COLLOCATION METHODS FOR CONTINUATION PROBLEMS IN NONLINEAR ELLIPTIC PDEs

Eusebius Doedel and Hamid Sharifi
Department of Computer Science Concordia University Montreal, Canada

Summary

A new class of collocation methods for nonlinear elliptic partial differential equations is described in the context of numerical continuation studies. It is shown how the methods are well-suited for a nested dissection solution algorithm, thereby reducing computational complexity. Numerical results are given to illustrate the accuracy of the methods.

1 Introduction

Piecewise polynomial collocation is widely used for solving boundary value problems in ordinary differential equations (ODEs). For example, it is the basic discretization in the software packages COLSYS [2], COLDAE [3] , and AUTO [8]. Its advantages are high accuracy [7], known mesh adaption strategies [14] , and efficient solution procedures for the associated linear systems [1]. For difficult problems, collocation has been the method of choice; see [9] for some representative applications.

Use of piecewise polynomial collocation has not been very widespread for partial differential equations (PDEs). A direct extension to elliptic PDEs was given in [13] ; see also [4, 5, 6, 11, 15, 16].

In this paper we describe an extended class of collocation methods, which we have implemented in an experimental software, written in C^{++}, for the numerical continuation of solutions to systems of nonlinear elliptic partial differential equations on the unit square in R^2. For simplicity of presentation, we here consider the scalar parameter-dependent nonlinear elliptic PDE

$$Nu(x) \equiv \Delta u(x) + f(u(x), \lambda) = 0, \quad \text{for} \quad x \in \Omega, \tag{1.1}$$

$$u(x) = 0, \quad \text{for} \quad x \in \partial\Omega, \tag{1.2}$$

where Δ is the Laplace operator, Ω the unit square in R^2, $\lambda \in R$, and $x = (x_1, x_2)^*$, where * denotes transpose.

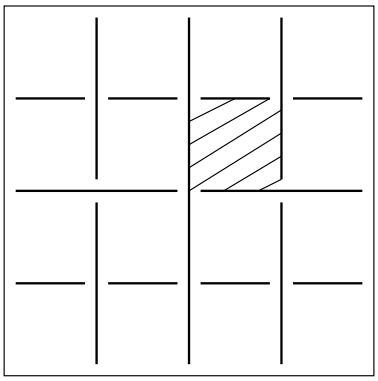

Figure 1 *The domain Ω and its recursive subdivision.*

Our main interest is continuation software for parameter-dependent problems. Therefore, in addition to $u(\cdot)$, we also treat λ as unknown, and we add an integral constraint

$$\int_\Omega q(x, u(x), \lambda) = 0, \tag{1.3}$$

where $q(\cdot) \in R$. This problem formulation makes it possible to use Keller's pseudo-arclength continuation method [12].

In Section 2 we describe the collocation method in detail for the equations above. In Section 3 it is shown how the discrete set of linear equations that arise from Newton's method can be solved by nested dissection, which reduces computational cost. Numerical results that illustrate the accuracy of the methods are given in Section 5.

The collocation methods and nested dissection algorithm generalize to problems with $x \in R^n$. Furthermore, the domain need not be a square, and more general elements are possible. More general boundary conditions and multiple integral constraints can also be treated and have, in fact, been implemented in our experimental software. However, in order to keep the technical presentation simple, we restrict attention in this paper to the problem (1.1-1.3) above.

2 A New Class of Collocation Methods

2.1 Description of the methods

Recursively subdivide the square Ω into 2^M "finite elements", as in Figure 1. To each finite element associate appropriate boundary *matching points* x_i, $i =$

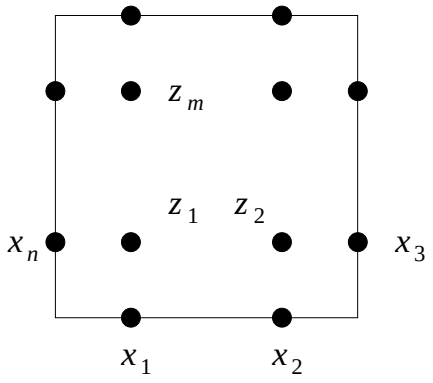

Figure 2 *A finite element of* Ω.

$1, \cdots, n$ and interior *collocation points* z_i, $i = 1, \cdots, m$. See Figure 2 for an example of the choice of these points. (The two sets of matching points on the common boundary of two adjacent elements must coincide.) Also associate a polynomial $p(x) \in P_{n+m}$ to each element. Here P_{n+m} is an appropriate $(n+m)$-dimensional polynomial space. At the matching points the values of the neighboring polynomials (u_i) are required to match. The normal derivatives of the neighboring polynomials (v_i) are also required to match at these points. Further require each polynomial to satisfy the *collocation* equations

$$Np(z_i) \equiv \Delta p(z_i) + f\Big(p(z_i), \lambda\Big) = 0, \qquad i = 1, \cdots, m. \tag{2.4}$$

(For notational simplicity we suppress indices that denote the element.) Finally, discrete, boundary conditions are imposed at matching points x_i on the domain boundary $\partial\Omega$. For each finite element the local polynomial has the form $p(x) = \sum_{i=1}^{n+m} c_i \phi_i(x)$, where $Span\Big\{\phi_1, \cdots, \phi_{n+m}\Big\} = P_{n+m}$. Then the collocation equations are

$$Np(z_k) \equiv \sum_{i=1}^{n+m} c_i \Delta\phi_i(z_k) + f\Big(\sum_{i=1}^{n+m} c_i \phi_i(z_k), \lambda\Big) = 0, \qquad k = 1, \cdots, m. \tag{2.5}$$

To impose the continuity requirements, associate unique variables u_i and v_i to each matching point x_i, and require that $p(x_i) = u_i$ and $\nabla p(x_i)^* \eta_i = v_i$. Let $u = (u_1, \cdots, u_n)^*$, $v = (v_1, \cdots, v_n)^*$, and

$$\Phi = \begin{pmatrix} \phi_1(x_1) & \cdots & \phi_1(x_n) \\ : & & : \\ \phi_{n+m}(x_1) & \cdots & \phi_{n+m}(x_n) \end{pmatrix},$$

$$R_\Phi = \begin{pmatrix} \nabla\phi_1(x_1)^*\eta_1 & \cdots & \nabla\phi_1(x_n)^*\eta_n \\ \vdots & & \vdots \\ \nabla\phi_{n+m}(x_1)^*\eta_1 & \cdots & \nabla\phi_{n+m}(x_n)^*\eta_n \end{pmatrix}.$$

Then we can write

$$u - \Phi^* c = 0, \tag{2.6}$$

$$v - R_\Phi^* c = 0. \tag{2.7}$$

The integral constraint (1.3) is applied to the union of all local polynomials. This gives

$$\sum_{\ell=1}^{2^M} \int_{\Omega_\ell} q\Big(x, p(x), \lambda\Big) dx = 0,$$

where Ω_ℓ denotes the ℓth finite element of Ω. Integrate over each element by an appropriate quadrature formula that uses the collocation points z_k, with weights ω_k :

$$\sum_{\ell=1}^{2^M} \sum_{k=1}^{m} \omega_k q\Big(z_k, p(z_k), \lambda\Big) = 0. \tag{2.8}$$

2.2 Newton's Method

The equations (2.5), (2.6), (2.7) and (2.8), together with the discrete boundary conditions, constitute the discretization. The unknowns are $c \in R^{n+m}$ for each finite element, the u_i and v_i associated with the points x_i on the inter-element boundaries, and λ. To solve these equations we use Newton's method. Omitting iteration indices, it can be written as

$$L_\Phi^*\ \delta c + \delta\lambda\ f_\lambda = -r_N, \tag{2.9}$$

$$\delta u - \Phi^* \delta c = -r_u, \tag{2.10}$$

$$\delta v - R_\Phi^* \delta c = -r_v. \tag{2.11}$$

and

$$\sum_{\ell=1}^{2^M} \Big\{ q_u^*\ \delta c \Big\} + \delta\lambda\ q_\lambda = -r_q. \tag{2.12}$$

Above

$$L_\Phi^* = \begin{pmatrix} L\Big[p(z_1)\Big]\phi_1(z_1) & \cdots & L\Big[p(z_1)\Big]\phi_{n+m}(z_1) \\ \vdots & & \vdots \\ L\Big[p(z_m)\Big]\phi_1(z_m) & \cdots & L\Big[p(z_m)\Big]\phi_{n+m}(z_m) \end{pmatrix}, f_\lambda = \begin{pmatrix} D_2 f\Big(p(z_1), \lambda\Big) \\ \vdots \\ D_2 f\Big(p(z_m), \lambda\Big) \end{pmatrix}$$

where L is the linearization of N, i.e., $L[p(z)]\phi(z)$ is the linearization of N about p acting on ϕ at z, or, more precisely,

$$L[p(z)]\phi(z) = \Delta\phi(z) + D_1 f\Big(p(z), \lambda\Big)\phi(z).$$

Further we have defined

$$\delta c = \begin{pmatrix} \delta c_1 \\ : \\ \delta c_{n+m} \end{pmatrix}, \quad r_N = \begin{pmatrix} Np(z_1) \\ : \\ Np(z_m) \end{pmatrix}, \quad \delta u = \begin{pmatrix} \delta u_1 \\ : \\ \delta u_n \end{pmatrix}, \quad \delta v = \begin{pmatrix} \delta v_1 \\ : \\ \delta v_n \end{pmatrix},$$

$$r_u = u - \Phi^* c, \qquad r_v = v - R_\Phi^* c,$$

and

$$q_u = \begin{pmatrix} \sum_{k=1}^m \omega_k D_2 q\Big(z_k, p(z_k), \lambda\Big)\phi_1(z_k) \\ : \\ \sum_{k=1}^m \omega_k D_2 q\Big(z_k, p(z_k), \lambda\Big)\phi_{n+m}(z_k) \end{pmatrix},$$

$$q_\lambda = \sum_{\ell=1}^{2^M}\sum_{k=1}^m \omega_k D_3 q\Big(z_k, p(z_k), \lambda\Big), \qquad r_q = \sum_{\ell=1}^{2^M}\sum_{k=1}^m \omega_k q\Big(z_k, p(z_k), \lambda\Big).$$

2.3 Elimination of δc

Equations (2.9) and (2.10) can be written

$$\boxed{\begin{pmatrix} \Phi^* \\ L_\Phi^* \end{pmatrix} \delta c = \begin{pmatrix} \delta u + r_u \\ -r_N - f_\lambda \delta\lambda \end{pmatrix}.} \tag{2.13}$$

Using (2.13) to eliminate δc in (2.11) one obtains

$$\delta v = R_\Phi^* \begin{pmatrix} \Phi^* \\ L_\Phi^* \end{pmatrix}^{-1} \begin{pmatrix} \delta u + r_u \\ -r_N - f_\lambda \delta\lambda \end{pmatrix} - r_v.$$

Define A and B as

$$\boxed{(\ \Phi \mid L_\Phi\ ) \begin{pmatrix} A^* \\ B^* \end{pmatrix} = R_\Phi.} \tag{2.14}$$

Then the above expression for δv can be rewritten as

$$\delta v = (\ A \quad B\ ) \begin{pmatrix} \Phi^* \\ L_\Phi^* \end{pmatrix} \begin{pmatrix} \Phi^* \\ L_\Phi^* \end{pmatrix}^{-1} \begin{pmatrix} \delta u + r_u \\ -r_N - f_\lambda \delta\lambda \end{pmatrix} - r_v,$$

that is,

$$\delta v = A\delta u - \delta\lambda B\ f_\lambda - B r_N - r_v + A r_u.$$

This equation is of the form

$$\boxed{\delta v = A\delta u + \delta\lambda\ b + r,} \tag{2.15}$$

where we have defined

$$r = -Br_N - r_v + Ar_u, \qquad b = -Bf_\lambda.$$

Next, using (2.13) in (2.12) gives

$$\sum_{\ell=1}^{2^M}\Big\{ q_u^* \begin{pmatrix} \Phi^* \\ L_\Phi^* \end{pmatrix}^{-1} \begin{pmatrix} \delta u + r_u \\ -r_N - f_\lambda \delta\lambda \end{pmatrix}\Big\} + \delta\lambda\ q_\lambda = -r_q. \tag{2.16}$$

Let $d \in R^n$, $\hat{d} \in R^m$ be the solution of

$$\boxed{(\,\Phi \mid L_\Phi\,) \begin{pmatrix} d \\ \hat{d} \end{pmatrix} = q_u.} \tag{2.17}$$

Then (2.16) can be written

$$\sum_{\ell=1}^{2^M}\Big\{ d^*(\delta u + r_u) + \hat{d}^*(-r_N - f_\lambda\delta\lambda)\Big\} + \delta\lambda\ q_\lambda = -r_q,$$

which is of the form

$$\boxed{\textstyle\sum_{\ell=1}^{2^M}\{d^*\delta u\} + e\ \delta\lambda = s,} \tag{2.18}$$

where

$$e = q_\lambda + \sum_{\ell=1}^{2^M}\Big\{-\hat{d}^* f_\lambda\Big\}, \qquad s = -r_q - \sum_{\ell=1}^{2^M}\Big\{ d^* r_u - \hat{d}^* r_N\Big\}.$$

Finally, the Newton equations for the boundary conditions are of the form

$$\boxed{\delta u = -u,} \tag{2.19}$$

to be applied only at the matching points on the exterior boundary $\partial\Omega$. One can, of course, exclude the boundary unknowns in case of explicit boundary conditions for u. However, we include (2.19) in the description of the solution algorithm, in order to suggest the treatment of the more general boundary conditions that can be handled by our prototype implementation.

2.4 General Algorithm

Given current approximations to u, v, c, and λ, a complete Newton iteration now consists of the following steps :

(1) For each finite element use (2.14) to compute the matrices A and B, and use (2.17) to compute d and $\hat{d}$.

(2) Solve the global set of equations (2.15), (2.18), (2.19) for δu, δv and $\delta\lambda$. This can be done by the nested dissection algorithm described in the following section.

(3) For each finite element compute δc using (2.13).

(4) Update : $u \to u + \delta u$, $v \to v + \delta v$, $c \to c + \delta c$, and $\lambda \to \lambda + \delta\lambda$.

Note that, for any given finite element, the matrix on the left hand side of (2.13) is the transpose of the matrix on the left hand side of (2.14) and (2.17). Thus only one LU-decomposition is required per finite element.

3 Nested Dissection

Here we show how to to apply the method of nested dissection to the solution of the global set of equations (2.15) and (2.18). The algorithm consists of backward recursive elimination of the unknowns δu and δv on boundaries separating adjacent regions. Here "adjacent" means that the two regions resulted from subdivision of the larger region formed by their union, i.e., the two regions correspond to the descendant nodes of a common parent node in the recursion tree. By "backward recursion" is meant that the elimination starts at the leaves and terminates at the root of the tree.

This procedure results in the elimination of all the δu's and δv's in the interior of the domain Ω. One is left with equations of the form (2.15) corresponding to the matching point on the exterior boundary $\partial\Omega$. The discrete boundary conditions can be then used to determine the values of δu, δv, and $\delta\lambda$ at the x_i on $\partial\Omega$. Thereafter a recursive backsubstitution gives the values of δu (and hence δv) on each interior boundary.

To make the above more precise, consider two adjacent regions Ω_1 and Ω_2 as in Figure 3. These regions need not be finite elements, i.e., they need not correspond to leaves in the recursion tree. The elimination of the unknowns δu and δv on the common boundary is now done as follows. Call the common boundary $\partial\Omega_{12}$, and let $\partial\Omega_1$ and $\partial\Omega_2$ denote the remaining parts of the boundaries of Ω_1 and Ω_2, respectively. The variable δu in (2.15) and (2.18) is split accordingly into δu_{12} and δu_1 for region Ω_1 and δu_{21} $(= \delta u_{12})$ and δu_2 for region Ω_1. The variable δv and the coefficients and inhomogeneous terms are split in a similar fashion.

Thus, for region Ω_1 the equation (2.15) can be written in the form

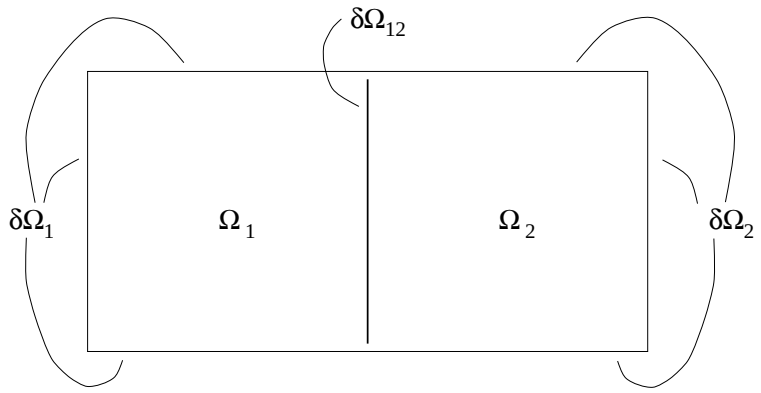

Figure 3 *Two adjacent regions.*

$$\delta v_1 \ = A_1\delta u_1 + A_{12}\delta u_{12} + \delta\lambda b_1 + r_1, \tag{3.20}$$

$$\delta v_{12} = B_1\delta u_1 + B_{12}\delta u_{12} + \delta\lambda b_{12} + r_{12}, \tag{3.21}$$

while for Ω_2 they have the form

$$\delta v_2 \ = A_2\delta u_2 + A_{21}\delta u_{21} + \delta\lambda b_2 + r_2, \tag{3.22}$$

$$\delta v_{21} = B_2\delta u_2 + B_{21}\delta u_{21} + \delta\lambda b_{21} + r_{21}. \tag{3.23}$$

By continuity one has

$$\delta u_{12} = \delta u_{21}, \tag{3.24}$$

$$\delta v_{12} = -\delta v_{21}. \tag{3.25}$$

From (3.21) and (3.23), using (3.24) and (3.25), it follows that

$$\delta u_{12} = -\hat{B}^{-1}\Big\{B_1\delta u_1 + B_2\delta u_2 + \delta\lambda(b_{12}+b_{21}) + r_{12} + r_{21}\Big\}, \tag{3.26}$$

where we have defined

$$\hat{B} \equiv B_{12} + B_{21}.$$

Substituting this into (3.20) and (3.22), again using (3.24), one obtains

$$\delta v_1 = A_1\delta u_1 - A_{12}\hat{B}^{-1}\Big\{B_1\delta u_1 + B_2\delta u_2 + \delta\lambda(b_{12}+b_{21}) + r_{12} + r_{21}\Big\} + \delta\lambda b_1 + r_1,$$

$$\delta v_2 = A_2\delta u_2 - A_{21}\hat{B}^{-1}\Big\{B_1\delta u_1 + B_2\delta u_2 + \delta\lambda(b_{12}+b_{21}) + r_{12} + r_{21}\Big\} + \delta\lambda b_2 + r_2,$$

which is of the form

$$\delta v_1 = \hat{A}_{11}\delta u_1 + \hat{A}_{12}\delta u_2 + \delta\lambda\hat{b}_1 + \hat{r}_1, \tag{3.27}$$
$$\delta v_2 = \hat{A}_{21}\delta u_1 + \hat{A}_{22}\delta u_2 + \delta\lambda\hat{b}_2 + \hat{r}_2, \tag{3.28}$$

where

$$\hat{A}_{11} = A_1 - A_{12}\hat{B}^{-1}B_1, \qquad \hat{A}_{12} = -A_{12}\hat{B}^{-1}B_2,$$
$$\hat{A}_{21} = -A_{21}\hat{B}^{-1}B_1, \qquad \hat{A}_{22} = A_2 - A_{21}\hat{B}^{-1}B_2,$$

and

$$\hat{r}_1 = r_1 - A_{12}\hat{B}^{-1}(r_{12} + r_{21}), \qquad \hat{b}_1 = b_1 - A_{12}\hat{B}^{-1}(b_{12} + b_{21}),$$
$$\hat{r}_2 = r_2 - A_{21}\hat{B}^{-1}(r_{12} + r_{21}), \qquad \hat{b}_2 = b_2 - A_{21}\hat{B}^{-1}(b_{12} + b_{21}).$$

The equations (3.27) and (3.28) represents the discrete equations for the enlarged region, that is, for the union of Ω_1 and Ω_2, after elimination of the common boundary unknowns δu_{12} and δv_{12}. Note that these new equations are again of the form (2.15).

Using (3.26) one can also eliminate the unknown δu's in the interior of Ω from the discrete integral constraint (2.18). Consider all adjacent regions Ω_1 and Ω_2 at the Mth level in the recursion tree, and pairwise combine their contribution to the total sum. Then (2.18) can be written

$$\sum_{\ell=1}^{2^{M-1}} \{d_1^*\delta u_1 + d_{12}^*\delta u_{12} + d_{21}^*\delta u_{21} + d_2^*\delta u_2\} + e\ \delta\lambda = s, \tag{3.29}$$

where the splitting of d and δu is done as before. Using (3.26) and (3.24), rewrite (3.29) as

$$\sum_{\ell=1}^{2^{M-1}} \Big\{ \big[d_1^* - (d_{12} + d_{21})^*\hat{B}^{-1}B_1\big]\delta u_1 + \big[d_2^* - (d_{12} + d_{21})^*\hat{B}^{-1}B_2\big]\delta u_2 \Big\} +$$

$$\big[e - \sum_{\ell=1}^{2^{M-1}} (d_{12} + d_{21})^*\hat{B}^{-1}(b_{12} + b_{21})\big]\delta\lambda = s + \sum_{\ell=1}^{2^{M-1}} (d_{12} + d_{21})^*\hat{B}^{-1}(r_{12} + r_{21}),$$

which is of the form

$$\sum_{\ell=1}^{2^{M-1}} \{\hat{d}_1^*\delta u_1 + \hat{d}_2^*\delta u_2\} + \hat{e}\ \delta\lambda = \hat{s}, \tag{3.30}$$

where

$$\hat{d}_1^* = d_1^* - (d_{12} + d_{21})^*\hat{B}^{-1}B_1, \qquad \hat{d}_2^* = d_2^* - (d_{12} + d_{21})^*\hat{B}^{-1}B_2,$$

$$\hat{e} = e - \sum_{\ell=1}^{2^{M-1}} (d_{12}+d_{21})^* \hat{B}^{-1}(b_{12}+b_{21}), \qquad \hat{s} = s + \sum_{\ell=1}^{2^{M-1}} (d_{12}+d_{21})^* \hat{B}^{-1}(r_{12}+r_{21}).$$

Note that Equation (3.30) is still of the same form as Equation (2.18), but the sum is now over half the number of regions, each of which is the union of two descendant regions in the recursive division of Ω.

The procedure is repeated recursively, and in synchrony with the eliminations that yield Equations (3.27) and (3.28). The final stage results in equations of the form (2.15), (2.18) (with $M = 0$), and (2.19), corresponding to the exterior boundary $\partial\Omega$. These equations can be solved for for $\delta\lambda$ and for δu and δv on $\partial\Omega$. A recursive backsubstitution process then gives the values of δu and δv in the interior of Ω.

The number of arithmetic operations required by the above nested dissection method can be shown to be $O(K^3)$ for a K by K mesh (with K^2 elements). This compares favorably with the $O(K^4)$ required by a standard band-solver. For details see [10].

4 Continuation

The problem formulation in Section 1 includes the integral constraint (1.3), which was also taken into account in the discretization and solution procedures in Sections 2-3. This makes it possible to use the pseudo-arclength continuation method [12] for following solution families for varying λ. In continuous form, the pseudo-arclength constraint is given by

$$\int_\Omega \Big(u(x) - u_0(x)\Big)\dot{u}_0(x)\ dx + (\lambda - \lambda_0)\dot{\lambda} - \Delta s = 0,$$

where $(u_0(\cdot), \lambda_0)$ denotes a given solution, and where $(\dot{u}_0(\cdot), \dot{\lambda}_0)$ is the direction vector of the solution branch at $(u_0(\cdot), \lambda_0)$, i.e., the normalized rate of change of the solution $(u(\cdot), \lambda)$ with respect to Δs.

Since Ω is the unit square, with area 1, one can rewrite the continuous pseudo-arclength equations as

$$\int_\Omega \Big(\Big(u(x) - u_0(x)\Big)\dot{u}_0(x) + (\lambda - \lambda_0)\dot{\lambda} - \Delta s\Big)\ dx = 0,$$

which is of the form (1.3), with

$$q(x, u(x), \lambda) = \Big(u(x) - u_0(x)\Big)\dot{u}_0(x) + (\lambda - \lambda_0)\dot{\lambda}_0 - \Delta s.$$

The discretization and solution procedure is now as described in the preceding sections. In particular, the discretized integral constraint (2.8) becomes

$$\sum_{\ell=1}^{2^M}\sum_{k=1}^{m}\Big\{\omega_k\Big(p(z_k)-p_0(z_k)\Big)\dot{p}_0(z_k)\Big\}+(\lambda-\lambda_0)\dot{\lambda}_0-\Delta s=0,$$

and the coefficients in the Newton equation (2.12) are now given by

$$q_u=\begin{pmatrix}\sum_{k=1}^{m}\omega_k\dot{p}_0(z_k)\phi_1(z_k)\\ \vdots\\ \sum_{k=1}^{m}\omega_k\dot{p}_0(z_k)\phi_{n+m}(z_k)\end{pmatrix},\qquad q_\lambda=\dot{\lambda}_0, \tag{4.31}$$

and

$$r_q=\sum_{\ell=1}^{2^M}\Big\{\sum_{k=1}^{m}\omega_k\Big(p(x)-p_0(z_k)\Big)\dot{p}_0(z_k)\Big\}+(\lambda-\lambda_0)\dot{\lambda}_0-\Delta s.$$

Above, p_0 denotes the restriction of a given piecewise polynomial solution to the ℓth element, and $\dot{p}_0$ is its derivative with respect to Δs. We can write $\dot{p}(x)=\sum_{i=1}^{n+m}\dot{c}_i\phi_i(x)$. The coefficients $\dot{c}_i$ can be determined by one extra back solve in the solution process. To see this, note that differentiating Equations (2.5), (2.6), (2.7), and (2.8) with respect to Δs gives

$$L_\Phi^*\ \dot{c}+\dot{\lambda}\ f_\lambda=0,$$

$$\dot{u}-\Phi^*\dot{c}=0,$$

$$\dot{v}-R_\Phi^*\dot{c}=0.$$

and

$$\sum_{\ell=1}^{2^M}\Big\{q_u^*\ \dot{c}\Big\}+\dot{\lambda}\ q_\lambda=1,$$

i.e., the same left hand sides as in Equations (2.9)-(2.12), with q_u and q_λ given by (4.31). Thus the linear equation solution procedure of Sections 2.4 and 3 can be used to compute the $\dot{u}$, $\dot{v}$, and $\dot{\lambda}$. Thereafter the $\dot{c}$ can be computed from

$$\begin{pmatrix}\Phi^*\\ L_\Phi^*\end{pmatrix}\dot{c}=\begin{pmatrix}\dot{u}\\ -f_\lambda\dot{\lambda}\end{pmatrix}.$$

nb.col. *mesh*	1 *max. error* (*o*)	$2*2$ *max. error* (*o*)	$3*3$ *max. error* (*o*)	$4*4$ *max. error* (*o*)
$2*2$	$8.84e^{-3}$	$4.54e^{-4}$	$2.09e^{-5}$	$4.04e^{-7}$
$4*4$	$2.80e^{-3}$ (1.66)	$5.18e^{-5}$ (3.13)	$7.64e^{-7}$ (4.78)	$7.40e^{-9}$ (5.77)
$8*8$	$7.21e^{-4}$ (1.96)	$4.30e^{-6}$ (3.59)	$2.72e^{-8}$ (4.81)	$1.32e^{-10}$ (5.81)
$16*16$	$1.81e^{-4}$ (1.99)	$3.02e^{-7}$ (3.83)	$8.84e^{-10}$ (4.95)	$2.14e^{-12}$ (5.95)
$32*32$	$4.54e^{-5}$ (2.00)	$1.99e^{-8}$ (3.92)	$2.81e^{-11}$ (4.97)	$3.40e^{-14}$ (5.98)
$64*64$	$1.14e^{-5}$ (2.00)	$1.28e^{-9}$ (3.96)		

Table 1 Maximum error and order of convergence at the matching points.

5 Numerical Results

First, consider the equation

$$\Delta u = (2x^2y^2 + 2x^2y + 2xy^2 - 6xy)e^{x+y} \quad \text{for} \quad (x,y) \in \Omega \subset R^2,$$

$$u = 0 \quad \text{on} \quad \partial\Omega,$$

where Ω is the unit square. The analytical solution is

$$u(x,y) = x(x-1)y(y-1)e^{x+y}.$$

Table 1 presents the maximum error at the matching points for different mesh sizes and number of collocation points. The collocation and matching points are chosen to be Gauss points. The convergence rates between consecutive items in each column are shown in parentheses.

Finally, Table 2 and Table 3 show the location and the maximum error of the fold in Bratu's problem

$$\Delta u + \lambda e^u = 0 \quad \text{for} \quad (x,y) \in \Omega,$$

$$u = 0 \quad \text{on} \quad \partial\Omega,$$

where Ω is the unit square, for various choices of the number of mesh and collocation points (The exact location of the fold is assumed to be at $\lambda = 6.808124423$.)

Bibliography

[1] Ascher, U. M. and Mattheij, R. M. M. and Russell, R. D., "Numerical solution of boundary value problems for ordinary differential equations", Prentice-Hall, 1988.

nb.col. *mesh*	1 λ	$2*2$ λ	$3*3$ λ	$4*4$ λ
$1*1$	5.886071059	8.829106588	6.468997582	6.896291085
$2*2$	7.848094745	6.657615709	6.819161991	6.807812085
$4*4$	6.861418190	6.805822324	6.808174115	6.80812296
$8*8$	6.820207335	6.808016922	6.808124690	6.808124424
$16*16$	6.811079454	6.808117899	6.808124427	6.808124423
$32*32$	6.808859009	6.808124018	6.808124423	6.808124423
$64*64$	6.808307808	6.808124397		

Table 2 The location of the fold in Bratu's problem.

nb.col. *mesh*	1 *max. error* (o)	$2*2$ *max. error* (o)	$3*3$ *max. error* (o)	$4*4$ *max. error* (o)
$1*1$	$9.22e^{-1}$	2.02	$3.39e^{-1}$	$8.82e^{-2}$
$2*2$	1.04 (0.17)	$1.51e^{-1}$ (3.75)	$1.10e^{-2}$ (4.94)	$3.12e^{-4}$ (8.14)
$4*4$	$5.33e^{-2}$ (4.29)	$2.30e^{-3}$ (6.03)	$4.97e^{-5}$ (7.80)	$1.46e^{-6}$ (7.74)
$8*8$	$1.21e^{-2}$ (2.14)	$1.08e^{-4}$ (4.42)	$2.67e^{-7}$ (7.54)	$1.00e^{-9}$ (10.5)
$16*16$	$2.96e^{-3}$ (2.03)	$6.52e^{-6}$ (4.04)	$4.00e^{-9}$ (6.06)	0
$32*32$	$7.35e^{-4}$ (2.01)	$4.05e^{-7}$ (4.01)	0	0
$64*64$	$1.83e^{-4}$ (2.00)	$2.60e^{-8}$ (3.96)		

Table 3 Maximum error at the matching points in Bratu's problem.

[2] Ascher, U. M. and Christiansen, J. and Russell, R. D., "A collocation solver for mixed order systems of boundary value problems", Math. Comp., Vol. 33, pp. 659-679, 1979.

[3] Ascher, U. M. and Spiteri, R. J., "Collocation software for boundary value differential-algebraic equations", SIAM J. Sci. Comput.,Vol. 15,pp. 938-952, 1995.

[4] Bialecki, B., "Convergence analysis of orthogonal spline collocation for elliptic boundary value problems", SIAM J. Numer. Anal., 1997.

[5] Bialecki B. and Cai X. C., "$H^1 - norm$ error bounds for piecewise Hermite bicubic orthogonal spline collocation schemes for elliptic boundary value problems", SIAM J. Numer. Anal., Vol. 31, pp. 1128-1146, 1994.

[6] Christara C. C., "Spline collocation methods, software and architectures for linear elliptic boundary value problems", Ph.D. Thesis, Purdue University, IN, USA, 1988.

[7] de Boor, C. and Swartz, B., "Collocation at Gaussian points", SIAM J. Numer. Anal. , Vol. 10, pp. 582-606, 1973.

[8] Doedel, E. J. and Champneys, A. R. and Fairgrieve, T. F. and Kuznetsov, Y. A. and Sandstede, B. and Wang, X.-J., "AUTO97 : Continuation and bifurcation software for ordinary differential equations", (Available by FTP from ftp.cs.concordia.ca. in directory pub/doedel/auto), Department of Computer Science, Concordia University, Montreal, Canada, 1997.

[9] Doedel, E. J., "Nonlinear Numerics", J. Franklin Inst., Vol. 334B, 5/6,pp. 1049-1073, 1997.

[10] Doedel, E. J., "On the construction of discretizations of elliptic partial differential equations", J. Difference Equations and Applications, Vol. 3, pp. 389-416, 1997.

[11] Fairweather G., "Spline collocation methods for a class of hyperbolic partialintegro-differential equations", SIAM J. Numer. Anal., Vol. 31, pp. 444-460, 1994.

[12] Keller H. B., "Numerical solution of bifurcation and nonlinear eigenvalue problems", Applications of Bifurcation Theory, Rabinowitz, P. H., Academic Press, pp. 359-384, 1977.

[13] Prenter, P. M. and Russell, R. D., "Orthogonal collocation for elliptic partial differential equations", SIAM J. Numer. Anal., Vol. 13, pp. 923-939, 1976.

[14] Russell, R. D. and Christiansen, J., "Adaptive mesh selection strategies for solving boundary value problems", SIAM J. Numer. Anal., Vol. 15, pp. 59-80, 1978.

[15] Sun W., "Cyclic reduction algorithm for solving collocation systems", Intern. J. Computer Math., Vol. 61, pp. 293-305, 1996.

[16] Sun W., "Iterative algorithm for orthogonal spline collocation linear system", SIAM. J. Sci. Comput., Vol. 16, pp. 720-737, 1995.

[17] Wheeler, M. F., "An elliptic collocation-finite element method with interior penalties", SIAM J. Numer. Anal., Vol. 15, No. 1, pp. 152-161, 1978.