*European Society of Computational Methods in Sciences and Engineering (ESCMSE)*

*Journal of Numerical Analysis, Industrial and Applied Mathematics (JNAIAM)*
vol. 4, no. 1-2, 2009, pp. 65-76
ISSN 1790–8140

# Variable Step/Order Generalized Upwind Methods for the Numerical Solution of Second Order Singular Perturbation Problems[1] [2]

**Pierluigi Amodio[3], Giuseppina Settanni[4]**

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

Received 14 March, 2009; accepted in revised form 19 April, 2009

*Abstract:* We propose a simple and quite efficient code to solve singular perturbation problems when the perturbation parameter $\epsilon$ is very small. The code is based on generalized upwind methods of order ranging from 4 to 10 and uses highly variable stepsize to fit the boundary regions with relatively few points. An extensive numerical test section shows the effectiveness of the proposed technique on linear problems.



*Keywords:* Two-point Boundary Value Problems, singular perturbation problems, finite difference schemes, upwind method, mesh variation.

*Mathematics Subject Classification:* 65L10, 65L12, 65L20

*PACS:* 02.60.Lj, 02.70.Bf

## 1 Introduction

The solution of two-point boundary value problems is one of the classical topics of numerical analysis since these problems model many real life phenomena arising, for example, from fluid mechanics, optimal control, chemical-reactor theory, reaction-diffusion process. We are here interested in the approximation of general second order ODEs

$$\mathcal{F}(x, y, y', y'') = 0, \qquad x \in [a,\ b], \tag{1}$$

subject to Dirichlet boundary conditions

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

[1] Work developed within the project "Numerical Methods and Software for Differential Equations"
[2] Published electronically June 15, 2009
[3] Corresponding author. E-mail: amodio@dm.uniba.it
[4] E-mail: settanni@dm.uniba.it

where $\mathcal{F}$ and $g$ are sufficiently smooth functions. Codes that can be used to solve (1)-(2) are TWPBVP [11] and its variants [8], MIRKDC and its new implementation BVP_SOLVER [19], COLSYS [4] and COLNEW [5] (see also [10]), and the MATLAB codes TOM [16] and BVP4c [18].

Most of the above numerical methods can only be applied to first-order equations and hence they require a rewriting of the original higher order problem (1). An alternative approach would be for finite differences to approximate each derivative separately and these would be cheaper when they are directly applied to (1)-(2). In [1] some new methods in this class have been proposed that combine generalizations of the central differences to approximate the second derivative with generalization of forward/backward/central differences for the first derivative. Following the idea inherited from Boundary Value Methods (see [6] for a review on the approach) the main feature of this approach is that of combining one main formula with additional initial and final ones. The obtained methods do not require additional boundary conditions apart from (2). Thus, methods of arbitrary high order with similar stability properties as the original two-step methods are easily derived.

In [2] the same methods have been combined in order to obtain high order generalizations of the first order upwind method and have been applied to singular perturbation problems in the form

$$\epsilon y'' = f(x, y, y'), \qquad x \in [a,\ b], \quad y \in \mathbb{R}, \tag{3}$$

where $f$ is a sufficiently smooth function and $\epsilon > 0$ is a small parameter. Due to this small parameter the solution has thin regions of fast variation (layers). This problem has been extensively studied in the past and several codes have been developed which are able to vary the stepsizes in order to compute accurate solutions even when $\epsilon \approx 0$. Among these codes we mention COLMOD (a modification of COLNEW), based on collocation methods, and ACDC (a modification of TWPBVP), based on Lobatto Runge-Kutta formulae. Both codes were developed by J. Cash and R. W. Wright and use an automatic continuation strategy to solve more difficult problems [9].

In this paper we develop a variable step, variable order approach that allows us to efficiently solve (3)-(2) when $\epsilon$ is very small, that is in cases where the usual solvers require a much larger number of points. Although the proposed strategy has only been tested on linear problems, preliminary results on nonlinear problems seem to confirm the effectiveness of both methods and, in particular, step-variation strategy.

The paper is organized as follows: in the next section we introduce the generalized upwind method and show the main features of this class of methods when variable step is used. Section 3 shows the algorithm which is used for varying the stepsize and the order. Finally, the last section is devoted to various test examples, chosen from the Test Set of Cash [7]. These have been useful to set some parameters in the code.

## 2 Generalized upwind methods

Upwind methods are classical difference schemes which have been used to solve both ordinary and partial differential equations. Historically, the idea of using a method which depends on the sign of the characteristic speeds was applied to the solution of hyperbolic PDEs (i.e., the wave equation) to numerically simulate more properly the direction of propagation of information in a flow field. For second order ODEs (1) the first derivative represents the diffusion term and needs to be treated accordingly to its sign.

In this paper the overall method is obtained by devising finite difference methods (all of even order) which are applied to each derivative appearing in the differential equation. In particular, the second derivative is approximated by centered differences while the first one is approximated by forward/backward differences (GFDFs/GBDFs) depending on the sign of $\partial f/\partial y'$ (see [2]). Let,

for example,

$$a = x_0 < x_1 < \cdots < x_{n+1} = b \tag{4}$$

be an equispaced mesh with stepsize $h = \frac{b-a}{n+1}$. To discretize each derivative we make use of the following formulae of even order $k$

$$y^{(\nu)}(x_i) \simeq y_i^{(\nu)} = \frac{1}{h^\nu} \sum_{j=-s}^{k-s} \alpha_{s+j}^{(s)} y_{i+j}, \qquad s = 1, \ldots, k-1, \tag{5}$$

where $\nu = 1, 2$, the integer $s$ depends on the choice of the number of initial conditions and the coefficients $\alpha_{s+j}^{(s)}$ are computed by imposing maximum-order conditions, that is by solving a Vandermonde linear system generated by the vector $[0 : k]$. Then, a high order generalized upwind (HOGUP) method of order $k$ is obtained by choosing for the second derivative

- $k/2 - 1$ different *initial* methods to approximate $y''(x_i)$, $i = 1, \ldots, k/2 - 1$ (we set $s = i$ in (5)),
- one *main* method to approximate $y''(x_i)$, $i = k/2, \ldots, n - k/2 + 1$ (we set $s = k/2$),
- $k/2 - 1$ different *final* methods to approximate $y''(x_i)$, $i = n - k/2 + 2, \ldots, n$ (we set $s = i - n + k - 1$),

and, for the first derivative,

- $k/2 - 1$ different *initial* methods to approximate $y'(x_i)$, $i = 1, \ldots, k/2 - 1$ (we set $s = i$),
- one method chosen between a GFDF ($s = k/2 - 1$) and an ECDF ($s = k/2$) to approximate $y'(x_{k/2})$ according to the sign of $\partial f/\partial y'$ in $x_{k/2}$ (less or greater than 0, respectively),
- the *main* method to approximate $y'(x_i)$, $i = k/2 + 1, \ldots, n - k/2$, chosen between GFDF and GBDF ($s = k/2 + 1$) according to the sign of $\partial f/\partial y'$ in $x_i$ (less or greater than 0, respectively),
- one method chosen between an ECDF and a GBDF to approximate $y'(x_{n-k/2+1})$ according to the sign of $\partial f/\partial y'$ in $x_{n-k/2+1}$ (less or greater than 0, respectively),
- $k/2 - 1$ different *final* methods to approximate $y'(x_i)$, $i = n - k/2 + 2, \ldots, n$ (we set $s = i - n + k - 1$).

In [2] it is proved that the combination of the above methods gives rise to stable approximations if used with constant stepsize to solve linear second order BVPs with constant coefficients. This means that we are able to obtain the correct behaviour of the solution with a very small number of mesh points. For well conditioned linear problems we have occasionally observed a wrong behaviour of the solution with a small number of points when the sign of the coefficient of the first derivative changes, that is, in the case where both GBDFs and GFDFs are used to discretize the first derivative. Anyway, this could happen for very small $\epsilon$ and a very small number of discretization points (less than 20 points for the order 4, 50 for the order 10). In such a situation the proposed code doubles the number of points. Conversely, in the case of ill-conditioned problems (the sign of $\partial f/\partial y$ in (3) is negative), the solution is wrong until a very small stepsize is used.

The aim of this paper is that of using variable stepsize to discretize singular perturbation problems with very small $\epsilon$. On the other hand, we do not want to waste the nice properties that these methods have when they are used with constant stepsize. Therefore, we discretize the integration interval $[a,\ b]$ by means of piecewise constant grids such that the stencil used for each

formula changes stepsize at most once. Moreover, in the first and last points we use the initial and final methods with constant stepsize.

The variable-step coefficients of the main methods

$$y''(x_{i+k/2-s}) \simeq \frac{1}{h_i^2} \sum_{j=-s}^{k-s} \tilde{\alpha}_j^{(s)} y_{i+j}, \qquad s = 1, \ldots, k-1, \tag{6}$$

$$y'(x_{i+k/2-s+t}) \simeq \frac{1}{h_i} \sum_{j=-s}^{k-s} \tilde{\beta}_j^{(t,s)} y_{i+j}, \qquad t = -1, 1, \quad s = 1, \ldots, k-1, \tag{7}$$

are still computed by solving Vandermonde linear systems generated by the vector $[-s : 0 \quad (1 : (k-s))v]$, where now $s$ represents the number of steps equal to $h_i$ (the others are equal to $h_{i+1}$) and $v = h_i/h_{i+1}$. For all these choices the coefficients have been computed algebraically. From their analysis we observe that they might change sign or increase/decrease for values of $v$ different from 1. For this reason we decided to change stepsize at least every $k+4$ points (we use 3 constant steps methods before changing the stepsize, if necessary) and to bound $v$ according to the following table

Table 1: Maximum ratio between two successive steps

| order | 4 | 6 | 8 | 10 |
|---|---|---|---|---|
| $v$ | 15 | 10 | 7 | 5 |

## 3 Stepsize and order variation strategy

The proposed algorithm combines order 4, 6, 8 and 10 HOGUP methods with piecewise constant stepsize to solve two-point linear singular perturbation problems with very small perturbation parameters $\epsilon$. The solution of nonlinear problems by means of standard techniques such as Newton's method or quasilinearization will be subject of research in the near future. Anyway, preliminary results on classical nonlinear problems seem to confirm that the HOGUP methods do not exhibit any particular problems. On the contrary, we have observed that the methods used and the stepsize variation techniques considered do not work with ill-conditioned problems, that is with problems (3) having the sign of $\partial f/\partial y$ negative. Modifications to this well-established approach will be necessary to solve this class of problems.

The choice of the order is strictly connected to the desired precision. Low orders allow us to determine the first variable meshes with a suggestion on the location of the layer and relatively few points. The computed mesh can then be used by higher orders to quickly obtain better accuracy.

The following algorithm summarizes the proposed variable order approach:

function $[x, y]$ = HOGUP ('problem', $tol$)
$ord = 4$;
$\widetilde{x} = a : (b-a)/10 : b$;
$\widetilde{y} = y_a : (y_b - y_a)/10 : y_b$;
$ltol = \max(1\text{e-}3, tol)$;
while $\|err\| < tol$
  $[x, y, err]$ = `genup` ('problem', $ord, ltol, \widetilde{x}, \widetilde{y}$);
  if $\|err\| > tol$
    $ltol = \max(\|err\|/100, tol)$;

$ord = ord + 2;$
$[\widetilde{x}, \widetilde{y}] =$ `adjmesh` $(x, y, ord + 4);$
end
end

Hence, we solve (3)-(2) starting from $n = 10$, constant stepsize and order 4. The starting mesh is updated until we obtain a solution with a computed absolute/relative error less then $10^{-3}$. This essentially means that the stepsize used is smaller than the width of the layer. Then, the process is iterated by increasing the order of the method and decreasing the exit tolerance.

It is remarkable to observe that, since we want to derive a piecewise constant mesh which depends on the order used, the vector computed for a certain order must be adapted to satisfy the restriction for the next one. This is made in `adjmesh` by increasing the number of constant points in each sub-interval with less than $ord + 4$ constant steps, or bringing together two sub-intervals with almost the same stepsize. Moreover the number of points in each sub-interval is increased in order to satisfy the restriction in Table 1.

The step variation technique is applied inside the `genup` function and allows us to compute, for fixed order *ord* and input tolerance *tol*, a numerical solution with maximum error less than *tol*. The error is approximated using the solution given by the method of order $ord + 2$ on the same mesh.

We pay particular attention to the function `monitor` which is to be found inside the function `genup` and allows us to compute the new grid $x$ starting from the old grid $\widetilde{x}$ and the computed error *err*. It is described by the following function:

function $x =$ `monitor` $(err, \widetilde{x}, ord, tol)$
$n = \text{length}(\widetilde{x}) - 1;$
$h = \widetilde{x}(2 : n) - \widetilde{x}(1 : n - 1);$
$t = \max(err(2 : n + 1), err(1 : n))\hat{}(1/ord);$
$n^* = \lfloor \|t\|_1 / tol\hat{}(1/ord) \rfloor;$
if $n\|t\|_\infty / \|t\|_1 \leq 1.2$ & $n^* \geq 2 \cdot n$
$x$ is obtained halving the step-length vector $h$
else
$n^* = \max(\min(n^*, \lfloor 1.2 \cdot n \rfloor), \lfloor n/1.2 \rfloor);$
$I = [0 \ \text{cumsum}(t)]/\|t\|_1;$
$z = 0 : 1/n^* : 1;$
$\widehat{x} =$ `linear_interp` $(I, \widetilde{x}, z);$
$x =$ `piecewise_grid` $(\widehat{x}, ord + 4);$
end

The function `monitor` is based on an equidistribution of the vector $t$ that contains the error in each step raised to the power of $1/ord$. With this aim, starting from $t$, we derive an ordered vector in the range $[0,\ 1]$ and equidistribute by means of inverse linear interpolation. The new number of points is chosen in the range $[n/1.2,\ 1.2n]$ and is originally set equal to $\left\lfloor \|t\|_1 / (tol)^{1/ord} \right\rfloor$. Then, the obtained grid is modified by `piecewise_grid` in order to have piecewise constant steps.

The function `piecewise_grid` starts from the minimum step and determines $ord+4$ consecutive constant steps which are able to overlay exactly some steps of the previous grid. This procedure is iterated on the remaining part of the grid in order that the new steps are greater than or equal to those already computed and the ratio of two consecutive steps is bounded by the values in Table 1.

## 4 Numerical examples

In this section we give some results on the convergence behavior of the matlab code HOGUP on four singular perturbation problems contained in the "BVP software page" of J. Cash [7], and we compare these with analogous results of the packages ACDC and COLMOD (see [7]). We stress that a fair comparison is not possible because the HOGUP code is based on multistep methods and hence the number of points exactly represents the size of the problem solved. The computational cost of the other two codes also depends on other parameters such as, for example, the order of the methods used, which are not reflected in the number of points. As a matter of fact, even if the package COLMOD always needs a number of mesh points lower than ACDC, its execution time is higher for the largest values of $\epsilon$. Finally, we recall that COLMOD needs to be applied to first order problems, and therefore its number unknowns should be doubled.

For our numerical experiments we have chosen a uniform initial mesh with 10 points and $tol = 10^{-8}$. Moreover, we have used methods of order 4, 6 and 8 for the order variation strategy. We have set the maximum number of allowed mesh points to 1500 and computed an approximation of the maximum relative error in the numerical solution by means of two successive orders and the formula

$$\max_{1\le i\le n} \frac{|y_i^{(p)} - y_i^{(p+2)}|}{1 + |y_i^{(p+2)}|}.$$

In the following Tables 2-5 we report the meshlength required by the codes HOGUP, ACDC and COLMOD for $\epsilon$ ranging from $10^{-1}$ to $10^{-10}$. In Figures 1-4 we report the number of points required by the order/step variation strategy for $\epsilon = 10^{-3}, 10^{-6}, 10^{-9}$. On the $y$ axis we show the exact maximum relative error given by the formula

$$\max_{1\le i\le n} \frac{|y_i^{(p)} - y_e(x_i)|}{1 + |y_e(x_i)|},$$

where $y_e$ represents the exact solution.

*Problem 1* Let us consider the test problem 4 in [7]

$$\epsilon y'' + y' - (1+\epsilon)y = 0, \qquad x \in [-1, 1],$$

with boundary conditions $y(-1) = 1 + \exp(-2)$, $y(1) = 1 + \exp(-2(1+\epsilon)/\epsilon)$. The exact solution

$$y_e(x) = \exp(x-1) + \exp\left(-\frac{1+\epsilon}{\epsilon}(1+x)\right).$$

has a boundary layer of width $O(\epsilon)$ at $x = -1$.

From Table 2 we deduce that the number of meshpoints of the HOGUP code is two/three times higher than ACDC and COLMOD for almost all values of $\epsilon$. ACDC does not work for $\epsilon = 10^{-9}, 10^{-10}$ while the value of COLMOD is not available for $\epsilon = 10^{-10}$.

In Fig. 1 we plot the behaviour of the HOGUP code for some values of $\epsilon$. Only few steps of the orders 6/8 are necessary to reach the requested tolerance when the order 4 method gives a solution with $err < 10^{-3}$. Nevertheless, the number of steps required by the order 4 method is 7, 13 and 17 for the three tests, that is to obtain a stepsize smaller than $O(\epsilon)$ in the layer. For $\epsilon = 10^{-9}$ COLMOD requires 9 continuation steps and the computation of 20 mesh (the same as HOGUP), but with half of the total number of points. ACDC has similar results, but with the same number of total points as HOGUP.

Table 2: Test problem 1: meshlength to obtain a solution with an error less than $10^{-8}$.

| $\epsilon$ | HOGUP | ACDC | COLMOD |
|---|---|---|---|
| $10^{-1}$ | 76 | 34 | 43 |
| $10^{-2}$ | 84 | 42 | 56 |
| $10^{-3}$ | 151 | 49 | 60 |
| $10^{-4}$ | 155 | 50 | 60 |
| $10^{-5}$ | 161 | 66 | 72 |
| $10^{-6}$ | 237 | 93 | 81 |
| $10^{-7}$ | 284 | 93 | 81 |
| $10^{-8}$ | 291 | 107 | 101 |
| $10^{-9}$ | 346 | 700 | 113 |
| $10^{-10}$ | 437 | 4311 | – |

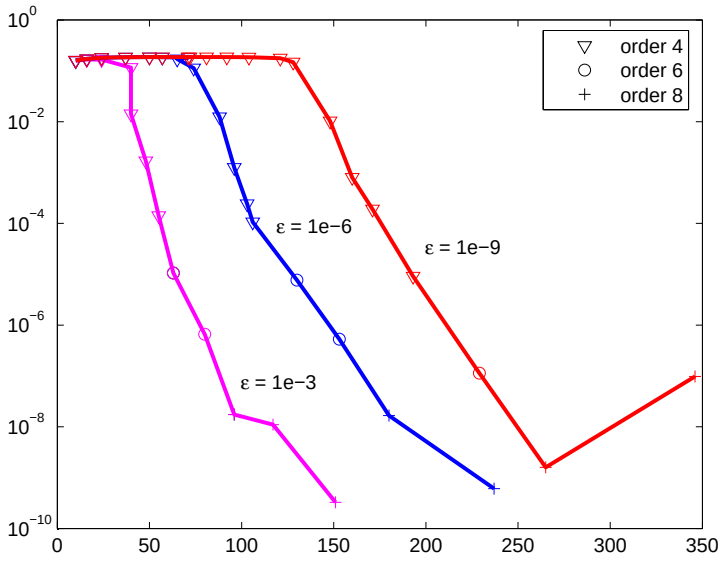


Figure 1: *Test problem 1: step/order variation strategy to obtain an error less than* $10^{-8}$.

*Problem 2* Let us consider the test problem 6 in [7]

$$\epsilon y'' + xy' = -\epsilon\pi^2 \cos(\pi x) - \pi x \sin(\pi x), \qquad x \in [-1, 1],$$

with boundary conditions $y(-1) = -2$, $y(1) = 0$. The exact solution

$$y_e(x) = \cos(\pi x) + \frac{\mathrm{erf}(x/\sqrt{2\epsilon})}{\mathrm{erf}(1/\sqrt{2\epsilon})}$$

has a shock layer in the turning point region near $x = 0$.

Here the coefficient of $y'$ changes its sign and there is no $y$-term. As noted in Section 2, the initial solutions (with 10/20 points) are different from the theoretical one, and the meshlength doubles in the first two or three steps. The number of points required by HOGUP is comparable

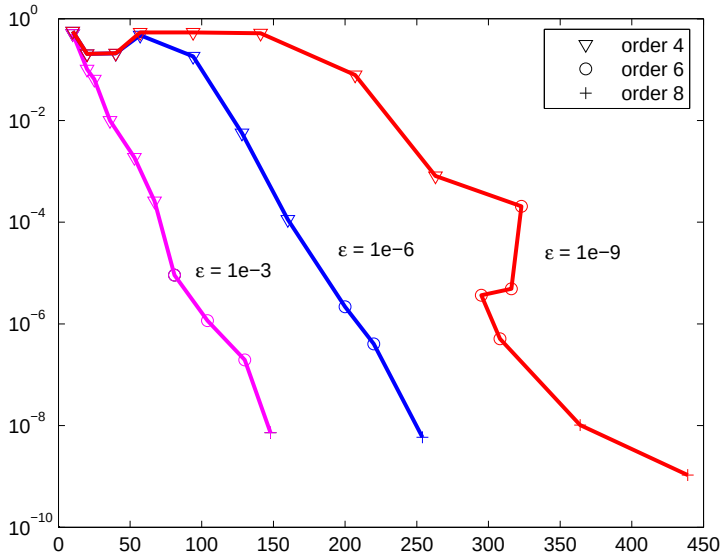


Figure 2: *Test problem 2: step/order variation strategy to obtain an error less than* $10^{-8}$.

to ACDC, but the latter code does not work for $\epsilon = 10^{-9}$. COLMOD requires a smaller number of points, but from $10^{-1}$ to $10^{-8}$ it requires an execution time higher than ACDC.

Table 3: Test problem 2: meshlength to obtain a solution with an error less than $10^{-8}$.

| $\epsilon$ | HOGUP | ACDC | COLMOD |
|---|---|---|---|
| $10^{-1}$ | 52 | 33 | 46 |
| $10^{-2}$ | 111 | 79 | 81 |
| $10^{-3}$ | 148 | 111 | 100 |
| $10^{-4}$ | 202 | 127 | 112 |
| $10^{-5}$ | 325 | 153 | 118 |
| $10^{-6}$ | 254 | 195 | 120 |
| $10^{-7}$ | 302 | 394 | 122 |
| $10^{-8}$ | 408 | 402 | 128 |
| $10^{-9}$ | 439 | 458 | 134 |
| $10^{-10}$ | 505 | 1153 | 178 |

From Fig. 2 we observe that the convergence behaviour of the HOGUP method is fairly good in all cases. For $\epsilon = 10^{-9}$ the mesh given by the order 4 method has been completely reassembled and reduced in length by the order 6 method. The number of computed meshes for each $\epsilon$ is up to 14, that is, almost the same number required by the other codes.

*Problem 3* Let us consider the test problem 7 in [7]

$$\epsilon y'' + xy' - y = -(1 + \epsilon\pi^2)\cos(\pi x) - \pi x \sin(\pi x), \qquad x \in [-1, 1],$$

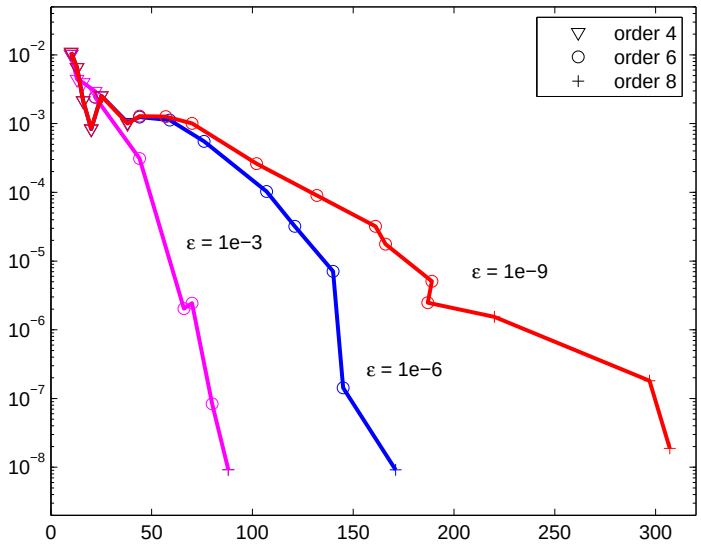


Figure 3: *Test problem 3: step/order variation strategy to obtain an error less than* $10^{-8}$.

with boundary conditions $y(-1) = -1$, $y(1) = 1$. The exact solution

$$y_e(x) = \cos(\pi x) + x + \frac{x\,\mathrm{erf}(x/\sqrt{2\epsilon}) + \sqrt{2\epsilon/\pi}\exp(-x^2/2\epsilon)}{\mathrm{erf}(1/\sqrt{2\epsilon}) + \sqrt{2\epsilon/\pi}\exp(-1/2\epsilon)}$$

has a corner layer in the turning point region near $x = 0$.

From Table 4 we deduce similar results as the previous example. The only differences are that the meshlength of ACDC becomes higher than that of HOGUP for $\epsilon < 10^{-6}$ and COLMOD is able to require up to 1/4 of the number of points of HOGUP.

From Fig. 3 we derive that order 4 method computes a solution with the requested tolerance with relatively few steps while the order 6 method requires some additional steps. For $\epsilon = 10^{-9}$, order 8 method seems to increase too much the number of points to reach $tol = 10^{-8}$.

Table 4: Test problem 3: meshlength to obtain a solution with an error less than $10^{-8}$.

| $\epsilon$ | HOGUP | ACDC | COLMOD |
|---|---|---|---|
| $10^{-1}$ | 40 | 40 | 45 |
| $10^{-2}$ | 77 | 56 | 58 |
| $10^{-3}$ | 88 | 81 | 68 |
| $10^{-4}$ | 121 | 100 | 70 |
| $10^{-5}$ | 144 | 116 | 70 |
| $10^{-6}$ | 171 | 204 | 70 |
| $10^{-7}$ | 201 | 300 | 72 |
| $10^{-8}$ | 252 | 464 | 76 |
| $10^{-9}$ | 307 | 728 | 76 |
| $10^{-10}$ | 321 | 1615 | 79 |

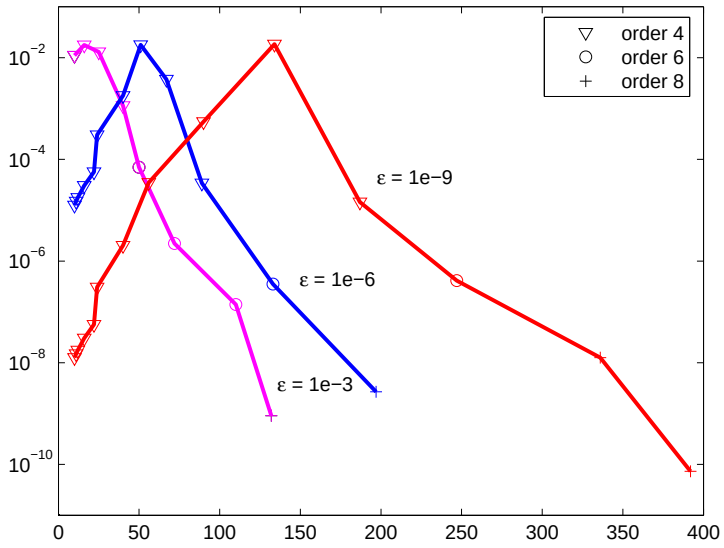


Figure 4: *Test problem 4: step/order variation strategy to obtain an error less than* $10^{-8}$.

*Problem 4* Let us consider the test problem 14 in [7]

$$\epsilon y'' - y = -\left(\epsilon\pi^2 + 1\right)\cos(\pi x), \qquad x \in [-1, 1],$$

with boundary conditions $y(-1) = y(1) = \exp(-2/\sqrt{\epsilon})$. The exact solution

$$y_e(x) = \cos(\pi x) + \exp\left((x-1)/\sqrt{\epsilon}\right) + \exp\left(-(x+1)/\sqrt{\epsilon}\right)$$

has boundary layers of width $O(\sqrt{\epsilon})$ near $x = -1$ e $x = 1$.

This is the only problem considered with two boundary layers, but the results obtained trace out those of the previous example (here the number of points of COLMOD is halved with respect to HOGUP). The only thing to note is that for small $\epsilon$ few points are sufficient to compute an

Table 5: Test problem 4: meshlength to obtain a solution with an error less than $10^{-8}$.

| $\epsilon$ | HOGUP | ACDC | COLMOD |
|---|---|---|---|
| $10^{-1}$ | 41 | 29 | 45 |
| $10^{-2}$ | 84 | 49 | 69 |
| $10^{-3}$ | 132 | 91 | 99 |
| $10^{-4}$ | 150 | 123 | 115 |
| $10^{-5}$ | 192 | 151 | 124 |
| $10^{-6}$ | 197 | 216 | 130 |
| $10^{-7}$ | 248 | 265 | 142 |
| $10^{-8}$ | 304 | 311 | 143 |
| $10^{-9}$ | 392 | 521 | 158 |
| $10^{-10}$ | 419 | 768 | 188 |

accurate solution because there are no points near the layer. In the step variation strategy, the error increases when the number of points increases and decreases when there are points inside the layer.

## 5 Conclusion

We have proposed a new code for solving second order singular perturbation problems. With reference to linear problems, we can conclude that the code is better than ACDC for small values of $\epsilon$ on the problems we have considered. COLMOD always requires small meshlengths, but its comparison with ACDC in terms of execution time allows us to deduce that its computational cost for the same $n$ is much higher.

The two main objectives of this research in the near future will be to extend this approach to the solution of nonlinear problems and to provide a Fortran implementation of the code. Both will be necessary to effectively compare the codes considered in this paper.

## References

[1] P. Amodio and I. Sgura, *High-order finite difference schemes for the solution of second-order BVPs*, J. Comput. Appl. Math. **176** (2005), 59–76.

[2] P. Amodio and I. Sgura, *High order generalized upwind schemes and numerical solution of singular perturbation problems*, BIT **47** (2007), 241-257.

[3] U.M. Ascher, R.M.M. Mattheij and R.D. Russell, *Numerical Solution of Boundary Value Problems for ODEs*, Classics in Applied Mathematics **13**, SIAM, Philadelphia, 1995.

[4] U. Ascher, J. Christiansen and R.D. Russell, *Algorithm 569: COLSYS: Collocation Software for Boundary-Value ODEs*, ACM Trans. Math. Software **7** (1981), 223–229.

[5] G. Bader and U. Ascher, *A new basis implementation for a mixed order boundary value ODE solver*, SIAM J. Sci. Statist. Comput. **8** (1987), 483–500.

[6] L. Brugnano and D. Trigiante, *Solving Differential Problems by Multistep Initial and Boundary Value Methods*, Gordon and Breach Science Publishers, Amsterdam, 1998.

[7] J. Cash, *BVP Software web page*, `http://www.ma.ic.ac.uk/~jcash/BVP_software/readme.html`.

[8] J.R. Cash and F. Mazzia, *A new mesh selection algorithm, based on conditioning, for two-point boundary value codes*, J. Comput. Appl. Math. **184** (2005), 362–381.

[9] J.R. Cash, G. Moore and R.W. Wright, *An automatic continuation strategy for the solution of singularly perturbed linear two-point boundary value problems*, J. Comput. Phys. **122** (1995), 266-279.

[10] J.R. Cash and M.H. Wright, *A deferred correction method for nonlinear two-point boundary value problems: implementation and numerical evaluation*, SIAM J. Sci. Statist. Comput. **12** (1991), 971-989.

[11] J.R. Cash and M.H. Wright, *Users Guide for TWPBVP: A Code for Solving Two-Point Boundary Value Problems*, `http://www.ma.ic.ac.uk/~jcash/BVP_software/twpbvp/twpbvp.pdf`.

[12] J.R. Cash and R.H. Wright, *User's guide for ACDC: An automatic continuation code for solving stiff two-point boundary value problems*, available at
`http://www.ma.ic.ac.uk/~jcash/BVP_software/readme.html`.

[13] W.H. Enright and P.H. Muir, *Runge–Kutta software with defect control for boundary value ODEs*, SIAM J. Sci. Comput. **17** (1996), 479-497.

[14] E.C. Gartland, *Uniform high-order difference schemes for a singularly perturbed two-point boundary value problem*, Math. Comp. **48** (1987), 551–564.

[15] R. Lynch and J.R. Rice, *A high-order difference method for differential equations*, Math. Comp. **34** (1980), 333–372.

[16] F. Mazzia, *BVP Software web page*,
`http://www.dm.uniba.it/~mazzia/bvp/index.html`.

[17] H.G. Roos, M. Stynes and L. Tobiska, *Numerical Methods for Singularly Perturbed Differential Equations: Convection-Diffusion and Flow Problems*, Springer-Verlag, Berlin, 1996.

[18] L.F. Shampine, M.W. Reichelt and J. Kierzenka, *Solving Boundary Value Problems for Ordinary Differential Equations in MATLAB with bvp4c*, available at
`ftp://ftp.mathworks.com/pub/doc/papers/bvp/`.

[19] L.F. Shampine, P.H. Muir and H. Xu, *A user-friendly Fortran BVP solver*, J. Numer. Anal. Ind. Appl. Math. **1** (2006), 201–217.