Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 224 (2007) 601–615
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Solving regularly and singularly perturbed reaction-diffusion equations in three space dimensions

Peter K. Moore

Department of Mathematics, Southern Methodist University, 208 Clements Hall, 3200 Dyer Street, Dallas, TX 75275, United States

Received 3 May 2006; received in revised form 9 October 2006; accepted 10 October 2006
Available online 8 December 2006

Abstract

In [P.K. Moore, Effects of basis selection and *h*-refinement on error estimator reliability and solution efficiency for higher-order methods in three space dimensions, Int. J. Numer. Anal. Mod. 3 (2006) 21–51] a fixed, high-order *h*-refinement finite element algorithm, *Href*, was introduced for solving reaction-diffusion equations in three space dimensions. In this paper *Href* is coupled with continuation creating an automatic method for solving regularly and singularly perturbed reaction-diffusion equations. The simple quasilinear Newton solver of Moore, (2006) is replaced by the nonlinear solver *NITSOL* [M. Pernice, H.F. Walker, NITSOL: a Newton iterative solver for nonlinear systems, SIAM J. Sci. Comput. 19 (1998) 302–318]. Good initial guesses for the nonlinear solver are obtained using continuation in the small parameter ϵ. Two strategies allow adaptive selection of ϵ. The first depends on the rate of convergence of the nonlinear solver and the second implements backtracking in ϵ. Finally a simple method is used to select the initial ϵ. Several examples illustrate the effectiveness of the algorithm.

MSC: 65N30; 65N50; 92E20

Keywords: Adaptive finite elements; Continuation methods; Perturbation problems; Reaction-diffusion equations

1. Introduction

A significant effort has been devoted to solving regularly and especially singularly perturbed reaction-diffusion equations. Several difficulties arise in solving such problems. Since boundary layers are often present a strategy for enhancing the discretization in these regions is essential. One approach, also used for convection-diffusion problems, employs fitted meshes such as Shishkin- or Bakhvalov-type meshes [22,24,26,37]. A related effort utilizes the high-order of pseudospectral methods coupled with special coordinate transformations to ensure that at least one collocation point lies in the boundary layer [23,39]. In more than one dimension these meshes can be anisotropic [1,21,41]. Generation of such grids often depends on a priori knowledge of the location of the boundary layers and does not lend itself easily to automation. Alternatively adaptivity

E-mail address: pmoore@smu.edu

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2006.10.015

can be used to generate graded meshes [6,8–10,19,20,25]. Adaptive methods have become ubiquitous in the development of automatic codes that require minimal user intervention [3]. These methods depend on accurate error estimation. Adaptive techniques come in three major varieties: grid refinement (*h*-refinement), order variation (*p*-refinement), grid equidistribution (*r*-refinement); and combinations thereof. In one dimension Kopteva et al. [19] have developed a grid equidistribution strategy based on a monitor function [11] coupled with a finite difference discretization. Castaings and Navon [10] compared the *h*-refinement adaptive elliptic solver PLTMG [6] with PLTMG combined with Shishkin meshes and no adaptivity in two dimensions. Melenk and Schwab [25] created a *hp*-adaptive finite element solver for linear singularly perturbed reaction-diffusion equations in two dimensions. Kunert [20] has developed an *h*-adaptive finite element approach that uses anisotropic meshes for linear singularly perturbed reaction diffusion equations in three dimensions.

For nonlinear problems there is the additional difficulty of convergence failure for iterative methods such as Newton's method. Performance may be enhanced through techniques such as backtracking [2,6,9,33]. If these are not sufficient obtaining better initial guesses of the solution can also lead to improved performance. For either automatic codes, where the initial grid is often uniform, or fixed-grid codes using Shishkin- or Bakhvalov-meshes this can be problematic if boundary layers are present. One approach to obtaining good initial guesses is to employ continuation in the small parameter ϵ [8,9]. Continuation methods typically solve the problem for a decreasing sequence of ϵ values using solutions at previous values to generate an initial guess at a subsequent value of ϵ. Continuation can also improve the reliability of error estimates since many estimators have been developed for problems without boundary layers. Continuation should be coupled with adaptivity since for larger values of ϵ coarser grids are likely to be sufficient leading to smaller problems at the beginning. Fixed grid methods coupled with continuation on the other hand will require the solution of large problems throughout.

Herein, I consider a code that combines continuation with the nonlinear solver *NITSOL* [33] and the high-order adaptive finite element method *Href* [32]. To solve the linear systems in *NITSOL*, *GMRES* [35] coupled with *ILUT* preconditioning [34] is employed. This adaptive-continuation code, referred to as *Cont3d*, is used to solve reaction-diffusion equations of the form

$$-\nabla\cdot(D(\mathbf{x},\epsilon)\nabla u) = f(u,\mathbf{x},\epsilon),\ \ \mathbf{x}=(x^1,x^2,x^3)\in\Omega\equiv(X_0^1,X_1^1)\times(X_0^2,X_1^2)\times(X_0^3,X_1^3), \tag{1}$$

$$u(\mathbf{x}) = b(\mathbf{x},\epsilon),\quad \mathbf{x}\in\partial\Omega_d,\quad \frac{\partial u}{\partial v}(\mathbf{x}) = 0,\ \ \mathbf{x}\in\partial\Omega_n, \tag{2}$$

where $\partial\Omega=\partial\Omega_d\cup\partial\Omega_n$ and $\partial\Omega_d$ ($\partial\Omega_n$) is the union of m_d (m_n) faces of Ω. Thus, $m_d+m_n=6$. Throughout I assume that (1) and (2) has an isolated solution of appropriate smoothness and an initial guess can be found sufficiently close to it. The user is required to provide initial (ϵ_I) and final (ϵ_F) values of the parameter ϵ, routines for the functions D, f and b and their partial derivatives, as well as an initial guess of the solution, $u_I(\mathbf{x})$, when $\epsilon=\epsilon_I$. When $b(\mathbf{x},\epsilon_I)\equiv B$ a convenient choice for $u_I(\mathbf{x})$ is $u_I(\mathbf{x})=B$.

Nonlinear reaction-diffusion equations may have multiple solutions [7,12,16,18]. Computing these solutions depends on having multiple good initial guesses. For one class of non-perturbed problems (1) with $D(\mathbf{x},\epsilon)\equiv 1$ and continuity and growth conditions on $f(u,\mathbf{x})$ [12] numerical methods based on the mountain pass lemma [12] or constrained mountain pass lemma [16] have been developed for finding multiple solutions. These methods could serve as preprocessors for *Cont3d* generating appropriate initial guesses $u_I(\mathbf{x})$ and will be investigated in a subsequent effort.

Several additional features have been added to *Cont3d* to improve its robustness. The initial grid is chosen to be uniform. Since the nonlinear solver might not converge for the user's choice of ϵ_I and $u_I(\mathbf{x})$ a simple startup procedure is used that increases ϵ_I until a solution is obtained. An adaptive strategy for selecting successive values of ϵ based on an approach used for bifurcation problems [15] has been implemented. It depends on the convergence behavior of the Newton solver. Finally *NITSOL* has been modified to include backtracking in ϵ as well as the standard backtracking in the Newton direction.

The definition of admissible grids, *h*-refinement rules and discretization are described in Section 2. In Section 3 the error estimation technique is presented along with the *h*-refinement strategy. Section 4 includes descriptions of the critical building blocks of the continuation method. Computational results for five prob-

lems, four singularly perturbed and one regularly perturbed, are given in Section 5 while some conclusions are made in Section 6.

2. Grid definition and discretization

Eqs. (1) and (2) are solved on a sequence of grids G_s, with corresponding values of ϵ, ϵ_s, $s \geqslant 0$ until $\epsilon_s = \epsilon_F$. It may happen that for some step s, $G_s = G_{s-1}$ or $\epsilon_s = \epsilon_{s-1} \equiv \epsilon_F$. Each grid G_s is generated from grid G_{s-1} by recursively trisecting a subset of its elements. Grid G_0 is uniform and obtained by recursively trisecting Ω. Thus, G_s has an octree structure with the root corresponding to Ω. Each vertex in the tree is either a terminal (leaf) vertex or has eight subvertices. A vertex with eight subvertices is referred to as a parent vertex and the eight subvertices are its offspring or children. The leaf vertices of the tree are called elements (unrefined elements in [40]). The elements of G_s are denoted by

$$\Delta_n^s = \left[m_n^{1,s} - \frac{h_n^{1,s}}{2}, m_n^{1,s} + \frac{h_n^{1,s}}{2}\right] \times \left[m_n^{2,s} - \frac{h_n^{2,s}}{2}, m_n^{2,s} + \frac{h_n^{2,s}}{2}\right] \times \left[m_n^{3,s} - \frac{h_n^{3,s}}{2}, m_n^{3,s} + \frac{h_n^{3,s}}{2}\right], \quad n = 1, \ldots, N_{\mathrm{el}}^s, \tag{3}$$

where $(m_n^{1,s}, m_n^{2,s}, m_n^{3,s})$ and $(h_n^{1,s}, h_n^{2,s}, h_n^{3,s})$ are the center and spacing of Δ_n^s, respectively and N_{el}^s is the number of elements. Each element in the grid is made up of element interiors, faces, edges and nodes. These four building blocks are henceforth referred to collectively as *components*. The level l of an element in the grid is the length of the path from the root to the element. A grid is said to be uniform if all its elements are at the same level. A grid is admissible in the sense of Babuška and Rheinboldt [4] if it is defined recursively by the following two rules:

1. G_0 is an admissible grid.
2. If G_s is an admissible grid and $\tilde{\Delta}$ is an element of G_s then the grid obtained from G_s and the eight elements created by trisecting $\tilde{\Delta}$ is admissible.

Such grids contain two general types of components *regular* and *irregular*. A node Γ of G_s is regular if for every element $\tilde{\Delta} \in G_s$ such that $\Gamma \in \tilde{\Delta}$, Γ is a corner node of $\tilde{\Delta}$. An edge Γ of G_s is regular if for every element $\tilde{\Delta} \in G_s$ such that $\Gamma \in \tilde{\Delta}$ its endpoints are corners of $\tilde{\Delta}$. Similarly a face Γ in G_s is regular if for every element $\tilde{\Delta} \in G_s$ such that $\Gamma \in \tilde{\Delta}$, the four corners of Γ are also corners of $\tilde{\Delta}$. All element interiors are regular. Any component that is not regular is irregular. Irregular components appear at the interface of two elements at different levels in the grid. Two elements Δ' and Δ'' are said to be neighbors if their intersection contains an edge of either Δ' or Δ''.

Several additional rules for governing grid refinement and coarsening have been proposed [5,27,40]. The most important rule, the one-irregular rule dictates that on any edge or in the interior of any face there can be no more than one irregular node. Equivalently, two neighboring elements cannot differ by more than one level in the tree. Additional rules and details can be found in [28].

The three-dimensional bases depend on the one-dimensional Lobatto polynomials [38] on the interval $\Lambda = [m - h/2, m + h/2]$ given by

$$\Phi_p(\xi; h; m) = \sqrt{\frac{2p-1}{2}} \int_{-1}^{2(\xi-m)/h} P_{p-1}(s)\,\mathrm{d}s, \quad p \geqslant 2, \tag{4}$$

where $P_{p-1}(s)$ is the Legendre polynomial of degree $p-1$. The Lobatto polynomials along with the linear hat functions

$$\Phi_0(\xi; h; m) = \frac{h/2 - (\xi - m)}{h}, \quad \Phi_1(\xi; h; m) = \frac{h/2 + (\xi - m)}{h}, \tag{5}$$

form a basis for the space of polynomials of degree p on Λ. Henceforth I assume $p > 1$.

The finite element space on Δ_n^s is given by

$$S_{\Delta_n^s}^{p,e,f} \doteq \mathrm{span}\{\Phi_i(x^1; h_n^{1,s}; m_n^{1,s})\Phi_j(x^2; h_n^{2,s}; m_n^{2,s})\Phi_k(x^3; h_n^{3,s}; m_n^{3,s}) | (i,j,k) \in \mathscr{S}^{p,e,f}\}, \tag{6}$$

where the index set $\mathscr{S}^{p,e,f}$ consists of the union of node-based, edge-based, face-based and interior-based index sets, i.e., $\mathscr{S}^{p,e,f} \doteq \mathscr{S}^{N} \cup \mathscr{S}^{E,p} \cup \mathscr{S}^{F,p,f} \cup \mathscr{S}^{I,p,e}$. The component index sets are

$$\mathscr{S}^{N} \doteq \{(i,j,k) | 0 \leqslant i,j,k \leqslant 1\}, \tag{7}$$

$$\mathscr{S}^{E,p} \doteq \{(i,j,k) | 0 \leqslant i,j \leqslant 1, 2 \leqslant k \leqslant p\} \cup \{(i,j,k) | 0 \leqslant i,k \leqslant 1, 2 \leqslant j \leqslant p\} \cup \{(i,j,k) | 0 \leqslant j,k \leqslant 1, 2 \leqslant i \leqslant p\}, \tag{8}$$

$$\mathscr{S}^{F,p,f} \doteq \{(i,j,k) | 0 \leqslant i \leqslant 1, 2 \leqslant j,k \leqslant p, j+k \leqslant f\} \cup \{(i,j,k) | 0 \leqslant j \leqslant 1, 2 \leqslant i,k \leqslant p, i+k \leqslant f\} \cup \{(i,j,k) | 0 \leqslant k \leqslant 1, 2 \leqslant i,j \leqslant p, i+j \leqslant f\}, \tag{9}$$

$$\mathscr{S}^{I,p,e} \doteq \{(i,j,k) | 2 \leqslant i,j,k \leqslant p, i+j+k \leqslant e\}. \tag{10}$$

Note that the space depends on two parameters e and f. If $e = 3p$ and $f = 2p$ the result is the familiar tensor-product space. An investigation of the effectiveness of different choices of e and f was performed in [32]. Based on these results I have selected $p = 4$, $e = 8$ and $f = 7$ for the examples in Section 5.

The finite element space $S^{p,e,f}_{G_s}$ is the space of piecewise continuous polynomials of degree p whose restriction to element Δ^s_n is $S^{p,e,f}_{\Delta^s_n}$. Thus, if $U^s \in S^{p,e,f}_{G_s}$,

$$U^s|_{\Delta^s_n}(\mathbf{x}) \equiv \sum_{(\bar{i},\bar{j},\bar{k}) \in \mathscr{S}^{p,e,f}} U^s_{n;\bar{i},\bar{j},\bar{k}} \Phi_{\bar{i}}(x^1; h^{1,s}_n; m^{1,s}_n) \Phi_{\bar{j}}(x^2; h^{2,s}_n; m^{2,s}_n) \Phi_{\bar{k}}(x^3; h^{3,s}_n; m^{3,s}_n), \tag{11}$$

where the $U^s_{n;\bar{i},\bar{j},\bar{k}}$ are the coordinates associated with element Δ^s_n. Since G_s satisfies the one-irregular rule the support of all basis functions associated with regular components is uniformly bounded [5,40]. The $U^s_{n;\bar{i},\bar{j},\bar{k}}$ associated with irregular components are constrained by continuity across element boundaries. The $U^s_{n;\bar{i},\bar{j},\bar{k}}$ associated with regular grid components constitute the degrees of freedom for the problem. The total number of degrees of freedom on G_s is denoted by N^s_{dof}.

The Galerkin form of (1) in the case of Dirichlet boundary conditions consists of determining $u \in H^1_E(\Omega)$ such that

$$a(u,v) = (f,v), \quad \forall v \in H^1_0(\Omega), \tag{12}$$

where

$$a(u,v) = \int_\Omega D\nabla u \cdot \nabla v \, \mathrm{d}x^1 \, \mathrm{d}x^2 \, \mathrm{d}x^3, \quad (f,v) = \int_\Omega f v \, \mathrm{d}x^1 \, \mathrm{d}x^2 \, \mathrm{d}x^3, \tag{13}$$

and the subscripts E and 0 further restrict functions to satisfy the Dirichlet and homogeneous Dirichlet boundary conditions, respectively. The finite element approximation $U^s \in S^{p,e,f}_{G_s,E}$ is determined as the solution of

$$a(U^s, V^s) = (f, V^s), \ \forall V^s \in S^{p,e,f}_{G_s,0}. \tag{14}$$

Dirichlet boundary conditions are handled using the Lobatto interpolant as presented in [32].

Explicit utilization of (11) in (14) yields the nonlinear system

$$\mathbf{g}(\mathscr{U}^s, \epsilon_s) = 0, \tag{15}$$

where $\mathscr{U}^s$ is the vector of the degrees of freedom (coordinates associated with regular components) of length N^s_{dof} corresponding to the solution U^s on grid G_s. A Newton step for solving (15) consists in computing

$$\mathbf{J}(\mathscr{U}^s, \epsilon_s) \Delta\mathscr{U}^s = -\mathbf{g}(\mathscr{U}^s, \epsilon_s), \tag{16}$$

where $\mathbf{J}$ is the Jacobian matrix and updating $\mathscr{U}^s$ via

$$\mathscr{U}^s \leftarrow \mathscr{U}^s + \Delta\mathscr{U}^s. \tag{17}$$

The solution procedure *Cont3d* (see Fig. 1) is a continuation method consisting of three building blocks: a refinement algorithm *NewGrid* that generates G_s from G_{s-1} using error estimates; a nonlinear routine *Solve*

input: $atol, rtol, \epsilon_I, \epsilon_F, u_I, l_I$
default: l_F, $\Delta\epsilon_{\min}$
Define initial uniform grid G_0 at level l_I and Lobatto interpolant $\bar{U}_0$ of u_I
InitSolve(ϵ_I, $\bar{U}^0$, G_0; ϵ_0, U^0)
$G_{-1} = G_0$
InitSolve($5\epsilon_0/4$, U^0, G_{-1}; ϵ_{-1}, U^{-1})
$l = l_I$, $\Delta l = l_F - l_I, s = 1, E_{-1} = 2$
$\Delta\epsilon_1 = \max((\epsilon_I/\epsilon_F)^{1/\Delta l}, \Delta\epsilon_{\min})$
While ($l \le l_F$) and (($E^{s-2} > 1$) or ($\epsilon_{s-2} > \epsilon_F$))
  ErrEst($G_{s-1}, U^{s-1}, \epsilon_{s-1}; E^{s-1}$)
  $\bar{\epsilon}_s = \max(\epsilon_{s-1}/\Delta\epsilon_s, \epsilon_F)$
  If ($E^{s-1} > 1$) or ($\epsilon_{s-1} > \epsilon_F$)
    If ($E^{s-1} > 1$)
      $l = l + 1$
      Newgrid($G_{s-1}, E^{s-1}; G_s$)
    Else
      $G_s = G_{s-1}$
    End If
    InitGuess($G_{s-2}, \epsilon_{s-2}, U^{s-2}, G_{s-1}, \epsilon_{s-1}, U^{s-1}, G_s, \bar{\epsilon}_s; \bar{U}^s$)
    Solve($\bar{\epsilon}_s, \bar{U}^s, G_s, s, \Delta\epsilon_s; \epsilon_s, U^s, \Delta\epsilon_{s+1}$)
  End If
  $s = s + 1$
End While

Fig. 1. Program *Cont3d*.

(see Fig. 2) for solving (15) on G_s; and an adaptive strategy for selecting ϵ_s. These building blocks are discussed in detail in Sections 3 and 4.

3. Error estimation and adaptivity

The *h*-refinement algorithm *NewGrid* and the *a posteriori* error estimator routine *ErrEst* that drives it were developed for *Href* [32]. In this section they are described briefly for completeness.

To obtain an error estimate consider the augmented finite element solution W^s defined on each element Δ_n^s by

$$W^s(\mathbf{x})|_{\Delta_n^s} = U^s(\mathbf{x})|_{\Delta_n^s} + W_n^{1,s}\psi_{p+1}(x^1; h_n^{1,s}; m_n^{1,s}) + W_n^{2,s}\psi_{p+1}(x^2; h_n^{2,s}; m_n^{2,s}) + W_n^{3,s}\psi_{p+1}(x^3; h_n^{3,s}; m_n^{3,s}), \tag{18}$$

where

$$\psi_{p+1}(\xi; h; m) = \frac{h^{p+1}(p-1)!(p+1)!}{4(2p-1)!}\sqrt{\frac{2}{2p+1}}\Phi_{p+1}(\xi; h; m). \tag{19}$$

The coefficients $W_n^{1,s}$, $W_n^{2,s}$ and $W_n^{3,s}$ are determined by requiring that

$$a(W^s, \widehat{V}_k^s)_{\Delta_n^s} = (f(U^s), \widehat{V}_k^s)_{\Delta_n^s}, \quad k = 1, 2, 3, \tag{20}$$

where $a(\cdot,\cdot)_{\Delta_n^s}$ and $(\cdot,\cdot)_{\Delta_n^s}$ are given by (13) with Ω replaced by Δ_n^s. The function $\widehat{V}_1^s(\mathbf{x})$ is given by

$$\widehat{V}_1^s(\mathbf{x}) = \psi_{p+1}(x^1; h_n^{1,s}; m_n^{1,s})\phi(x^2; h_n^{2,s}; m_n^{2,s})\phi(x^3; h_n^{3,s}; m_n^{3,s}), \tag{21}$$

where

$$\phi(\xi; h; m) = \begin{cases} \frac{\psi_p(\xi;h;m)}{\xi - m}, & p \text{ odd} \\ \frac{\psi_{p+1}(\xi;h;m)}{\xi - m}, & p \text{ even}, \end{cases} \tag{22}$$

```
input: $\bar{\epsilon}_s,\ \bar{U}^s,\ G_s, s, \Delta\epsilon_s$
output: $\epsilon_s,\ U^s,\ \Delta\epsilon_{s+1}$
default: $ftol,\ utol,\ t,\ \eta,\ e_{\max},\ b_{\max}$
$i = 0,\ U = \bar{U}^s,\ \epsilon_s = \bar{\epsilon}_s,\ e = 0$
Repeat
   Compute $J,\ g(\mathcal{U}_i^s, \epsilon_s),\ N_i^s = \|g(\mathcal{U}^s, \epsilon_s)\|$
   If error occurs in assembling $J$ or $g$
      Compute new $\epsilon_s$
   Else
      Solve $J\Delta\mathcal{U}_i^s = -g(\mathcal{U}_i^s, \epsilon_s)$
      Compute $g(\mathcal{U}_i^s + \Delta\mathcal{U}_i^s, \epsilon_s),\ N_{i+1}^s = \|g(\mathcal{U}_i^s + \Delta\mathcal{U}_i^s, \epsilon_s)\|,\ M_{i+1} = \|\Delta\mathcal{U}_i^s\|$
      If $(N_{i+1}^s \le (1 - t(1-\eta))N_i^s)$
         $i = i + 1$
         $\mathcal{U}_i^s = \mathcal{U}_{i-1}^s + \Delta\mathcal{U}_{i-1}^s$
      Else
         $b = 0$
         While $(b < b_{\max})$ and $(N_{i+1}^s > (1 - t(1-\eta))N_i^s)$
            $b = b + 1$
            Choose $\theta$
            $\Delta\mathcal{U}_i^s = \theta\Delta\mathcal{U}_i^s,\ \eta = 1 - \theta(1-\eta)$
            $N_{i+1}^s = \|g(\mathcal{U}_i^s + \Delta\mathcal{U}_i^s, \epsilon_s)\|$
         End While
         If $(N_{i+1}^s > (1 - t(1-\eta))N_i^s)$
            $e = e + 1$
            Choose new $\epsilon_s$, Reinitialize $\mathcal{U}_0^s$
         End If
      End If
   End If
Until $((N_i^s < ftol)$ and $(M_i < utol))$ or $(e > e_{\max})$
Estimate converge rate $r$ from $N_j^s,\ j = 1, \ldots, i$
Compute $\Delta\epsilon_{s+1}$
```

Fig. 2. Subroutine *Solve*.

with comparable definitions for $\widehat{V}_2^s$ and $\widehat{V}_3^s$. On elements with no irregular nodes the error estimate has the form

$$|E^s|^2_{1,\Delta_n^s} \equiv |W^s - U^s|^2_{1,\Delta_n^s} = \left(\frac{(p-1)!}{(2p-1)!}\right)^2 \frac{h_n^{1,s}h_n^{2,s}h_n^{3,s}}{4(2p+1)}\left(((h_n^{1,s})^p W_n^{1,s})^2 + ((h_n^{2,s})^p W_n^{2,s})^2 + ((h_n^{3,s})^p W_n^{3,s})^2\right) + h.o.t. \tag{23}$$

For elements with irregular nodes the formulas are more involved and depend on the location and number of these nodes [32]. Results on the efficacy of this estimator can be found in [31,32].

The *h*-refinement strategy *NewGrid* is based on an earlier one-dimensional approach [14]. Error estimates are measured in the root-mean-square-H^1 norm [14]

$$|E^s|_{\Delta_n^s,\mathrm{rms}} = \frac{|E^s|_{1,\Delta_n^s}}{\mathrm{atol} + \mathrm{rtol}|U|_{1,\Delta_n^s}}, \quad |E^s|^2_{G_s,\mathrm{rms}} = \sum_{n=1}^{N_{\mathrm{el}}^s} |E^s|^2_{\Delta_n^s,\mathrm{rms}}, \tag{24}$$

where atol and rtol are the absolute and relative error tolerances, respectively. If on G_s, $|E|_{G_s,\mathrm{rms}} > 1$ elements Δ_n^s with $|E^s|_{\Delta_n^s,\mathrm{rms}} > \frac{\mathrm{rf}}{\sqrt{N_{\mathrm{el}}^s}}$ are refined as described in Section 2 where rf is a refinement safety factor. Eight sibling elements are coarsened into one element if the error indicators, $|E^s|_{\Delta_n^s,\mathrm{rms}}$, on all eight are smaller than $\frac{\mathrm{cf}}{\max(1,2^{p-3})\sqrt{N_{\mathrm{el}}^s}}$ [14] where cf is the coarsening safety factor.

4. Solution procedures

In this section a description of the main features of *Cont3d*, as shown in Fig. 1, is provided. Given the user-supplied initial guesses ϵ_I and $u_I(\mathbf{x})$, the program begins by generating a default uniform initial grid G_0 and a Lobatto interpolant [29,32] $\overline{U}^0(\mathbf{x})$ of $u_I(\mathbf{x})$. Then a solution U^0 and corresponding $\epsilon_0 \geqslant \epsilon_I$ are found using *InitSolve*. A second solution on the same grid with a slightly larger ϵ value is then computed. For each step $s \geqslant 1$, a sequence of routines is called until the tolerance is satisfied with $\epsilon = \epsilon_F$. First a guess, $\bar{\epsilon}_s$, of the next value of ϵ for which a solution of (15) can be obtained, is computed (see below). Next the error in U^{s-1} is computed by *ErrEst* as described in Section 3. If E^{s-1} is too large a new grid G_s is generated by *NewGrid* (see Section 3), otherwise the current grid is reused. In either case *InitGuess* computes an initial guess, $\overline{U}^s$, of the solution by extrapolation from the two previous solutions and grids. Finally the Newton-based routine *Solve* solves (15) and determines the size of the next decrease in ϵ, $\Delta\epsilon_{s+1}$.

Solve is adapted from the nonlinear solver *NITSOL* [13,33]. The heart of *Solve*, as shown in Fig. 2, consists in solving (15) using Newton's method with backtracking, both the traditional backtracking in the Newton direction [2,33] and an additional backtracking in ϵ. The solution of (16) is computed by *GMRES* [35] with the *ILUT* preconditioner of Saad [34]. If the reduction in $\|g\|$ is sufficient (see [13,33] for details) the Newton step is accepted. If not, backtracking in the Newton direction $\Delta\mathscr{U}$ is attempted (the value of this backtracking is demonstrated in Example 5.5) [13,33]. If Newton backtracking fails the algorithm switches to backtracking in ϵ. A new value of ϵ is generated by taking the reciprocal of the average of the reciprocals of the current ϵ and the last successful value of ϵ, and the Newton iteration is restarted. The same approach is also used if on any Newton step an error occurs in assembling either J or g. Error detection in the assembly routines for computing D, f and b and their partial derivatives (for J) must be supplied by the user (see Example 5.5).

Once convergence is attained the appropriate reduction in ϵ for the next step, $\Delta\epsilon_{s+1}$, is estimated. When step s is finished the next guess at the appropriate value of ϵ is given by

$$\bar{\epsilon}_{s+1} = \frac{\epsilon_s}{\Delta\epsilon_{s+1}}, \tag{25}$$

where $\Delta\epsilon_{s+1} > 1$. Two different strategies are employed depending on whether or not backtracking in ϵ occurs. If no ϵ-backtracking is needed $\Delta\epsilon_{s+1} = \rho\Delta\epsilon_s$. The algorithm for automatic selecting ρ is based on the hypothesis [15] that

$$N_1^s = \alpha\Delta\epsilon, \tag{26}$$

where $N_1^s = \|g(\overline{\mathscr{U}}^s, \epsilon_s)\|$ and that there exist C and r such that

$$\lim_{j\to\infty} \frac{N_{j+1}^s}{(N_j^s)^{1+r}} = C. \tag{27}$$

Then following [15] if the number of successful Newton iterations to achieve convergence is J, ρ is chosen so that $\rho < \rho_{\max}$ and

$$\rho = \begin{cases} \left(\frac{\text{ftol}}{N_{J-1}^s}\right)^{1/(1+r)^{J-2}}, & J > J_{\max} \\ \left(\frac{\text{ftol}}{N_J^s}\right)^{1/(1+r)^{J-1}}, & \text{otherwise,} \end{cases} \tag{28}$$

where ftol is the tolerance control for N_j^s (see Fig. 2). If $J > 2$ the rate r is estimated by computing

$$r = \frac{1}{J-2}\sum_{i=1}^{J-2} \frac{\ln\left(\frac{N_{i+2}^s}{N_{i+1}^s}\right)}{\ln\left(\frac{N_{i+1}^s}{N_i^s}\right)}, \tag{29}$$

otherwise $r = 1$. If backtracking in ϵ occurs the previous history of ϵ reductions is discarded. In this situation $\Delta\epsilon_{s+1} = 0.9\rho\epsilon_s/\epsilon_{s-1}$ where 0.9 represents a safety factor and ρ is given by (28). In either case I also require that $\Delta\epsilon_{s+1} \geqslant \Delta\epsilon_{\min}$.

The startup routine *InitSolve* adaptively selects an initial ϵ, ϵ_0, on a uniform grid G_0 if the user-prescribed value ϵ_I is too small. *InitSolve* uses Newton's method but with no backtracking in the Newton direction. Instead, if Newton's method is not converging ϵ is increased by the ratio of the norms of g for two successive iterates for which the Newton convergence criteria is not satisfied. This heuristic works well in the one example of Section 5 where it is needed. *InitSolve* is invoked a second time on G_0 with $\epsilon = 5\epsilon_0/4$. This generates a second solution for the predictor computed by *InitGuess*. Since computer-resource use is dominated by the solution on the finest grid the cost associated with this second solution on G_0 is negligible (cf. Example 5.1).

The routine *InitGuess* computes an initial guess $\overline{U}^s$ for *Solve* in two steps. First U^{s-2} and U^{s-1} are interpolated to G_s using the Lobatto interpolation strategy described in [32]. Then linear extrapolation in ϵ is used to generate $\overline{U}^s$ from the interpolants. This amounts to an explicit Euler predictor.

The closest codes to my approach are a pair of one-dimensional continuation methods [8,9] due to Cash et al. The first is based on the collocation strategy *COLMOD* and the second, *ACDC* uses deferred correction coupled with Lobatto Runge–Kutta formulas. I will refer to these codes collectively as *Cont1d*. These codes include backtracking in both ϵ and the Newton direction comparable to the approach taken here.

In addition to differences due to dimensionality (direct versus iterative methods for the linear systems, for example), discretization and error estimation, these codes solve (1) and (2) on the same grid twice for two values of ϵ whereas *Cont3d* is not required to do so. As a result they use the second solution as the initial guess on the same grid with a new ϵ while I use an explicit Euler predictor. The adaptive algorithm for selecting the next guess at ϵ in *Cont1d* takes advantage of the solutions on two grids with the same ϵ adjusted (in the nonlinear case) by a simple convergence factor due to Seydel [36]. The approach in *Cont3d* is based solely on the convergence of Newton's method [15]. In *Cont1d* the user is required to specify the initial guess ϵ_I and no adaptive procedure is invoked if a solution cannot be found. My approach, although it is simple and could easily be done "by hand", proves effective for one problem.

5. Computational results

All calculations are performed in double precision either on a HP Xeon workstation with 8GB RAM (Examples 5.1–5.3) or on a SGI Altix with 64GB RAM (Examples 5.4 and 5.5). For many of the modules various parameters must be specified. Associated with *ILUT* are two parameters, the drop tolerance *itol*, and the maximum row fill-in *ifil*. I use $itol = 10^{-6}$ and $ifil = 800$ for Examples 5.1–5.3 and $ifil = 2000$ for the final two examples. The tolerance for *GMRES* is chosen to be 10^{-12}. The maximum size of the Krylov subspace is set at 10 (see [32] for a discussion of this choice). The h-adaptivity safety factors rf and cf (cf. Section 3) have values 0.8 and 0.1, respectively. I coded my own version of *NITSOL*. Its parameters (cf. Fig. 2) are $\text{ftol} = 10^{-4}$, $utol = 10^{-4}$, $t = 0.1$, $\eta = 0.5$ and $b_{\max} = 5$. The other parameter values (cf. Section 4) are $e_{\max} = 7$, $J_{\max} = 5$, $\rho_{\max} = 2$ and $\Delta\epsilon_{\min} = 1.1$. The initial grid is uniform $4 \times 4 \times 4$ with $l_I = 2$. For comparison purposes a non-continuation code is also employed, referred to as *Bench*. It consists of the adaptive refinement code *Href* with the simple nonlinear solver replaced by *NITSOL*. The codes are written in a combination of Fortran 77 and Fortran 90. For the nonlinear equations no analytic or benchmark solutions are available for comparison. However, both numerical and asymptotic solutions for the same equations in lower dimensions lend credence to the accuracy of the solutions obtained by the two codes.

Example 5.1. Consider the linear equation [25]

$$-\epsilon\Delta u = 1 - u, \ \mathbf{x} \in \Omega \equiv (0,1)^3, \tag{30}$$

$$u(\mathbf{x}) = 0, \ \mathbf{x} \in \partial\Omega_d, \ \frac{\partial u}{\partial \nu}(\mathbf{x}) = 0, \ \mathbf{x} \in \partial\Omega_n, \tag{31}$$

where $\partial\Omega_d = \{(x^1,x^2,x^3) \in \partial\Omega | x^3 z = 0\}$. The exact solution of (30) and (31) is

$$u(\mathbf{x}) = 1 - \frac{\cosh((1-x^3)/\epsilon)}{\cosh(1/\epsilon)}. \tag{32}$$

Problem (30) and (31) is solved with $\epsilon_I = 0.01$, $\epsilon_F = 0.0001$, atol $= 0.01$, rtol $= 0$ and $u_I(\mathbf{x}) = 0$. *Bench* is able to solve the problem with $\epsilon = \epsilon_F$. On the finest grid it requires $N^4_{\text{dof}} = 438,465$. *Bench* needs 4 levels of refinement and 1.2*e*4 s of CPU time.

The continuation history of *Cont3d* is shown in Table 1. It requires 5 steps and 4 levels of refinement but only 1.0*e*4 s of CPU time. On the finest grid $N^5_{\text{dof}} = 427,881$. As expected since (30) is linear only two Newton iterations per step are required. The number of *GMRES* iterations is almost fixed reflecting the flexibility of the preconditioner.

For each step of continuation s, the error in the H^1-seminorm $|e^s|_1$ where $e^s = u^s - U^s$, effectivity index $\theta^s = \frac{|E^s|_1}{|e^s|_1}$ and percentage of time used are displayed in Table 2. The results indicate that the error estimator is accurate throughout the computation. Thus, the error on the final grid satisfies the tolerance. Only 1% of the time is consumed on each of the first three steps while the final step uses three quarters of the total time. This validates the claim in Section 4 that the cost of the extra work in solving twice on the initial grid is insignificant. It also demonstrates that continuation with a fixed grid would be computationally inefficient.

Example 5.2. Consider [37]

$$-\epsilon\Delta u = -(1+u)(1+(1+u)^2), \quad \mathbf{x} \in \Omega \equiv (0,1)^3, \tag{33}$$

$$u(\mathbf{x}) = 0, \quad \mathbf{x} \in \partial\Omega. \tag{34}$$

Eqs. (33) and (34) are solved with $\epsilon_I = 0.1$, $\epsilon_F = 0.002$, atol $= 0.01$, rtol $= 0$ and $u_I(\mathbf{x}) = 1$. *Bench* solves the problem with $\epsilon = \epsilon_F$. On the finest grid after 4 levels of refinement $N^4_{\text{dof}} = 599,485$. The total CPU time needed is 2.6*e*4 s.

From the continuation history displayed in Table 3 it is seen that *Cont3d* requires 4 steps and 3 refinement levels. In the first step, with $\epsilon_1 = 0.0521$, the uniform initial grid is still adequate. The finest grid is the same as for the benchmark code. However, the total CPU time is 2.9*e*4 s. This slight increase is due to the additional overhead associated with the continuation method. No backtracking is necessary. For this problem the Euler predictor does not provide any savings since if it is turned off the total CPU time is 2.8*e*4 s. This is not surprising given that the benchmark code has no predictor and runs faster on this problem.

Table 1
The refinement level l, value of ϵ_s, number of degrees of freedom, N^s_{dof}, number of Newton iterations, N^s_{NEWT} and number of iterations of *GMRES*, N^s_{GMRES}, at each step s of *Cont3d* in solving Example 5.1

s	l	ϵ_s	N^s_{dof}	N^s_{NEWT}	N^s_{GMRES}
0	2	0.0100	3585	2	4
1	3	0.00464	8877	2	5
2	3	0.00108	8877	2	5
3	4	0.000125	29,313	2	5
4	5	0.000100	109,605	2	6
5	6	0.000100	427,881	2	6

Table 2
The error in the H^1-seminorm $|e^s|_1$, effectivity index θ^s, and percentage of the total time spent, % time, at each step s of *Cont3d* in solving Example 5.1

s	$\lvert e^s\rvert_1$	θ^s	% Time
0	1.181×10^{-2}	1.052	1
1	5.021×10^{-3}	1.034	1
2	7.667×10^{-2}	1.119	1
3	3.538×10^{-1}	1.250	4
4	7.686×10^{-2}	1.081	17
5	7.891×10^{-3}	1.036	76

Table 3
The refinement level l, value of ϵ_s, number of degrees of freedom, N^s_{dof}, number of Newton iterations, N^s_{NEWT} and number of iterations of *GMRES*, N^s_{GMRES}, at each step s of *Cont3d* in solving Example 5.2

s	l	ϵ_s	N^s_{dof}	N^s_{NEWT}	N^s_{GMRES}
0	2	0.100	3585	5	11
1	2	0.0521	3585	4	9
2	3	0.0136	25,505	4	15
3	4	0.00232	122,941	5	22
4	5	0.00200	599,485	3	16

Example 5.3. Consider

$$-\epsilon\Delta u = u^2, \ \mathbf{x} \in \Omega \equiv (0,1)^3, \tag{35}$$

$$u(\mathbf{x}) = 1, \ \mathbf{x} \in \partial\Omega. \tag{36}$$

Eqs. (35) and (36) are solved with $\epsilon_{\text{I}} = 0.01$, $\epsilon_{\text{F}} = 0.001$, atol $= 0.01$, rtol $= 0$ and $u_{\text{I}}(\mathbf{x}) = 1$. *Bench* solves (35) and (36) with $\epsilon = \epsilon_{\text{F}}$. On the final level $N^4_{\text{dof}} = 572,661$ and it uses 2.4*e*4 s of CPU time.

Table 4 shows the continuation history in solving (35) and (36) with *Cont3d*. It converges after 4 steps and 3 levels of refinement. Again the finest grid is the same as for the benchmark code. *Cont3d* uses 3.2*e*4 s of CPU time. If the predictor in *Cont3d* is turned off the total CPU time increases to 4.1*e*4. This is primarily due to two additional Newton iterations needed on the final step on the finest grid although the final number of unknowns $N^4_{\text{dof}} = 570,813$ is slightly smaller.

In these three examples *Bench* is able to solve the problems with $\epsilon = \epsilon_{\text{F}}$ directly. In all cases the backtracking capability of *NITSOL* is not needed. This demonstrates how well adaptivity alone works in solving singularly perturbed reaction-diffusion equations for moderately-sized values of ϵ, even when the initial guesses are crude. Since Jacobian assembly and preconditioner factorization are the most expensive steps [30] the difference in CPU time between *Bench* and *Cont3d* on the first example is due to the larger number of unknowns needed by *Bench* on the final grid. In the other two examples the overhead costs of *Cont3d* make it less efficient than *Bench*. The usefulness of the predictor is not clear-cut for these examples. The convergence to ϵ_{F} is relatively rapid since $N^s_{NEWT} < J_{\max}$ on almost all steps. The number of *GMRES* iterations is also modest in all cases.

In the next problem ϵ_{I} is chosen so that *Bench* no longer works. This allows a simple demonstration of the initialization strategy of Section 4. The final problem is significantly more difficult than the first four. Continuation is essential since *Bench* is no longer capable of obtaining a solution.

Example 5.4. Consider the generalization to three dimensions of the singularly-perturbed equation from [7]:

$$-\epsilon\Delta u = 2(1 - ((x^1)^2 + (x^2)^2 + (x^3)^2))u + u^2 - 1, \quad \mathbf{x} \in \Omega \equiv (-1,1)^3, \tag{37}$$

$$u(\mathbf{x}) = 0, \ \mathbf{x} \in \partial\Omega. \tag{38}$$

Eq. (37) is solved with $\epsilon_{\text{I}} = 0.1$, $\epsilon_{\text{F}} = 0.01$, atol $= 0.01$, rtol $= 0$ and $u_{\text{I}}(\mathbf{x}) = 0$. *Bench* is unable to solve the problem on the initial grid, the Newton solver fails to converge.

Table 4
The refinement level l, value of ϵ_s, number of degrees of freedom, N^s_{dof}, number of Newton iterations, N^s_{NEWT} and number of iterations of *GMRES*, N^s_{GMRES}, at each step s of *Cont3d* in solving Example 5.3

s	l	ϵ_s	N^s_{dof}	N^s_{NEWT}	N^s_{GMRES}
0	2	0.0100	3585	6	14
1	3	0.00681	25,505	3	10
2	4	0.00281	122,941	4	18
3	4	0.00100	122,941	4	20
4	5	0.00100	572,661	3	16

The Newton solver also fails to converge in *Cont3d* for $\epsilon_I = 0.1$. It then automatically increases ϵ to 0.5094 for which a solution is obtained. As seen in the continuation history displayed in Table 5, six steps and four levels of refinement are required to obtain a solution. On the finest grid $N^7_{\text{dof}} = 1,790,207$. The code reaches $\epsilon = 0.01$ on the fourth step. A total of 5.6*e*5 s of CPU time is needed. If the predictor is turned off the code requires an additional Newton iteration on each step s, $1 \leqslant s \leqslant 3$. This leads to a slightly larger CPU time of 5.8*e*5 s though the value N^s_{dof} is the same on each step.

Isosurfaces $U^6 = -0.1, -0.5, -0.8, -1.5$ and -2.1, along with a contour plot on the surface $z = 0$ of the solution obtained by *Cont3d* are displayed in Fig. 3. The presence of boundary layers at each face can be clearly seen. In one dimension the authors demonstrate the existence of multiple solutions [7]. In three dimensions the solution shown in Fig. 3 (corresponding to the first of their solutions) is the only one I was able to obtain despite trying a variety of initial guesses $u_I(\mathbf{x})$. As noted earlier a more robust approach would be to use mountain-pass based strategies [12,16] to generate initial guesses.

Example 5.5. Consider the chemical reaction problem [17]

$$\Delta u = -\lambda^2(1 + \beta - u)\exp\left(\frac{u-1}{\epsilon u}\right), \quad \Omega \equiv (0,1)^3, \tag{39}$$

$$u(\mathbf{x}) = 1, \ \mathbf{x} \in \partial\Omega_d, \quad \frac{\partial u}{\partial v}(\mathbf{x}) = 0, \ \mathbf{x} \in \partial\Omega_n, \tag{40}$$

Table 5
The refinement level l, value of ϵ_s, number of degrees of freedom, N^s_{dof}, number of Newton iterations, N^s_{NEWT}, number of iterations of *GMRES*, N^s_{GMRES} and number of ϵ-backtracking steps, $N^s_{\epsilon-\text{backtrack}}$, at each step s of *Cont3d* in solving Example 5.4

s	l	ϵ_s	N^s_{dof}	N^s_{NEWT}	N^s_{GMRES}	$N^s_{\epsilon-\text{backtrack}}$
0	2	0.509	3585	6(5)	14(11)	1
1	2	0.265	3585	4	9	0
2	2	0.104	3585	5	11	0
3	3	0.0205	13,981	4	9	0
4	4	0.0100	106,513	4	10	0
5	5	0.0100	477,589	3	10	0
6	6	0.0100	1,790,207	3	13	0

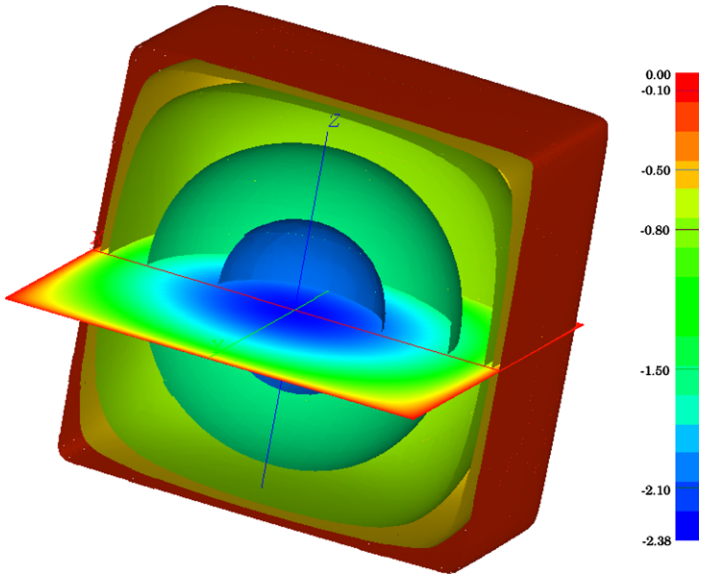

Fig. 3. Isosurfaces $U^6 = -0.10, -0.50, -0.80, -1.50$ and -2.10 and a contour plot on the surface $z = 0$ of the solution of Example 5.4.

where $\partial\Omega_d = \{(x^1,x^2,x^3) \in \partial\Omega | x^1 = 0 \text{ or } x^2 = 0 \text{ or } x^3 = 0\}$. Eqs. (39) and (40) are solved with $\beta = \lambda = 1$, $\epsilon_I = 0.2$, $\epsilon_F = 0.05$, atol $= 0.05$, rtol $= 0$ and $u_I(\mathbf{x}) = 1$. An error is flagged in the routines for computing f and $\frac{df}{du}$ if $U^s < 0$ at any Gauss–Legendre integration point. Without this the code fails due to overflow. *Bench* is unable to solve the problem on the initial grid since the Newton solver fails to converge.

The continuation history in solving (39) and (40) with *Cont3d* is shown in Table 6. On the first five steps the problem is solved on the initial grid. Newton backtracking is used on the first Newton iteration of the first step allowing the algorithm to continue without increasing ϵ. At the third step $\epsilon_3 = 0.0636$ but this leads to a solution that is no longer positive after one Newton iteration. A new value of $\epsilon_3 = 0.0879$ again results in $U^s < 0$ at an integration point after one Newton iteration leading to a second backtracking step in ϵ. With ϵ_3 now 0.0917 Newton's method converges in 5 iterations. A similar situation happens at steps five and seven but the algorithm backtracks in ϵ only once. The total time taken is 9.8e5 s with $N^{10}_{\text{dof}} = 1,544,267$ unknowns on the final grid. Without the predictor the code requires 13 steps and 6 levels of refinement. It uses 26 more Newton iterations, 12 more Newton-backtracks and one more ϵ-backtrack than *Cont3d*. On the final grid $N^{13}_{\text{dof}} = 2,565,823$. The estimate of the error on the final grid is about half of *Cont3d*. The total CPU time taken is 2.2e6 s.

To study the usefulness of Newton backtracking I also ran *Cont3d* with this backtracking turned off allowing only ϵ-backtracking. In this case 11 steps and 4 refinement levels (from the uniform initial grid) are required. Backtracking in ϵ occurs on six of the steps for a total of 10 as opposed to 4 for *Cont3d*. The code without Newton backtracking uses 10 more Newton iterations and has $N^{11}_{\text{dof}} = 994,578$ on the final grid. The total CPU time of 7.1e5 s is less than that used by *Cont3d* while the estimated error is twice that of *Cont3d*.

A comparison of the performances of *Cont3d* and the two variants on this example is more difficult than in the earlier examples since their continuation histories are so different. As noted in [32] fixed high-order *h*-refinement codes tend to over-refine on the final grid yielding an error estimate that is significantly smaller than the tolerance. This behavior magnifies differences in solution strategies leading to large variations in N^s_{dof} and corresponding error estimates on final grids. As a result CPU time alone is misleading as an indicator of effectiveness. If the number of Newton steps, and Newton- and ϵ-backtracks are taken into account then *Cont3d* appears to be the most robust.

The solution to (39) and (40) is displayed in Fig. 4 as a family of isosurfaces corresponding to $U^{10} = 1.10$, 1.50, 1.80, 1.95 and 2.00. All five surfaces are part of the boundary layer that occurs at the faces $x^1 = x^2 = x^3 = 1$.

6. Conclusions

Herein I have presented a high-order *h*-refinement algorithm coupled with continuation for solving regularly and singularly perturbed reaction diffusion equations in three space dimensions. The continuation pro-

Table 6
The refinement level l, value of ϵ_s, number of degrees of freedom, N^s_{dof}, number of Newton iterations, N^s_{NEWT}, number of iterations of GMRES, N^s_{GMRES}, number of ϵ-backtracking steps, $N^s_{\epsilon-\text{backtrack}}$ and number of Newton-backtracking steps, $N^s_{\text{NEWT-backtrack}}$, at each step s of *Cont3d* in solving Example 5.5

s	l	ϵ_s	N^s_{dof}	N^s_{NEWT}	N^s_{GMRES}	$N^s_{\epsilon-\text{backtrack}}$	$N^s_{\text{NEWT-backtrack}}$
0	2	0.2	3585	4	10	0	0
1	2	0.159	3585	6	15	0	1
2	2	0.142	3585	4	9	0	0
3	2	0.109	3585	7(5)	19(12)	2	0
4	2	0.0917	3585	5	13	0	0
5	2	0.0764	3585	8(7)	24(21)	1	0
6	3	0.0694	16,246	5	14	0	0
7	4	0.0581	60,598	10(9)	45(37)	1	0
8	5	0.0529	279,911	8	28	0	0
9	6	0.0500	630,210	5	18	0	0
10	7	0.0500	1,544,267	3	11	0	0

The numbers in parentheses are the numbers of iterations at the final ϵ for that step.

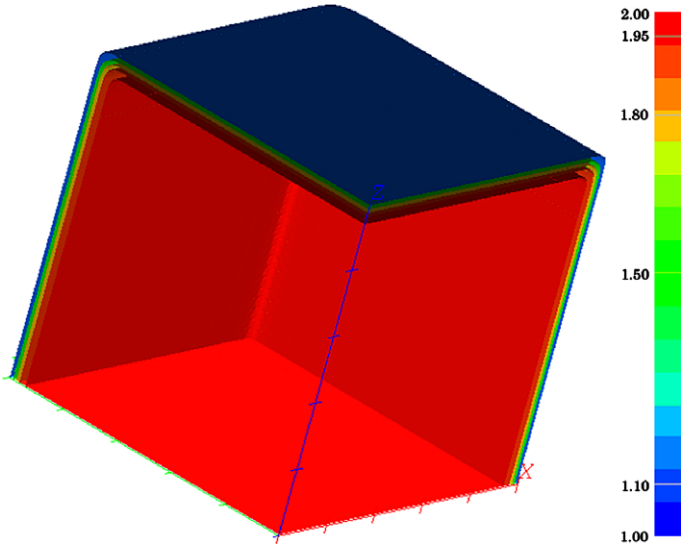

Fig. 4. Isosurfaces $U^{10} = 1.10$, 1.50, 1.80, 1.95 and 2.00 of the solution of Example 5.5.

cedure allows for backtracking, both in the Newton direction (first choice) and in the small parameter ϵ. Guesses of the solution from step to step are computed using an Euler predictor. As a comparison two other algorithms are used. The first is the *h*-refinement procedure without continuation and the second involves using the current solution interpolated to the next grid as a predictor.

The algorithms are used to solve five equations. The first equation is linear while the remaining four are nonlinear. For the first three equations both methods are able to obtain solutions with the non-continuation algorithm delivering slightly better performance as measured by total CPU time. Thus, adaptivity alone is sufficient for some problems. For the final two examples only the continuation code *Cont3d* is successful. This is especially true of the last example where both types of backtracking are invoked. On the last three examples the Euler predictor enhances the performance.

Convergence of the error estimation strategy [31,32] has not been proved for regularly and singularly perturbed problems but coupled with continuation it appears to work well as demonstrated in the first example.

Boundary layers that require refinement, especially on box-shaped domains, are often aligned with the domain boundaries. Further improvements in effectiveness could be achieved by using directional (anisotropic) refinement [20]. The continuation algorithm described in this paper would likely extend to adaptive methods on more general domains but the error estimation and *h*-adaptive strategies might need significant revision.

For problems with multiple solutions a promising approach is to use algorithms based on the mountain pass lemma [12,16] to obtain multiple solutions for $\epsilon = \epsilon_I$. Each of these solutions would then serve as an initial guess U^0 for *Cont3d*. A key step in these methods is the solution of a linear elliptic equation for the steepest descent direction. A variation of *Cont3d* could be used solve this equation improving accuracy and robustness.

Acknowledgment

This research was partially supported by NSF Grant #DMS-0203154.

References

[1] T. Apel, G. Lube, Anisotropic mesh refinement for a singularly perturbed reaction diffusion model problem, Appl. Numer. Math. 26 (1998) 415–433.
[2] U.M. Ascher, R.M.M. Mattheij, R.D. Russell, Numerical Solution of Boundary Value Problems for Ordinary Differential Equations, SIAM, Philadelphia, 1995.

[3] I. Babuška, J.E. Flaherty, W.D. Henshaw, J.E. Hopcroft, J.E. Oliger, T. Tezduyar (Eds.), Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations, Springer-Verlag, New York, 1995.
[4] I. Babuška, W.C. Rheinboldt, Error estimates for adaptive finite element computations, SIAM J. Numer. Anal. 15 (1978) 736–754.
[5] R.E. Bank, The efficient implementation of local mesh refinement algorithms, in: I. Babuska, J. Chandra, J.E. Flaherty (Eds.), Adaptive Computational Methods for Partial Differential Equations, SIAM, Philadelphia, 1983, pp. 74–81.
[6] R.E. Bank, PLTMG: A Software Package for Solving Elliptic Partial Differential Equations Users Guide 8.0, SIAM, Philadelphia, 1998.
[7] C.M. Bender, S.A. Orszag, Advanced Mathematical Methods for Scientists and Engineers, McGraw-Hill Book Co., New York, 1978.
[8] J.R. Cash, G. Moore, R.W. Wright, An automatic continuation strategy for the solution of singularly perturbed linear two point boundary value problems, J. Comp. Phys. 122 (1995) 266–279.
[9] J.R. Cash, G. Moore, R.W. Wright, An automatic continuation strategy for the solution of singularly perturbed nonlinear boundary value problems, ACM Trans. Math. Soft. 27 (2001) 245–266.
[10] W. Castaings, I.M. Navon, Mesh refinement strategies for solving singularly perturbed reaction-diffusion problems, Comp. Math. Appl. 41 (2001) 157–176.
[11] W.M. Cao, W.Z. Huang, R.D. Russell, A study of monitor functions for two-dimensional adaptive mesh generation, SIAM J. Sci. Comput. 20 (1999) 1978–1994.
[12] Y.S. Choi, P.J. McKenna, A mountain pass method for the numerical solution of semilinear elliptic problems, Nonlinear Anal. 20 (1993) 417–437.
[13] S.C. Eisenstat, H.F. Walker, Globally convergent inexact Newton methods, SIAM J. Optim. 4 (1994) 393–422.
[14] J.E. Flaherty, P.K. Moore, Integrated space-time adaptive *hp*-refinement methods for parabolic systems, Appl. Numer. Math. 16 (1995) 317–341.
[15] W.J.F. Govaerts, Numerical Methods for Bifurcations of Dynamical Equilibria, SIAM, Philadelphia, 2000.
[16] J. Horák, Constrained mountain pass algorithm for the numerical solution of semilinear elliptic problems, Numer. Math. 98 (2004) 251–276.
[17] A.K. Kapila, B.J. Matkowsky, Reactive-diffuse systems with Arhenius kinetics: multiple solutions, ignition and extinction, SIAM J. Appl. Math. 36 (1979) 373–389.
[18] N. Kopteva, M. Stynes, Numerical analysis of a singularly perturbed nonlinear reaction-diffusion problem with multiple solutions, Appl. Numer. Math. 51 (2004) 273–288.
[19] N. Kopteva, N. Madden, M. Stynes, Grid equidistribution for reaction-diffusion problems in one dimension, Numer. Alg. 40 (2005) 305–322.
[20] G. Kunert, A posteriori H^1 error estimation for a singularly perturbed reaction diffusion problem on anisotropic meshes, IMA J. Numer. Anal. 25 (2005) 408–428.
[21] J. Li, I.M. Navon, Uniformly convergent finite element methods for singularly perturbed elliptic boundary value problems I: reaction-diffusion type, Comput. Math. Appl. 35 (1998) 57–70.
[22] T. Linss, H.G. Roos, R. Vulanovic, Uniform pointwise convergence on Shishkin-type meshes for quasi-linear convection-diffusion problems, SIAM J. Numer. Anal. 38 (2000) 897–912.
[23] W. Liu, T. Tang, Error analysis for a Galerkin-spectral method with coordinate transformation for solving singularly perturbed problems, Appl. Numer. Anal. 38 (2001) 315–345.
[24] N. Madden, M. Stynes, A uniformly convergent numerical method for a coupled system of two singularly perturbed linear reaction-diffusion problems, IMA J. Numer. Anal. 23 (2003) 627–644.
[25] J.M. Melenk, C. Schwab, *HP* FEM for reaction-diffusion equations I: robust exponential convergence, SIAM J. Numer. Anal. 35 (1998) 1520–1557.
[26] J.J.H. Miller, E. O'Riordan, G.I. Shishkin, Fitted Numerical Methods for Singular Perturbation Problems, World Scientific, Singapore, 1996.
[27] P.K. Moore, Finite difference methods and spatial a posteriori error estimates for solving parabolic equations in three space dimensions on grids with irregular nodes, SIAM J. Numer. Anal. 36 (1999) 1044–1064.
[28] P.K. Moore, An adaptive finite element method for parabolic differential systems: some algorithmic considerations in solving in three space dimensions, SIAM J. Sci. Comput. 21 (2000) 1567–1586.
[29] P.K. Moore, Applications of Lobatto polynomials to an adaptive finite element method: a posteriori error estimates for hp-adaptivity and grid-to-grid interpolation, Numer. Math. 94 (2003) 367–401.
[30] P.K. Moore, An incomplete assembly with thresholding algorithm for systems of reaction-diffusion equations in three dimensions: IAT for reaction-diffusion systems, J. Comput. Phys. 189 (2003) 130–158.
[31] P.K. Moore, Implicit interpolation error-based error estimation for reaction-diffusion equations in two space dimensions, Comp. Meth. Appl. Mech. Eng. 192 (2003) 4379–4401.
[32] P.K. Moore, Effects of basis selection and *h*-refinement on error estimator reliability and solution efficiency for higher-order methods in three space dimensions, Int. J. Numer. Anal. Mod. 3 (2006) 21–51.
[33] M. Pernice, H.F. Walker, NITSOL: a Newton iterative solver for nonlinear systems, SIAM J. Sci. Comput. 19 (1998) 302–318.
[34] Y. Saad, ILUT: A dual threshold incomplete LU factorization, Numer. Linear Algebra Appl. 1 (1994) 387–402.
[35] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (1986) 856–869.
[36] R. Seydel, Tutorial on continuation, Int. J. Bifur. Chaos 1 (1991) 3–11.

[37] G. Sun, M. Stynes, An almost fourth order uniformly convergent difference scheme for a semilinearly singularly perturbed reaction-diffusion problem, Numer. Math. 70 (1995) 487–500.
[38] B. Szabó, I. Babuška, Finite Element Analysis, Wiley/Interscience, New York, 1991.
[39] T. Tang, M.R. Trummer, Boundary layer resolving pseudospectral methods for singular perturbation problems, SIAM J. Sci. Comput. 17 (1996) 430–438.
[40] A. Weiser, Local-mesh, local-order, adaptive finite element methods with a posteriori error estimators for elliptic partial differential equations, Tech. Rep. 213, Dept. Computer Science, Yale University, New Haven, 1981.
[41] Z. Zhang, Finite element superconvergence on Shishkin mesh for 2-d convection-diffusion problems, Math. Comp. 72 (2003) 1147–1177.