A deferred correction approach to the solution of singularly perturbed BVPs by high order upwind methods: implementation details[1]

Pierluigi Amodio and Giuseppina Settanni

Dipartimento di Matematica, Università di Bari, I-70125 Bari, Italy

Abstract. In this note we give implementation details on the computation of the monitor function which is used inside a code based on high order upwind methods. The considered strategy is based on deferred correction and allows to compute an approximation of the error for a method of order p with essential no additional computational cost.

Keywords: Two-point Boundary Value Problems, singular perturbation problems, finite difference schemes, upwind method, deferred corrections.
PACS: 65L10, 65L50.

INTRODUCTION

Let us consider the numerical solution of second order singular perturbation problems

$$\varepsilon y'' = f(x,y,y'), \qquad x \in [a,\, b], \quad y \in \mathbb{R}, \tag{1}$$

subject to the boundary conditions

$$g(y(a),y(b)) = 0, \tag{2}$$

where f and g are sufficiently smooth functions and $\varepsilon > 0$ is a small parameter. Several codes have been already developed to solve (1)-(2): among the others we recall TWPBVP [8] and its variants [7], MIRKDC and its new implementation BVP_SOLVER [11], COLSYS [4], COLNEW, and the matlab codes TOM [9] and BVP4c [10]. Most of these codes require that the underline numerical methods are applied to a first order system and hence transform the second order equation (1) in an equivalent system with y and y' as unknowns, thus doubling the size of the discrete problem.

In [1, 2, 3] a completely different approach, based on the approximation of each derivative, has been proposed. The equation (1) is solved in its original form by using even order finite difference schemes: the second derivative by means of generalized central differences, the first derivative by means of generalized forward, central or backward differences (called GFDFs, ECDFs and GBDFs, respectively). In [1] a prototype of a matlab code has been proposed using high order upwind schemes (see also [3]), i.e., GFDFs or GBDFs depending on the sign of the coefficient of y'.

If the considered formulae have order $p > 2$, then the method approximating each derivative is built as a BVM (see [5]), namely the considered scheme (called main method) is applied to all the internal feasible points while on the first and last points of the mesh it is substituted by schemes defined on different stencils (called initial and final methods).

FINITE DIFFERENCE SCHEMES

Let

$$a = x_0 < x_1 < \cdots < x_{N+1} = b \tag{3}$$

be a mesh with $N+2$ points,the idea developed in [1, 2, 3] is that of approximating the BVP (1) in x_n, $n = 1,\ldots,N$, by considering a specific finite difference scheme (of even order) for each derivative in (1). The same result is

[1] Work developed within the project “Numerical methods and software for differential equations”.

also obtainable by approximating $y'(x_n)$ and $y''(x_n)$ with two different (i.e., based on different stencils) interpolation formulae. In particular, for the second derivative and $n = s,\dots,N-s+1$, we use the Lagrange polynomial of degree $2s$ (the order of the method) which interpolates $y(x)$ at the points $x_{n-s},\dots,x_{n+s}$,

$$q_{2s}(x) = \sum_{i=n-s}^{n+s} y_i L_i(x), \qquad \text{where} \qquad L_i(x) = \prod_{\substack{j=n-s \\ j\neq i}}^{n+s} \frac{x-x_j}{x_i-x_j}.$$

Then $y''(x_n) \approx q''_{2s}(x_n) = \sum_{i=n-s}^{n+s} \alpha_{i+s}^{(s,n)} y_i$, where $\alpha_{i+s}^{(s,n)} = L_i''(x_n)$, $i = n-s,\dots,n+s$. We remark that these coefficients depend on the stepsizes h_i, $i = n-s,\dots,n+s-1$ only in case of variable mesh. For $s > 1$ this procedure requires some adjustment in order to approximate $y''(x_i)$, for $i = 1,\dots,s-1$ and $i = N-s+2,\dots,N$. In fact, for the first $s-1$ points we always have to use the stencil $x_0,\dots x_{2s}$ while for the last points the stencil must be $x_{N-2s+1},\dots x_{N+1}$. This choice gives rise to the initial and final formulae, respectively.

For the first derivative $y'(x_n)$ we consider an analogous polynomial of degree $2s$ which interpolates $y(x)$ in the points $x_{n-s+t},\dots,x_{n+s+t}$, where $t = -1,0,1$ depends on the chosen formula among GBDF, ECDF or GFDF, respectively. Similarly to the second derivative, the coefficients of these formulae derive from the first derivative of the Lagrange polynomial computed at x_n and the approximation of $y'(x_i)$ in the first and last points of the mesh requires *ad-hoc* formulae.

In conclusion, the solution of (1)-(2) on the mesh (3) is obtained by

$$\Phi_{2s}(\mathbf{y}) = \begin{pmatrix} \varepsilon A_{2s}\mathbf{y} - f(\mathbf{x},\mathbf{y},B_{2s}\mathbf{y}) \\ g(y_0,y_{N+1}) \end{pmatrix} = 0, \tag{4}$$

where $(\mathbf{x},\ \mathbf{y})$ denotes the discrete solution, and A_{2s} and B_{2s} contain the coefficients of the formulae of order $2s$. To emphasize the relation between two methods of consecutive orders we now rewrite the previous formulae by using a Newton-like polynomial rather than the Lagrange polynomial. Always keeping in mind the approximation of $y''(x_n)$, we define

$$p_{2s}(x) = y[x_n] + \sum_{i=1}^{s} \left(y[x_{n-i},\dots,x_{n+i-1}] \prod_{j=n-i+1}^{n+i-1} \frac{x-x_j}{x_{n-i}-x_j} + y[x_{n-i},\dots,x_{n+i}] \prod_{j=n-i}^{n+i-1} \frac{x-x_j}{x_{n+i}-x_j} \right)$$

where, in case of constant stepsize, $y[x_i,\dots,x_{i+k}] = \Delta^k y_i$ is the forward difference of order l. Then

$$\begin{aligned} y''(x_n) \approx p''_{2s}(x_n) &= \frac{2y[x_{n-1},x_n,x_{n+1}]}{(x_{n+1}-x_n)(x_{n+1}-x_{n-1})} + 2\sum_{i=2}^{s} \left(\gamma_{i,1} y[x_{n-i},\dots,x_{n+i-1}] + \gamma_{i,2} y[x_{n-i},\dots,x_{n+i}] \right) \\ &= p''_{2s-2}(x_n) + 2\left(\gamma_{s,1} y[x_{n-s},\dots,x_{n+s-1}] + \gamma_{s,2} y[x_{n-s},\dots,x_{n+s}]\right) \end{aligned} \tag{5}$$

where $\gamma_{i,1} = \dfrac{1}{x_{n-i}-x_n} \displaystyle\sum_{\substack{j=n-i+1 \\ j\neq n}}^{n+i-1} \frac{1}{x_{n-i}-x_j} \prod_{\substack{r=n-i+1 \\ r\neq j,n}}^{n+i-1} \frac{x_n-x_r}{x_{n-i}-x_r}$ and $\gamma_{i,2} = \dfrac{1}{x_{n+i}-x_n} \displaystyle\sum_{\substack{j=n-i \\ j\neq n}}^{n+i-1} \frac{1}{x_{n+i}-x_j} \prod_{\substack{r=n-i \\ r\neq j,n}}^{n+i-1} \frac{x_n-x_r}{x_{n+i}-x_r}$.

The main advantage of this representation is the possibility to compute an approximation of the error term with relatively few operations. It will be particularly useful in the next section. If constant stepsize is used, then the coefficients $\gamma_{i,1} = 0$ due to symmetry while $\gamma_{i,2} = \{-1/12, 1/90, -1/560, 1/3150\}$ for the even orders form 4 to 10.

With a similar reasoning, we can obtain an approximation of $y'(x_n)$ in the form

$$y'(x_n) \approx \sum_{i=1}^{s} \left(\delta_{i,1} y[x_{n-i+t},\dots,x_{n+i-1+t}] + \delta_{i,2} y[x_{n-i+t},\dots,x_{n+i+t}] \right), \tag{6}$$

where $\delta_{i,1} = \dfrac{1}{x_{n-i+t}-x_n} \displaystyle\prod_{\substack{j=n-i+1 \\ j\neq n-t}}^{n+i-1} \frac{x_n-x_{j+t}}{x_{n-i+t}-x_{j+t}}$ and $\delta_{i,2} = \dfrac{1}{x_{n+i+t}-x_n} \displaystyle\prod_{\substack{j=n-i \\ j\neq n-t}}^{n+i-1} \frac{x_n-x_{j+t}}{x_{n+i+t}-x_{j+t}}$.

COMPUTATION OF AN ESTIMATE OF THE GLOBAL ERROR

The numerical codes for BVPs often estimate the error on a given mesh by using formulae in the same family and of two consecutive orders. The code proposed in [1, 3] is based on upwind formulae of even order from 4 to 10 and uses formulae of order $p+2$ to estimate the order p solution. Let $\mathbf{y}^{(p)}$ the solution of order p, i.e., such that $\Phi_p(\mathbf{y}^{(p)}) = 0$. Then the effective computation of the more accurate solution (which should be more expensive since the associated linear system is banded with a larger bandwidth) can be avoided by considering one step of the iterative procedure (recall that $\Phi_{p+2} = \Phi_p + e_{p+2}$)

$\mathbf{z}_0 = \mathbf{y}^{(p)}$
$i = 0$
while $\|\Phi_{p+2}(\mathbf{z}_i)\| > \text{tol}$
    compute $\mathbf{z}_{i+1}$ from $\Phi_p(\mathbf{z}_{i+1}) = -e_{p+2}(\mathbf{z}_i)$
    $i = i+1$
end

which is monotonically convergent to $\mathbf{y}^{(p+2)}$. This approach is equivalent to deferred correction and allows to compute $\mathbf{y}^{(p+2)}$ without directly solving the associated system. On the other hand, since $\mathbf{z}_1$ is a better approximation than $\mathbf{y}^{(p)}$, the difference $\mathbf{z}_1 - \mathbf{y}^{(p)}$ may be used as a good estimate of the error and, hence, as a monitor function to compute the new grid (see [1]). This means that, since the computational cost of the algorithm essentially depends on the number of factorizations and Φ_p has already been used to compute $\mathbf{y}^{(p)}$, $\mathbf{z}_1$ is computed with no additional cost. Moreover, also $e_{p+2}(\mathbf{z}_1)$ may be computed in a cheaper form by using the formulae in (5)–(6).

TEST EXAMPLE

In this section we consider an example taken from the test page [6] to show the efficiency of the proposed technique. We have updated the matlab code HOGUP, introduced in [1] to solve scalar second order singular perturbation problems including the new computation of the monitor function. The problem analyzed is the following (test problem 6 in [6]):

$$\varepsilon y'' + xy' = -\varepsilon\pi^2\cos(\pi x) - \pi x\sin(\pi x), \qquad x \in [-1,1], \tag{7}$$

with boundary conditions $y(-1) = -2$, $y(1) = 0$. Its exact solution has a shock layer in the turning point region near $x = 0$.

In (7) the coefficient of y' changes its sign (the upwind strategy requires the use of both GBDFs and GFDFs) and there is no y-term (consequently, the coefficient matrix of the generated linear system is at most weakly well conditioned).

In Figure 1 we plot for $\varepsilon = 10^{-2}$ the numerical order of accuracy of both the solution of order 8 and of $\mathbf{z}_1$ computed as explained in the previous section. The following formula is used to evaluate the order of accuracy:

$$ord(n) = \frac{\log(err(n)/err(n+10))}{\log((n+10)/n)}, \qquad n = 30,\ldots,190.$$

We have to note that, in general, the numerical order oscillates around the theoretical order p and the order of accuracy of $\mathbf{z}_1$ is only slightly better than p.

In the Figure 2 we plot the number of points required by the variable order/variable stepsize code in order to obtain an error estimate less than 10^{-8}. The code starts with order 4 and exit tolerance 10^{-2} and proceeds with (order,exit tolerance) $= (6,10^{-4}),(8,10^{-6})$ to conclude with $(10,10^{-8})$.

For $\varepsilon = 10^{-9}$ the initial solutions (with 10/20 points) are completely wrong and the meshlength is doubled. With respect to the previous version of the code we observe that order 10 may be favorably used to improve the obtained numerical solution, provided that it is sufficiently accurate.

REFERENCES

1. P. Amodio and G. Settanni, *JNAIAM J. Numer. Anal. Ind. Appl. Math.* **4**, 65–76 (2009).
2. P. Amodio and I. Sgura, *J. Comput. Appl. Math.* **176**, 59–76 (2005).
3. P. Amodio and I. Sgura, *BIT* **47**, 241–257 (2007).

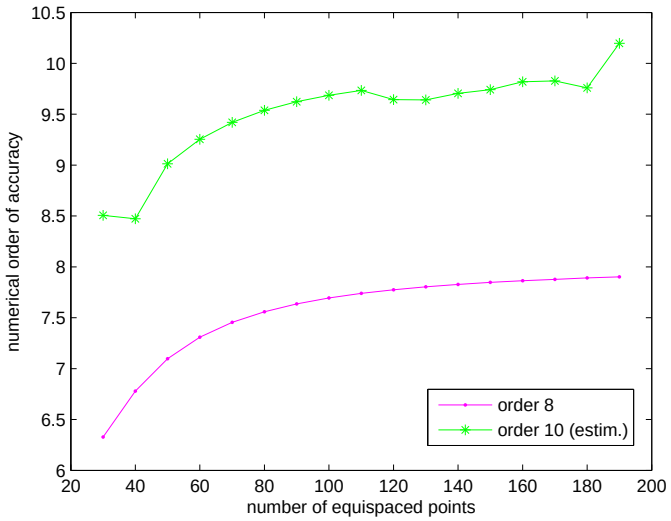

FIGURE 1. *Test problem 6 in [6] ($\varepsilon = 10^{-2}$): numerical computation of the order of accuracy.*

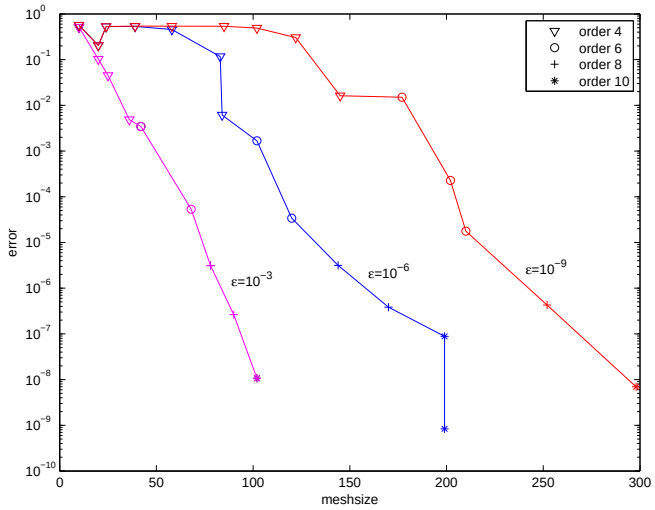

FIGURE 2. *Test problem 6 in [6]: step/order variation strategy to obtain an error less than* 10^{-8}.

4. U. Ascher, J. Christiansen and R.D. Russell, *ACM Trans. Math. Software* **7**, 223–229 (1981).
5. L. Brugnano and D. Trigiante, *Solving Differential Problems by Multistep Initial and Boundary Value Methods*, Gordon and Breach Science Publishers, Amsterdam, 1998.
6. J. Cash, `http://www.ma.ic.ac.uk/~jcash/BVP_software/readme.html`.
7. J.R. Cash and F. Mazzia, *J. Comput. Appl. Math.* **184**, 362–381 (2005).
8. J.R. Cash and M.H. Wright, `http://www.ma.ic.ac.uk/~jcash/BVP_software/twpbvp/twpbvp.pdf`.
9. F. Mazzia, `http://www.dm.uniba.it/~mazzia/bvp/index.html`.
10. L.F. Shampine, M.W. Reichelt and J. Kierzenka, available at `ftp://ftp.mathworks.com/pub/doc/papers/bvp/`.
11. L.F. Shampine, P.H. Muir and H. Xu , *J. Numer. Anal. Ind. Appl. Math.* **1**, 201–217 (2006).