3 consecutive times, or if $N/2$, N have been used alternatively 3 consecutive times, or if $drift \leq 2$. We also do the same, but we allow one more (fourth) redistribution of the points for the same N before doubling if $1.1 < drift \leq 2$ and $err\_est1/err\_est > err\_est/TOL$, where err_est1 is the first error estimate, err_est the current error estimate. The reason for this difference from COLSYS is that through our experiments we noticed that having a better point distribution helps when we are close to the desired tolerance, while it may worsen the results if we are far from the desired tolerance. Being far from the desired tolerance is usually due to the fact that the grid size is too small, so a proper point distribution cannot occur. Notice that the relation $err\_est1/err\_est > err\_est/TOL$ can be interpreted as being less than half-way away from the desired error tolerance. Thus, we allow a fourth redistribution if the mesh is fairly well but not very well distributed, and if we are "close" to the desired tolerance.
(b) COLSYS computes the error estimate only when the grid is doubled and exits the loop if $err\_est < TOL/10$. We do the same, but we also compute the error estimate if a redistribution has taken place as long as the data for the half grid size have been computed. Thus we have a more updated error estimate. In addition we do not use a factor of 10 to toughen the tolerance, but we do one more redistribution if $TOL/4 < err\_est < TOL$ and $drift > 2$. Through our experiments we found that the (updated) error estimate is in general reliable and does not need toughening, except if the points are not well distributed.
(c) COLSYS starts over (i.e. picks a completely new grid size $N = NestC/2$ and a uniform grid) if $N < NestC < 2N$. We start over if $N < NestC < 2N/1.1$. The reason for the 1.1 factor is that when $NestC$ is too close to $2N$, starting over wastes all the information already computed (i.e. the distribution of points), while at the same time the choice $N = NestC/2$ will only save a very small number of points.

As mentioned above, algorithm **AdaptSolve1** does not compute a mapping. Doubling the grid size means making the current midpoints and grid points new grid points and setting $N = 2N$, just as COLSYS does. A variation of the algorithm implements doubling by computing a mapping function for each new mesh, then computing a double size mesh by mapping the uniform double size mesh to a non-uniform one. We refer to this variation as **AdaptSolve2**. Thus, **AdaptSolve2** for CSC computes a mapping for each new mesh except the last (largest) one. We implemented this algorithm for CSC, but, since the results were very similar to the results of **AdaptSolve1**, and since **AdaptSolve2** involves extra computation, we do not present them.

It is worth mentioning that algorithm **AdaptSolve1** can be easily adjusted for QSC, if we also take into consideration that, for QSC, the location of the collocation points ("midpoints") needs to be computed for each new mesh. Thus, **AdaptSolve1** for QSC computes a mapping function each time a new mesh (including the last one) is computed either by redistribution or by doubling or by choosing a new meshsize. Moreover, since, in the QSC case, the computation of the mapping function is not avoided, the implementation of **AdaptSolve2** does not involve extra computation, compared to **AdaptSolve1**. We have implemented both algorithms for QSC and present some results for comparison.

3.1 An alternative adaptive mesh generation algorithm

Algorithm **AdaptSolve1** (or its variation **AdaptSolve2**) can be used in a more indirect context. More specifically, the algorithm as presented above is used to select a grid size and an appropriate placement of the points, so that the solution computed for that grid reaches a certain tolerance. But, the same algorithm can be used to select an appropriate grid size, on which an effective mapping function is computed. Then, the mapping function and extrapolation can be used to compute the final grid size and placement of points, on which the solution is computed so that it reaches a certain tolerance. This procedure gives rise to an alternative adaptive mesh generation algorithm, which computes an approximation to u within tolerance TOL. We refer to it as **AdaptSolve3** and summarize it as follows:

Apply algorithm **AdaptSolve1** with tolerance $\sqrt{TOL}$
Let $\hat{N}$ be the mesh size selected by **AdaptSolve1**
Apply algorithm **PlaceMap** with $N = \hat{N}$
Let $\hat{s}_i$, $i = 0, \ldots, \hat{N}$, be the grid points, w the mapping function
    and $u_{[\hat{N}]}$ the solution computed by **PlaceMap**
Generate points s_i, $i = 0, \ldots, 2\hat{N}$, by the mapping function w
Compute $u_{[2\hat{N}]}$ on s_i, $i = 0, \ldots, 2\hat{N}$
Use extrapolation between $\hat{N}$ and $2\hat{N}$ to obtain err_est
If $err\_est < TOL$, exit algorithm, end
Use extrapolation between $\hat{N}$ and $2\hat{N}$ to predict N so that $u_{[N]}$ reaches TOL
Generate points s_i, $i = 0, \ldots, N$, by the mapping function w
Compute $u_{[N]}$ on s_i, $i = 0, \ldots, N$
Use extrapolation between $2\hat{N}$ and N to obtain err_est
Output N, s_i, $i = 0, \ldots, N$, $u_{[N]}$ and err_est

Note that **AdaptSolve3** generates a mapping at a relatively small, but problem and tolerance dependent grid size $\hat{N}$. The choice of tolerance $\sqrt{TOL}$ (when running **AdaptSolve1** in order to calculate $\hat{N}$) is supported by our experiments and the following arguments. Using tolerance $\sqrt{TOL}$, we try to balance the trade-off between using too few points to calculate the mapping (as in the case of using $N = 32$ for all problems and tolerances) and using too many points (as in the case of **AdaptSolve2**, where redistributions are applied to all but the last grid). When too few points are used to calculate the mapping the behaviour of the solution may not be captured, while when lots of points are used, the cost increases, and, in addition, our experiments indicate that more points do not necessarily improve the effectiveness of the mapping function, but sometimes degrade it, possibly due to increased computational errors.

Algorithm **AdaptSolve3** can be implemented for CSC and QSC. We present results from this algorithm in the next section. Here we note that, such an algorithm is particularly effective with QSC, since it avoids the computation of mapping functions for large grids.

4 Numerical Results

In this section, we first present numerical results to demonstrate the convergence of the CSC method for BVPs with non-uniform grids, using predefined mappings or adaptively computed mapping functions with the procedure **PlaceMap** described in [6] adjusted for CSC. We then present results to demonstrate the effectiveness of the adaptive mesh generation algorithms presented in Section 3, with CSC and QSC.

All computations in this section were carried out in double precision. The QSC and CSC methods were programmed in MATLAB by us. The linear systems arising were solved by Gauss elimination using the backslash operator or the lu function in MATLAB. We used the MATLAB functions $pchip$ and $ppval$ to construct and evaluate a monotone Hermite piecewise cubic interpolant.

In all tables, the notation $x.y \pm z$ means $x.y \times 10^{\pm z}$. The observed errors of CSC are denoted by ϵ^3. The uniform norm $\|\cdot\|_\infty$ is approximated by the maximum absolute value on a constant grid of 2001 evaluation points, independently of the discretization grid.

We first present results on Problems 1 , 3 , 4 , 5 and 6 in [6]. For convenience, we repeat the definition of the problems here.

PROBLEM 1
$$\exp(x)u'' + \sin(x)u' - \tfrac{1}{2+x}u = g \quad \text{in} \quad (0,1),$$
$$u(0) - u'(0) = \gamma(0), \;\; u(1) + u'(1) = \gamma(1).$$

The function g is chosen so that $u(x) = \sin(x)$ is the solution.

PROBLEM 3
$$\{(1+\eta x)u'\}' = 0 \quad \text{in} \quad (0,1), \quad u(0) = 0, \;\; u(1) = 1.$$

The solution of this problem is $u(x) = \frac{\log(1+\eta x)}{\log(1+\eta)}$.

PROBLEM 4
$$u'' + u' - u = g \quad \text{in} \quad (0,1), \quad u(0) = 0, \;\; u(1) = 0.$$

The function g is chosen so that $u(x) = \frac{\log(1+x\eta)\log(1+(1-x)\eta)}{\log(1+\eta)^2}$ is the solution to the problem.

PROBLEM 5
$$-(1/\nu + \nu(x-\mu)^2)u'' - (2\nu(x-\mu))u' = g \quad \text{in} \quad (0,1), \quad u(0) = 0, \;\; u(1) = 0.$$

The function g is chosen so that the exact solution to this problem is $u(x) = (1-x)(\arctan(\nu(x-\mu)) + \arctan(\nu\mu))$, which for large ν has an interior layer near μ.

PROBLEM 6
$$u'' - \exp(u) = 0 \quad \text{in} \quad (0,1), \;\; u(0) = u(1) = 0.$$

The unique solution to this problem is $u = 2\log(\zeta\sec(\zeta(x-0.5)/2)) - \log 2$, where $\zeta = 1.3360556949061$.

We run CSC with a predefined mapping on Problem 1 in [6]. Table 1 demonstrates the optimal convergence of our method, both global and local, for general variable coefficients differential equations with general mixed boundary conditions, and an "arbitrary" w.

We then present results from experiments with the mapping functions constructed by the adaptive technique of [6], adjusted for CSC. In the adaptive methods, we used $tol = 10^{-2}$, $maxstep = 20$ and a grid size of $N = 32$ to construct w, for each problem, independently of the size of the discretization grid used to solve the problem. Let w_{c2}, w_{c7}, w_{c8}, and w_{c9} be the Hermite interpolant mappings resulting from applying the adaptive algorithm of [6] for CSC to the grading functions ξ_{c2}, ξ_{c7}, ξ_{c8}, and ξ_{c9}, respectively.

CSC	$\|\epsilon^3(x)\|_\infty$		$\vert\epsilon^3(s_i)\vert$		$\vert\epsilon^{3'}(s_i)\vert$		$\vert\epsilon^{3''}(\sigma_{ij})\vert$	
N	error	order	error	order	error	order	error	order
32	3.57-8		3.57-8		3.57-8		2.82-6	
64	2.06-9	4.1	2.06-9	4.1	2.06-9	4.1	3.39-7	3.1
128	1.23-10	4.1	1.23-10	4.1	1.23-10	4.1	4.15-8	3.0
256	7.48-12	4.0	7.48-12	4.0	7.35-12	4.1	5.13-9	3.0

Table 1: Observed errors and respective orders of convergence corresponding to Problem 1 in [6] solved by CSC with mapping function $w(x) = \frac{\exp(x)-1}{\exp(1)-1}$.

CSC	w_e				w_{c8}				w_{c9}			
	$\|\epsilon^3(x)\|_\infty$		$\vert\epsilon^{3'}(s_i)\vert$		$\|\epsilon^3(x)\|_\infty$		$\vert\epsilon^{3'}(s_i)\vert$		$\|\epsilon^3(x)\|_\infty$		$\vert\epsilon^{3'}(s_i)\vert$	
N	error	order	error	order	error	order	error	order	error	order	error	order
32	3.14-3		2.21+1		3.07-3		2.13+1		9.94-3		7.03+1	
64	2.10-4	3.9	1.43+0	3.9	2.04-4	3.9	1.39+0	3.9	5.67-4	4.1	4.45+0	4.0
128	1.34-6	4.0	9.39-2	3.9	1.30-5	4.0	9.32-2	3.9	3.53-5	4.0	3.57-1	3.6
256	8.49-7	4.0	6.12-3	3.9	8.27-7	4.0	6.17-3	3.9	2.22-6	4.0	2.56-2	3.8

Table 2: Observed errors and respective orders of convergence corresponding to Problem 3 in [6] solved by CSC with $\eta = 10,000$ and mapping functions w_e, w_{c8}, and w_{c9}.

The solution of Problem 3 has a boundary layer at the left endpoint. The parameter η controls the sharpness of the boundary layer. Table 2 shows results for CSC and the mapping functions w_{c8}, w_{c9}, and $w_e = ((1+\eta)^x - 1)/\eta$. The adaptive method of [6] for w_{c8} and w_{c9} reached the tolerance in 6 and 7 iterations, respectively. It should be noted that, for this problem, w_e produces errors very close to those of w_{c8}.

The solution of Problem 4 has layers at both boundaries. The adaptive method of [6] applied to either w_{c7} or w_{c8} (for CSC) required 7 iterations to reach the tolerance. Table 3 shows results from the application of CSC to this problem. Notice that the approximate solution arising from w_{c7} has less derivative error than the one arising from w_{c8} while the opposite is true if we look at the function error. This is expected from the definitions of w_{c7} and w_{c8}.

Problem 5 was taken from [5] and its solution has an interior layer. The adaptive method of [6] applied to w_{c7}, w_{c8}, w_{c9} and w_{c2} (for CSC) required 19, 19, 6 and 6 iterations, respectively, to reach the tolerance. Table 4 shows that the approximate solution arising from CSC with the mapping functions approximated by the respective Hermite interpolants have optimal local convergence. It should be noted that the errors from the CSC method are, in general, slightly lower than those from the QSC method presented in [6], something which is expected.

Problem 6 is a non-linear problem which was solved by CSC using the simple iteration scheme $u^{[k+1]''} - u^{[k+1]} = \exp(u^{[k]}) - u^{[k]}$, and Newton's method $u^{[k+1]''} - \exp(u^{[k]})u^{[k+1]} = \exp(u^{[k]}) - \exp(u^{[k]})u^{[k]}$, applied first to the first step of CSC until convergence and then to the second step of CSC

CSC	w_{c7}				w_{c8}			
	$\lvert\epsilon^3(s_i)\rvert$		$\lvert\epsilon^{3'}(s_i)\rvert$		$\lvert\epsilon^3(s_i)\rvert$		$\lvert\epsilon^{3'}(s_i)\rvert$	
N	error	order	error	order	error	order	error	order
32	1.68-2		1.16+1		8.37-3		5.27+1	
64	8.34-4	4.3	6.01-1	4.3	5.64-4	3.9	4.51-0	3.6
128	6.05-5	3.8	2.07-2	4.9	4.00-5	3.8	3.27-1	3.8
256	3.92-6	3.9	1.15-3	3.9	2.67-6	3.9	2.60-2	3.7

Table 3: Observed errors and respective orders of convergence corresponding to Problem 4 in [6] solved by CSC with $\eta = 10,000$ and mapping functions w_{c7} and w_{c8}.

CSC	w_{c7}		w_{c8}		w_{c9}		w_{c2}	
N	error	order	error	order	error	order	error	order
32	1.79-3		1.12-3		5.23-4		2.23-1	
64	6.37-5	4.8	2.87-5	5.1	4.09-5	3.7	7.49-3	4.9
128	3.12-6	4.4	1.28-6	4.5	2.21-6	4.2	4.63-4	4.0
256	1.75-7	4.2	7.06-8	4.2	1.24-7	4.2	2.80-5	4.0

Table 4: Observed grid point errors and respective orders of convergence corresponding to Problem 5 solved in [6] by CSC with $\mu = .5$ and $\nu = 100$, and mapping functions w_{c7}, w_{c8}, w_{c9} and w_{c2}.

until convergence. Table 5 gives the errors of CSC for this problem. We also present the results from HPCC in COLSYS, which uses Newton's method. The relaxation factor for the damping of Newton's iteration in COLSYS was equal to 1, so no damping was used. A uniform grid was good enough for both adaptive procedures with CSC, as well as for HPCC. Since COLSYS always doubles the grid size at least once, and because we did not want to make changes to the COLSYS code, to obtain the results for grid size N, we set the tolerance to something easily reached, and let the code run with starting grid size $N/2$. In all occasions, COLSYS did 2 iterations at grid size $N/2$ and 1 at N. We present the total number of iterations, though we acknowledge that this cannot be used to directly compare HPCC with CSC.

	simple iteration						Newton's						Newton's		
CSC	Step 1			Step 2			Step 1			Step 2			HPCC-COLSYS		
N	$itnl$	error	order	$itnl$	error	order	$itnl$	error	order	$itnl$	error	order	$itnl$	error	order
16	3	3.19-5		3	1.45-8		3	3.20-5		2	1.47-8		3	4.63-8	
32	3	7.95-6	2.01	3	9.81-10	3.88	3	7.99-6	2.00	2	9.86-10	3.90	3	2.95-9	3.97
64	4	2.00-6	1.99	3	6.13-11	4.00	3	2.00-6	2.00	2	6.28-11	3.97	3	1.86-10	3.99
128	4	4.99-7	2.00	3	3.57-12	4.10	3	4.99-7	2.00	2	3.94-12	4.00	3	1.17-11	3.99

Table 5: Observed number of non-linear iterations $itnl$, grid point errors and respective orders of convergence for both steps of CSC and for HPPC on Problem 6.

We now present numerical results from (the mapping-free) CSC integrated with the adaptive procedure **AdaptSolve1** of Section 3 and from HPCC as implemented in COLSYS on Problems 3 , 4 and 5 , as well as two singular perturbation problems taken from [3]. Problems 7 and 8 have a sharp interior layer when ϵ is close to zero.

PROBLEM 7 $$\epsilon u'' + 2xu' = g \quad \text{in} \quad (-1, 1), \quad u(-1) = -1, \;\; u(1) = 1.$$

The exact solution to this problem is $u(x) = \mathrm{erf}(x/\sqrt{\epsilon})$.

PROBLEM 8 $$\epsilon u'' + xu' = g \quad \text{in} \quad (-1, 1), \quad u(-1) = -2, \;\; u(1) = 0.$$

The exact solution to this problem is $u(x) = \cos(\pi x) + \mathrm{erf}(x/\sqrt{2\epsilon})/\mathrm{erf}(1/\sqrt{2\epsilon})$.

For both CSC and HPCC (COLSYS), the tolerance is first set to $TOL = 10^{-6}$, the grading function is ξ_{c8}, and the initial grid has 32 uniform subintervals. Table 6 shows the CSC and HPCC sequences of grid sizes the adaptive algorithms generated. When the grid size remains the same, the method attempts to improve error equidistribution. Note that the error estimator in COLSYS is applied only when the grid is doubled, while in CSC, it is also applied when a new redistribution takes place after doubling.

For all problems and methods the tolerance is successfully reached, and the actual error obtained by CSC and HPCC are about compatible. Taking into account that HPCC uses two Gaussian points per subinterval, on Problems 5 and 7 CSC requires the same number of collocation points as HPCC, and gives about the same error as HPCC; on Problems 4 and 8 CSC requires less collocation points than HPCC, for about the same error; and on Problem 3 CSC requires more collocation points. Overall, CSC is a competitive method with respect to both the number of collocation points and the reliability of the error estimator.

In Figure 1, we plot the exact solution and the non-uniform grids generated by CSC and HPCC for Problems 4 , 7 and 8. The non-uniform grid points (y-coordinates) are plotted versus the respective uniform grid points (x-coordinates). This type of plot, gives a visualization of the mappings w generated by each of the two methods. (We emphasize that the computation of the mappings is not required by CSC and **AdaptSolve1**, but we only compute them in order to visualize them by the plot.) In regions where the graph of w is flat, the non-uniform grid points are very dense (i.e. the uniform points span a large region, and the non-uniform points a small one), while when w rises sharply, the non-uniform grid points are very sparse. We note that in some regions of the domains the mappings of the two methods may differ in a visible way, but in the layer regions, the differences are invisible. In the same figure, we show the location of selected CSC and HPCC grid points. Because the number of points is large, we select to show one grid point for every two of them. It is worth pointing out that, when the solution to some problem exhibits certain symmetry (e.g. Problems 4 , 7), we expect the mapping (and the distribution of points) to reflect that symmetry to some extent. It turns out that COLSYS deviates visibly from the symmetric mapping in Problem 4 and even more visibly in Problem 7. We believe that this may be attributed partly to the too few redistributions that COLSYS applies.

In Table 7, we present brief results on Problems 3 , 4 , 5 , 7 and 8 for more values of the tolerance TOL. The final grid sizes N to reach the estimated errors and the respective actual errors for CSC and HPCC are shown. For CSC, there is one case where the tolerance is missed by a little (Problem 5 , tolerance 10^{-8}), and for HPCC, there are two such cases (Problem 7, tolerances 10^{-5} and 10^{-7}). There are also few

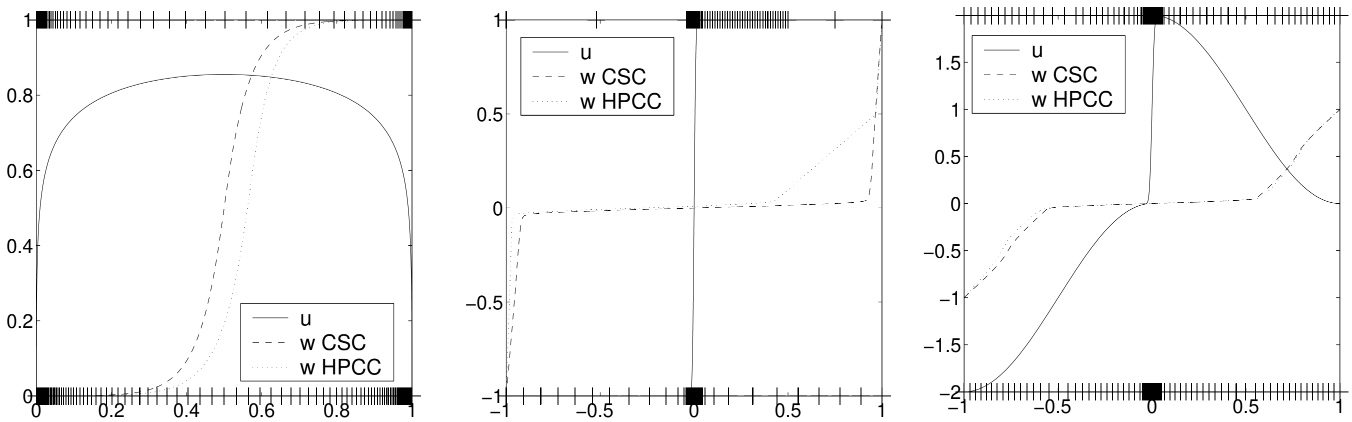

Figure 1: Exact solutions u and non-uniform grids by CSC and HPCC (COLSYS) for Problems 4 , 7 and 8 (from left to right). The location of every second CSC and HPCC grid point is shown by $|$ along the bottom and top axes, respectively.

discrepancies in both methods, in that the grid obtained for a tougher tolerance may be coarser than for a less tough tolerance. However, both methods perform reasonably well. There are about five cases where CSC uses fewer collocation points than HPCC and about four cases where HPCC uses fewer collocation points.

In Table 8, we present results from QSC integrated with algorithms **AdaptSolve1** and **AdaptSolve2**. We emphasize that both algorithms, when implemented with QSC, require the construction of a mapping function for each new mesh, including the last (largest) one. The difference between the two algorithms is that, in **AdaptSolve1** the mapping is used to find only the location of the collocation points ("midpoints"), while in **AdaptSolve2** the mapping is used to find the location of the collocation points, as well as the location of the grid points and midpoints of the double size mesh when doubling. From the experiments, it seems that **AdaptSolve2** for QSC is slightly preferable to **AdaptSolve1**, in that it requires a smaller grid size to reach the tolerance, for some of the cases considered. (A similar comparison of **AdaptSolve1** and **AdaptSolve2** for CSC did not reveal any significant differences.) Both algorithms miss the tolerance by a little bit in few cases, but both algorithms perform satisfactorily. Comparing QSC and CSC, CSC requires a smaller grid size than QSC for several cases (especially for tough tolerances), while there are few cases among those considered where the opposite happens.

In Table 9, we present results from QSC and CSC integrated with algorithm **AdaptSolve3**. This algorithm turns out to be very effective for the cases considered. In several cases, both QSC and CSC with **AdaptSolve3** require smaller grid sizes than the respective methods with **AdaptSolve1** or **AdaptSolve2**. We emphasize that **AdaptSolve3** generates a mapping at a relatively small, but problem and tolerance dependent grid size $\hat{N}$. This extra computation is not substantial compared to the total, therefore **AdaptSolve3** is a competitive alternative to **AdaptSolve1** or **AdaptSolve2**. However, we note that **AdaptSolve3** relies heavily on extrapolation, which, in turn, relies on whether the computed mapping function properly captured the behaviour of the solution. Clearly, there is no theory that guarantees that this is always the case, but, similarly, no theory guarantees that **AdaptSolve1** always computes a solution within the given tolerance. In general, all methods considered performed satisfactorily for the cases considered.

CSC - **AdaptSolve1**									
Probl	N and est. error								act. error
3	32	32	32	64 2.8-3	36	72 1.0-2	36	72 6.8-3	
	36	72 2.7-2	144 3.4-2	72	144 1.6-5	144 1.6-5	288 8.5-7		2.8-7
4	32	32	32	64 2.4-2	62	124 1.3+1	109	218 4.1+0	
	109	218 3.7-3	109	218 1.1-5	218 1.0-5	436 6.9-7			2.3-7
5	32	32	32	64 7.1-5	128 4.2-6	128 5.8-6	256 2.8-7		6.1-7
7	32	64 6.2-2	128 3.8-2	256 7.3-3	128	256 1.1-2	128	256 5.1-3	
	512 3.9-4	256	512 1.8-7						5.6-8
8	32	64 5.4-2	128 2.0-2	128 1.9-2	64	128 2.5-2	128 2.2-2	64	
	128 3.0-2	256 1.4-2	128	256 8.7-4	128	256 2.7-7			5.5-7

HPCC-COLSYS									
Probl	N and est. error								act. error
3	32	32	32	64 9.5-4	51	102 2.0-6	51	102 1.5-7	1.8-7
4	32	32	32	64 2.3-3	64	64	128 5.8-6	256 5.0-7	1.1-7
5	32	32	32	64 8.1-6	128 6.1-7				5.3-7
7	32	64 8.6-2	64	64	128 1.2-5	64	128 2.5-6		
	64	128 1.4-5	256 1.8-7						1.3-7
8	32	32	32	64 8.0-4	62	124 3.7-6	248 2.9-7		2.6-7

Table 6: Sequences of grid sizes N, estimated errors, and actual errors for CSC and HPCC (COLSYS) with a tolerance of 10^{-6} for Problems 3 , 4 , 5 , 7 and 8. The estimated error is given in the form $x.y \pm z$ below the respective grid size. For some problems, the sequence of grid sizes continues over in a second line.

	TOL	1.0-4	1.0-5	1.0-6	1.0-7	1.0-8
CSC - **AdaptSolve1**						
Probl 3	N	128	256	288	512	1024
	act.err.	7.3-6	4.6-7	2.8-7	8.1-8	5.1-9
	est.err.	2.2-5	1.4-6	8.5-7	8.1-8	5.1-9
Probl 4	N	128	256	436	864	2048
	act.err.	8.6-5	5.4-6	2.3-7	4.1-8	6.6-10
	est.err.	6.9-5	5.4-6	6.9-7	4.1-8	1.4-9
Probl 5	N	64	128	256	512	512
	act.err.	3.5-5	2.9-6	6.1-7	1.5-8	1.2-8
	est.err.	6.7-5	3.2-6	2.8-7	1.3-8	1.0-8
Probl 7	N	256	512	512	394	1024
	act.err.	1.2-6	9.4-8	5.6-8	2.9-8	3.0-10
	est.err.	7.1-7	3.9-8	1.8-7	4.1-8	3.3-10
Probl 8	N	146	256	256	500	1024
	act.err.	2.4-6	3.4-7	5.5-7	2.1-8	1.2-9
	est.err.	1.1-4	6.8-7	2.7-7	1.3-8	8.7-10
HPCC-COLSYS						
Probl 3	N	64	116	102	176	256
	act.err.	1.8-6	1.9-7	1.8-7	1.9-8	4.1-9
	est.err.	9.2-6	1.2-6	1.5-7	1.7-8	3.8-9
Probl 4	N	108	100	256	512	564
	act.err.	1.3-5	5.3-6	1.1-7	7.2-9	4.2-9
	est.err.	2.7-5	6.6-6	5.0-7	3.5-8	6.0-9
Probl 5	N	36	64	128	256	512
	act.err.	5.9-5	2.4-6	5.3-7	2.7-8	2.1-9
	est.err.	5.6-5	3.9-6	6.1-7	4.1-8	2.7-9
Probl 7	N	118	116	256	348	544
	act.err.	2.4-7	1.3-5	1.3-7	2.9-7	2.7-9
	est.err.	1.1-6	3.8-6	1.8-7	7.8-8	1.4-8
Probl 8	N	64	140	248	512	1024
	act.err.	2.6-5	1.9-6	2.6-7	1.3-8	8.1-10
	est.err.	4.4-5	2.1-6	2.9-7	2.5-8	1.5-9

Table 7: Grid sizes N, estimated errors, and actual errors for CSC and HPCC of COLSYS, with the indicated tolerances on the indicated problems.

	TOL	1.0-04	1.0-05	1.0-06	1.0-07	1.0-08
QSC - **AdaptSolve1**						
Probl 3	N	128	256	512	1024	1664
	act.err.	7.1-5	2.4-6	1.4-7	9.5-9	1.7-9
	est.err.	8.3-5	4.3-6	2.8-7	2.1-8	3.9-9
Probl 4	N	128	256	512	1536	2976
	act.err.	1.6-4	1.1-5	8.2-7	1.8-8	3.9-9
	est.err.	6.8-5	8.4-6	8.0-7	2.6-8	4.4-9
Probl 5	N	128	512	1024	2048	8192
	act.err.	1.0-5	1.1-6	2.0-7	6.2-8	4.3-9
	est.err.	1.2-5	1.4-6	2.3-7	6.7-8	4.7-9
Probl 7	N	144	256	1024	1024	2048
	act.err.	8.1-6	3.9-7	1.2-7	5.1-8	8.5-9
	est.err.	1.2-5	9.6-6	1.3-7	9.5-8	8.7-9
Probl 8	N	128	432	1184	4096	8192
	act.err.	1.7-5	7.2-6	2.1-7	2.0-8	2.5-9
	est.err.	5.9-5	8.8-6	2.2-7	2.2-8	2.6-9
QSC - **AdaptSolve2**						
Probl 3	N	128	256	512	1024	1664
	act.err.	6.6-5	2.0-6	1.2-7	7.7-9	1.0-9
	est.err.	3.9-5	2.0-6	1.3-7	9.3-9	1.4-9
Probl 4	N	128	256	512	1536	1488
	act.err.	8.4-5	5.8-6	4.4-7	7.2-9	8.8-9
	est.err.	1.0-4	4.3-6	4.0-7	7.2-9	8.7-9
Probl 5	N	64	128	1024	2048	4096
	act.err.	4.4-5	2.7-6	1.2-7	1.2-7	1.1-8
	est.err.	5.0-5	4.9-6	9.7-8	7.5-8	7.8-9
Probl 7	N	144	256	1024	1024	2048
	act.err.	5.1-6	2.3-6	1.3-8	3.0-8	4.3-9
	est.err.	2.1-5	9.8-6	2.5-8	2.2-8	3.0-9
Probl 8	N	128	216	592	4096	8192
	act.err.	1.2-5	7.5-6	9.4-7	1.5-8	1.1-8
	est.err.	2.8-5	6.9-6	7.6-7	1.1-8	6.6-9

Table 8: Grid sizes N, estimated errors, and actual errors for QSC with algorithms **AdaptSolve1** and **AdaptSolve2**, with the indicated tolerances on the indicated problems.

	TOL	1.0-4	1.0-5	1.0-6	1.0-7	1.0-8
QSC - **AdaptSolve3**						
Probl 3	$\hat{N}, N$	32, 81	64, 128	112, 224	200, 400	128, 820
	act.err.	4.52-5	7.17-6	7.83-7	7.78-8	9.72-9
	est.err.	5.36-5	7.22-6	8.39-7	8.86-8	5.19-9
Probl 4	$\hat{N}, N$	32, 115	46, 220	104, 480	128, 910	156,2080
	act.err.	6.46-5	1.25-6	4.56-7	6.59-8	7.36-9
	est.err.	4.83-5	4.97-6	4.63-7	4.18-8	3.94-9
Probl 5	$\hat{N}, N$	32, 64	40, 261	64, 518	128, 807	128,1434
	act.err.	6.79-5	8.25-6	3.74-7	1.60-7	6.92-9
	est.err.	9.10-5	1.63-6	2.32-7	2.66-8	2.81-9
Probl 7	$\hat{N}, N$	58, 116	46, 114	70, 241	70, 428	70, 761
	act.err.	2.28-6	9.79-7	6.18-7	7.12-8	1.00-8
	est.err.	5.48-6	1.45-5	3.38-7	3.58-8	3.60-9
Probl 8	$\hat{N}, N$	90, 180	146, 292	134, 805	128,1200	128,2134
	act.err.	2.24-5	5.32-6	3.26-7	1.11-7	1.30-8
	est.err.	3.61-5	9.66-6	1.86-7	2.56-8	2.53-9
CSC - **AdaptSolve3**						
Probl 3	$\hat{N}, N$	32, 106	64, 185	64, 329	64, 584	64, 1039
	act.err.	4.37-5	4.71-6	4.70-7	4.73-8	4.72-9
	est.err.	4.44-5	4.73-6	4.72-7	4.76-8	4.75-9
Probl 4	$\hat{N}, N$	64, 128	64, 264	64, 469	78, 834	96, 1481
	act.err.	8.59-5	4.73-6	4.74-7	4.75-8	4.71-9
	est.err.	8.69-5	4.75-6	4.77-7	4.77-8	4.80-9
Probl 5	$\hat{N}, N$	32, 64	64, 128	64, 196	64, 347	64, 618
	act.err.	5.42-5	3.10-6	5.71-7	5.86-8	5.84-9
	est.err.	4.36-5	2.60-6	5.63-7	5.71-8	5.68-9
Probl 7	$\hat{N}, N$	132, 264	132, 264	132, 264	214, 428	214, 515
	act.err.	2.21-7	2.21-7	2.21-7	9.26-9	4.61-9
	est.err.	3.68-7	3.68-7	3.68-7	1.01-8	4.51-9
Probl 8	$\hat{N}, N$	128, 256	118, 236	124, 248	124, 386	146, 696
	act.err.	2.18-7	2.68-7	2.38-7	4.07-8	3.61-9
	est.err.	4.91-7	2.98-7	2.81-7	4.01-8	3.60-9

Table 9: Grid sizes $\hat{N}$ (for which the mapping is computed) and N (for which the final solution is computed), estimated errors, and actual errors on N for QSC and CSC with algorithm **AdaptSolve3**, with the indicated tolerances on the indicated problems.

5 Outlook and Future Work

In this paper, we focused the discussion to the one-dimensional BVP but some of the ideas and methods considered are useful for higher order and multi-dimensional BVPs. The extension of the CSC method to multi-dimensional rectangular domains with rectangular discretizations (non-uniform in each dimension) is relatively straightforward. However, the extension of the method to non-rectangular discretizations that allow refinement at certain regions of the domain is not straightforward. We are currently [13] working on two approaches, one that uses skipped rectangular grids (grids that refine the discretization in a tree-like hierarchical manner), and another one that uses moving mesh partial differential equations as in [12].

A discussion on comparing the efficiency of CSC and HPCC is a bit premature at this point, since efficiency issues become more critical when solving multi-dimensional problems, than when solving one-dimensional ones. However, we think it is worth making a few points. In most cases, HPCC of COLSYS requires fewer grid points than CSC to reach a certain error, but it has to collocate on $2N$ points, which results in a double size linear system to solve. Thus, it turns out that in most cases, HPCC requires more collocation points to reach the required tolerance. Moreover, the two-step CSC matrix is tridiagonal, while the HPCC matrix has 4 non-zero entries per row and can be viewed as pentadiagonal. Note that the two-step CSC matrix is solved twice, and if that is done by Gauss elimination, the LU factorization is performed once only. The one-step CSC matrix is pentadiagonal and again half-size as the HPCC matrix.

If we consider two-dimensional problems, the HPCC matrix is four times as large as the two-step CSC matrix, it has about twice the bandwidth of two-step CSC, and 16 nonzero entries per row, while two-step CSC has 9. The one-step CSC matrix has about the same bandwidth as HPCC, but is one quarter size. Moreover, the HPCC matrix requires pivoting, while the CSC matrix (as well as the QSC matrix) in most cases do not. Numerical experiments on some two-dimensional problems with uniform grid discretizations indicate that the error of CSC is about twice the error of HPCC, and that the QSC error is about 6 times the HPCC error. Preliminary timing results indicate that HPCC with Gauss elimination is about 30 times slower than CSC for the same grid size. (Theoretically, the flops count of HPCC is more than 64 times larger than that of CSC, due to pivoting.) Based on these observations we expect that the asymptotic rate of computation time versus error of CSC will be better than that of HPCC.

Comparing QSC in [6] and CSC, the main advantage of CSC is that it requires only the (non-uniform) grid points, while QSC requires both the grid points and the midpoints. Thus CSC does not require the computation of a mapping function. This allows the efficient application of the adaptive technique **AdaptSolve1** of Section 3 which applies redistributions of the points at the large grids. However, the adaptive technique **AdaptSolve3**, which applies redistributions of the points at relatively small (or medium) grid sizes, seems to be a competitive alternative with both CSC and QSC. Moreover, as mentioned in [6], QSC may be advantageous for certain problems, since it does not apply the differential operator on the boundary (or any other grid) points, thus avoiding potential singularities.

References

[1] D. Archer. An $O(h^4)$ cubic spline collocation method for quasilinear parabolic equations. *SIAM J. Numer. Anal.*, 14(4):620–637, 1977.

[2] U. Ascher, J. Christiansen, and R. D. Russell. A collocation solver for mixed order systems of boundary value problems. *Math. Comp.*, 33(146):659–679, 1979.

[3] U. M. Ascher, R. M. Mattheij, and R. D. Russell. *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*. SIAM, 1995.

[4] G. Bader and U. Ascher. A new basis implementation for a mixed order boundary value ”ode” solver. *SIAM J. Sci. Stat. Comp.*, 8(4):483–500, 1987.

[5] G. F. Carey and H. T. Dinh. Grading functions and mesh redistribution. *SIAM J. Numer. Anal.*, 22(5):1028–1040, 1985.

[6] C. C. Christara and K. S. Ng. Optimal quadratic spline collocation on non-uniform partitions. Submitted to *Computing*, 2003.

[7] J. W. Daniel and B. K. Swartz. Extrapolated collocation for two-point boundary-value problems using cubic splines. *J. Inst. Maths Applics*, 16:161–174, 1975.

[8] C. de Boor. Good approximation by splines with variable knots II. *Lecture notes in Mathematics*, 363:12–20, 1973. Conference on Numerical Solution of Differential Equations.

[9] C. de Boor and B. Swartz. Collocation at Gaussian points. *SIAM J. Numer. Anal.*, 10(4):582–606, 1973.

[10] D. J. Fyfe. The use of cubic splines in the solution of two-point boundary value problems. *Comput. J.*, 12:188–192, 1969.

[11] E. N. Houstis, E. A. Vavalis, and J. R. Rice. Convergence of an $O(h^4)$ cubic spline collocation method for elliptic partial differential equations. *SIAM J. Numer. Anal.*, 25(1):54–74, 1988.

[12] W. Huang and R. D. Russell. Moving mesh strategy based on a gradient flow equation for two-dimensional problems. *SIAM J. Sci. Comput.*, 20:998–1015, 1999.

[13] K. S. Ng. *Spline Collocation on Adaptive Grids and Non-Rectangular Regions*. PhD thesis, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, 2005.