ANNALI DELL'UNIVERSITA' DI FERRARA
https://doi.org/10.1007/s11565-022-00440-7

Mesh selection strategies of the code TOM for Boundary Value Problems

Francesca Mazzia[1]

Received: 19 May 2022 / Accepted: 16 August 2022

Abstract
This paper presents new hybrid mesh selection strategies for boundary value problems implemented in the code TOM. Originally the code was proposed for the numerical solution of stiff or singularly perturbed problems. The code has been now improved with the introduction of three classes of mesh selection strategies, that can be used for different categories of problems. Numerical experiments show that the mesh selection and, in the nonlinear case, the strategy for solving the nonlinear equations are determinant for the good behaviour of a general purpose code. The possibility to choose the mesh selection should be considered for all general purposes codes to make them suitable for wider classes of problems.

Keywords Mesh selection · Boundary value problems · Boundary value methods

Mathematics Subject Classification 65L04 · 65L06 · 65L10 · 65D07

1 Introduction

The development of a general purpose code for the solution of two point Boundary Value Problems (BVPs) described by

This paper is dedicated to prof. Ilio Galligani. I met him because he was a great friend of prof. Donato Trigiante. They were the first numerical analysts in Italy to promote mathematical research in applicative fields and, in particular, support the development of scientific software for industrial applications. The code TOM, whose first release was written in collaboration with Donato Trigiante, is one example of the results of their efforts.

✉ Francesca Mazzia
francesca.mazzia@uniba.it

[1] Department of Computer Science, Università degli Studi di Bari Aldo Moro, Via Orabona 4, Bari 70125, Italy

Published online: 30 August 2022

$$\begin{aligned} y' &= f(x, y), \qquad a \le x \le b \\ g(y(a), y(b)) &= 0 \end{aligned} \tag{1}$$

with $f(x, y) \in \mathbb{R}^m$ and g continuous functions, requires to pay a special attention to: the error estimation, the mesh selection strategy and the method for solving the nonlinear equations. Codes based on the same numerical scheme show completely different behaviour depending on how these aspects are exploited. An interesting work related to the choice of the error estimation used in the mesh selection strategy has been made in [1] where the authors show that properly changing the mesh selection, changes the behavior of the code and its efficiency. In [1] three strategies for estimating the global error are compared: the one based on Richardson Extrapolation (RE), the one based on the use of a higher order formula (HO), the one based on the use of deferred correction (DC). Additionally, the authors consider also a global error bound based on the estimation of the conditioning constant and the defect. The mesh selection and the method for solving the nonlinear problems remain essentially the same, but the final results differ. The conclusion is that the HO and DC strategies give the best results compared to RE. The authors suggest that the best is to have both the strategies available and to leave to the user the choice. The last release of the code BVP_SOLVER includes four different error estimation routines. A comparison of the efficiency of the code changing the mesh selection has been made for the code TOM in [2] where an hybrid mesh selection strategy based on conditioning has been analyzed and in [3, 4] where the mesh selection introduced in [2] has been adapted to the Fortran codes TWPBVP and TWPBVPL. The last release of these codes allow to choose two different mesh selections: the hybrid one based on conditioning suggested for stiff or singularly perturbed problems and the one based on the estimation of the global error using a deferred correction strategy. In [5] the Fortran codes have been translated in Matlab and the mesh selection based on conditioning has been added to the code ACDC, that uses an automatic continuation strategy. We observe that the error estimation and the strategy for solving the nonlinear equations have not been changed in all the codes.

The mesh selection strategy needs a special attention when the problem is considered stiff or singularly perturbed. For these classes of problems many studies have been carried out, often associated to the definition of new numerical methods. In this case the solution presents regions of rapid variation that can be located at the boundary (boundary layers) or in any points internal to (a, b) (turning points). In particular we recall the Shishkin mesh and the Bakhvalov mesh strategies, that have special properties of uniform convergence [6, 7]. These strategies are effective when the location and the width of the layer are known, even if recently, adaptive meshes for weakly coupled system of singularly perturbed delay differential equations have been proposed in [8]. Robust hybrid methods of higher order for singularly perturbed convection-diffusion problems have been proposed in [9]. An efficient automatic grid control strategy has been proposed in [10] and implemented in the code bvpsuite, a Matlab code designed for the numerical solution of singular boundary value problems [11], which is the new version of the code sbvp [12]. The authors compute the grid density function used for the mesh variation with a feedback control law that adjusts the grid. The importance of generating a smooth mesh is highlighted in [13] were the authors develop

a semi-analytic technique called W-grids for generating a suitable smooth mesh for singularly perturbed boundary value problems. The advantages of these meshes over the piecewise uniform Shishkin meshes are also demonstrated with computational experiments.

Recently a comparison of some of the codes available in Matlab for the solution of optimal control problems has been made in [14]. General purpose codes for BVPs are used for the solution of many problems in applications see, for example, [15–17] for optimal control problems and path planning, [18–20] for numerical simulation, and [21, 22] for analysis of bifurcation. General purpose codes are really useful for many real world problems. The users just want to find the solution to their model using functions available in a problem solving environment, like Matlab, Python and R. The R package bvpSolve [23, 24], allows the use of the Fortran codes TWPBVP, TWPBVPL and ACDC, and of the collocation codes colsys, colnew and colmod [25, 26]. The Matlab environment includes the codes bvp4c, bvp5c and the codes bvptwp [5] and bvpsuite [11]. A collection of Fortran and Fortran 90 codes is available in the website "Testset for BVP solvers", together with many numerical examples [27].

The aim of this paper is to describe the mesh selection strategies that have been included in the code TOM, with the main purpose to make the code comparable in time with the other available codes when the problem is not stiff. Moreover, the computation of the conditioning constants has been modified and theoretically justified, making the code more robust. The paper is structured as follows. In Sect. 2 we review the code TOM, the error estimation criteria and the conditioning parameters used in the mesh selection. In Sect. 3 we present the new mesh selection strategies and in Sect. 4 the method used for solving nonlinear problems. In Sect. 5 we show some numerical experiments with a discussion, comparing the results with some of the available Matlab codes.

2 The code TOM

We first describe the code TOM for the solution of the linear problem

$$\begin{aligned} &y' = J(x)y + v(x), && a \le x \le b \\ &C_a y(a) + C_b y(b) = w \end{aligned} \tag{2}$$

where $J(x)$, $v(x)$ are respectively $\mathbb{R}^{m\times m}$ and $\mathbb{R}^m$ continuous functions, $w \in \mathbb{R}^m$, $C_a, C_b \in \mathbb{R}^{m\times m}$ are constant matrices related to the boundary conditions. For linear problems the solution can be easily defined using the fundamental solution matrix $Y(x)$, the Green's function $G(x,t)$ and the nonsingular matrix $Q = C_a Y(a) + C_b Y(b)$ as:

$$y(x) = Y(x)Q^{-1}w + \int_a^b G(x,t)v(t)dt, \tag{3}$$

and the continuous conditioning parameters are defined by

$$\kappa_1 = \max_{a\le x\le b} \|Y(x)Q^{-1}\|, \kappa_2 = \sup_{a\le x\le b} \int_a^b \|G(x,t)\|dt, \tag{4}$$

where κ_1 is related to the perturbation of the boundary conditions and κ_2 is related to the perturbation of the differential equations. The following parameter takes into account both the perturbations

$$\kappa = \max_{a\le x\le b} \left(\|Y(x)Q^{-1}\| + \int_a^b \|G(x,t)\|dt \right) \tag{5}$$

see [25, 28, 29] for a description of the conditioning of linear problems. We also observe that it is possible to compute the conditioning using a different norm and in particular with the scaled 1-norm the parameter corresponding to κ_1 is

$$\gamma_1 = \frac{1}{b-a} \int_a^b \|Y(x)Q^{-1}\|dx. \tag{6}$$

Moreover we can have information about the stiffness by computing a stiffness ratio related to the conditioning parameters computed in two different norms:

$$\sigma = \max_{w, w\neq 0} \frac{\max_{a\le x\le b} \|Y(x)Q^{-1}w\|}{\frac{1}{b-a}\int_a^b \|Y(x)Q^{-1}w\|dx}. \tag{7}$$

The approximation of the conditioning parameters in both norms is one of the main characteristic of the hybrid mesh selection algorithm used in the code TOM and in the code bvptwp. The procedure used in bvptwp has been described in detail in [30] for one step methods. The numerical methods used in the code TOM are however linear multistep methods, so the algorithm needs to be adapted for the computation of κ and κ_2.

The linear multistep methods used as boundary value methods implemented in the code TOM are all symmetric formulae and in particular are in the class of Top Order Methods (TOM), Extended Trapezoidal Rules (ETR), Extended Trapezoidal Rules of second type (ETR2) and BS methods (BS) [31, 32]. The last class turns out to be particularly interesting because generates a collocation method using the B-spline basis. For all the class of methods the discretization of the linear problem (2) with a method of order p generates a linear system of the following form:

$$M_p\, \mathbf{y}^{(p)} = \mathbf{e}_1 \otimes w + \mathcal{B}_p \mathbf{v}, \tag{8}$$

where, given $\pi = \{a = x_0, x_1, \ldots, x_N = b\}$ the discretization of $[a, b]$, $\mathbf{y}^p := (\mathbf{y}_0^{pT}, \ldots, \mathbf{y}_N^{pT})^T$ is the vector containing the approximation of the solution at the mesh points, $\mathbf{e}_1 := (1, 0, \cdots, 0)^T \in \mathbb{R}^{N+1}$, $\mathbf{v} := (v(x_0)^T, \ldots, v(x_N)^T)^T$ and

$$M_p := \mathcal{A}_p - \mathcal{B}_p\, \mathcal{J} \tag{9}$$

with $\mathcal{J} := diag(J(x_0), \ldots, J(x_N))$ and

$$\mathcal{A}_p := \begin{pmatrix} (C_a,\ 0_{m,m(N-1)},\ C_b) \\ A_p \otimes I_m \end{pmatrix}, \qquad \mathcal{B}_p := \begin{pmatrix} 0_{m,m(N+1)} \\ (H B_p) \otimes I_m \end{pmatrix}, \tag{10}$$

where, we denote by $h_i = x_i - x_{i-1}$ for any $x_i \in \pi$, $H := \mathrm{diag}(h_1, \cdots, h_N)$ and A_p and B_p are banded matrices of size $N \times (N+1)$ whose entries are all the coefficients of the linear multistep methods involved. The number of elements different from zero in each rows are $k+1$, where k is the number of steps of the methods. The linear multistep method is used with $k_1 < k$ initial and $k_2 = k - k_1$ final conditions and the matrices have a band structure with k_1 lower diagonals, except for the first k_1 rows and the last k_2 rows, where the number of elements different from zero is $k+1$.

The BS method has a spline continuous extension computed using the algorithm presented in [32] so that the numerical solution satisfies the differential equation at the mesh points. The same continuous extension defines a quasi-interpolation method that, in this release of the code, is used as a continuous extension for the other methods [33]. The coefficients of the BS-methods and of the corresponding quasi-interpolation are computed using the Matlab interface of the QIBSH++ library [34].

Given s_p the BS Hermite spline quasi-interpolant associated to $(x_i, y_i^p, J(x, y_i^p) y_i^p + v(x_i))$, $i = 0, N$, the defect is defined as

$$r_p(x) := s_p'(x) - J(x, s_p(x)) s_p(x) - v(x) \tag{11}$$

and it is estimated using only its evaluation at the mesh points:

$$(r_p)_i := s_p'(x_i) - J(x_i, \mathbf{y}_i)\mathbf{y}_i - v(x_i). \tag{12}$$

We estimate the local truncation error using a method of higher order $q = p+2$ as

$$\tau_p := M_q\, \mathbf{y}^{(p)} - \mathbf{e}_1 \otimes w + \mathcal{B}_q \mathbf{v}, \tag{13}$$

and we estimate the absolute global error as $\mathbf{e}_E = |\mathbf{y}^{(p)} - \mathbf{y}^{(q)}|$ where $\mathbf{y}^{(q)}$ is computed by using an approximated deferred correction method solving the linear system:

$$M_p\, \mathbf{y}^{(q)} - \mathbf{e}_1 \otimes w + \mathcal{B}_p \mathbf{v} = \tau_p, \tag{14}$$

Depending on the user choice, the code computes either the defect, the local truncation error, or the global error. We consider as approximation of the absolute global error one of the following:

- $(\mathbf{e}_R)_i := max(h_i, h_{i-1}) |s_p'(x_i) - J(x_i, \mathbf{y}_i)\mathbf{y}_i - v(x_i)|$;
- $(\mathbf{e}_T)_i := h_i^{-1/2} |(\tau_p)_i|$;
- $(\mathbf{e}_E)_i$.

We have suitably scaled the defect and the truncation error in order to make the related error comparable with the global error. The code tries to compute a solution with the

mixed componentwise relative approximation of the error that satisfies the following inequality

$$E^Z = \max_{0\le i\le N} \left\| \frac{(\mathbf{e}_Z)_i}{\max(atol/rtol, \min(|\mathbf{y}_i^p|, |\mathbf{y}_i^q|))} \right\|_\infty \le rtol \tag{15}$$

where Z is R, T or E depending on the approximation used to estimate the error, $atol$, a vector of m entries, and $rtol$ are the requested absolute and relative tolerances and the operators are defined componentwise.

The estimation of the conditioning parameters need some care. The value of κ_1 can be calculated just computing the first m columns of the inverse of M_p, since it depends only on the initial conditions. The values of κ and κ_2 are instead related to the differential problems and they involve all the other block columns of the inverse. For one-step methods it has been proved that the approximation of κ is given by $\|M_p^{-1}\|_1$ [25]. In [29] the authors have defined an efficient algorithm for computing κ, κ_1 and κ_2.

In order to have a good approximation of the Green function for boundary value methods we scale the linear system (17) by the inverse of the following matrix:

$$\mathcal{S}_p := \begin{pmatrix} I_m & 0_{m,mN} \\ O_{mN,m} & (HB_p^1) \otimes I_m \end{pmatrix} \tag{16}$$

where the matrix $B_p = (B_p^0, B_p^1)$ with B_p^0 denoting the first column of the matrix and B_p^1 all the other columns. The inverse of this function exists for quasi uniform meshes since the methods are symmetric and A_{k_1,k_2}-stable, that is the stability region is equal to the negative half plane (see [31]). After the scaling, the resulting linear problem is:

$$\mathcal{S}_p^{-1} M_p \mathbf{y}^{(p)} = \mathbf{e}_1 \otimes w + ((0, ((B_p^1)^{-1} B_p^0)^T)^T \otimes I_m) v(x_0) + I_e \mathbf{v}, \tag{17}$$

where $I_e = diag(0 \otimes I_m, I_{Nm})$.

Theorem 1 *If the boundary value problem (2) is discretized by a symmetric BVMs, then*

$$\|M_p^{-1} S_p\|_1 = \|\Omega\|_1 \approx \kappa$$

Proof The proof follows by considering that, denoting by $\Omega_{i,j}, 0 \le i, j \le N$ the blocks of size m of the matrix Ω, the numerical solution is given by:

$$\mathbf{y} = \Omega_{:,0} w + \Omega_{:,1:N} v_{1:N} + M_p^{-1} \mathcal{S}_p ((0, ((B_p^1)^{-1} B_p^0)^T)^T \otimes I_m) v(x_0)$$

and considering that $\mathcal{S}_p((0, ((B_p^1)^{-1} B_p^0)^T)^T \otimes I_m) = ((0, (HB_p^0)^T)^T \otimes I_m)$, the vector B_0 has only $k_1 + 1$ elements different from zero, so only $k_1 + 1$ columns of M_p^{-1} are involved. Hence, we conclude that $\|M_p^{-1}((0, (HB_p^0)^T)^T \otimes I_m) v(x_0)\| \le hK$,

with K a constant that does not depend on N and $h = \max h_i$, therefore,

$$\mathbf{y} = \Omega_{:,0} w + \Omega_{:,1:N} v_{1:N} + O(h).$$

This relation shows that the matrix Ω has the same role of the fundamental matrix and of the Green function for the discrete solution and its 1-norm is an $O(h)$ approximation of κ. □

Using the results in Theorem 1 we can approximate the value of κ, κ_1 and κ_2 with an algorithm similar to the one used in [29]. If we have not computed the first block column of the inverse we propose, however, a different initialization of the algorithm, saving computational time. This initialization is based on the approximation of the conditioning parameters presented in the next section.

3 Mesh selection strategies

In this section we introduce the mesh selection strategies that are possible to use in the code TOM. The numerical experiments will show that are all useful, depending on the problem under consideration. The variants will be denoted by taking a combination of three or four letters, the first two letters will be HS (High stiff), MS (Middle Stiff), NS (Non Stiff) and give information to the class of problems for which the mesh is suggested. If we compute the conditioning parameters by computing the first block column of the inverse we add the letter (C). If we estimate them using the computed solution we add the letter (S). The three new parameters computed using the approximation $\mathbf{y}$ of the solution in the current mesh π are called κ_y, γ_y, σ_y. We have

$$\kappa_y := \max_{i=0,N} \|\mathbf{y}_i\|_\infty \tag{18}$$

$$\gamma_y := \max_{1\le j\le m} \frac{1}{b-a} \sum_{i=1}^{N} h_i \max(|\mathbf{y}_{i-1,j}|, |\mathbf{y}_{i,j}|) \tag{19}$$

$$\sigma_y := \max_{1\le j\le m,} \left(\frac{\max_{0\le i\le N} \max(|\mathbf{y}_{i-1,j}|, |\mathbf{y}_{i,j}|)}{\frac{1}{b-a}\sum_{i=1}^{N} h_i \max(|\mathbf{y}_{i-1,j}|, |\mathbf{y}_{i,j}|)} \right). \tag{20}$$

The parameter κ_y gives indication about the maximum norm of the approximated solution. The parameter γ_y gives information about the integral of the absolute value of the functions associated to each components of $\mathbf{y}$. If the values of κ_y and γ_y with different meshes are stabilized, then we have a good preliminary approximation of the solution and a good mesh. The value of σ_y gives indication about the stiffness of the problem: it is the ratio between the maximum absolute value of each component of the approximated solution and the corresponding approximated scaled integral. This value is very high if the solution has rapid variation. In the mesh selection κ_y, γ_y, σ_y are used in substitution of the approximations of κ_1, γ_1 and σ obtained computing the first block column of the matrix Ω, the inverse of $\mathcal{S}_p^{-1} M_p$.

When the mesh selection is denoted with only three letters we are working with nonstiff problems and the conditioning parameters are not used in the monitor function. The last letter is E if we use an estimation of the global error (E^E), R if we use a scaled estimation of the residual (E^R) and T if we use a scaled estimation of the local truncation error (E^T).

The mesh selection denoted by HSCE is the hybrid mesh selection introduced in [2] where for high stiff problems, we use the order two method to select the mesh and, when a good approximation is computed, the selected order p is used. The mesh selection is based on the computation of the first block column of the matrix Ω in combination with the error estimation. In particular we choose as monitor function $\phi_c(x_i) = |\|\Omega_{i,0}\|_\infty - \|\Omega_{i-1,0}\|_\infty|$. If the conditioning parameters are stabilized, that is the relative error between the approximation computed with two different meshes has a relative error less than 10^{-2}, also the error estimation is used in the mesh selection. Moreover the border and internal values of the biggest values of the monitor function are multiplied by a factor, in order to add more points in the regions of rapid variation, we use 100 for the values at the two extremes of the interval and 125 for the one in the center of the interval. This monitor function is extremely useful for high stiff problems, but when the problem is only middly stiff the procedure used for changing the order does not work well and for relatively simple problems the code uses the order two, thus requiring much more points. The first new mesh selection added is the one called MSCE for middle stiff problems, that does not use the order two method to selecting an initial appropriate mesh, but uses the same order in all the steps. The multiplication by a factor of the maximum value of the monitor function is useful only for high stiff or middly stiff problems, for general problems the monitor function gives soon information about the error, so the third considered new mesh selection is the one called NSCE, that uses the monitor function as it is. The HSCR, MSCR, NSCR and HSCT, MSCT, NSCT mesh selection strategies use respectively the residual and the truncation error in the mesh selection.

Since the computation of the first block column of the matrix Ω requires the solution of m linear systems, we have introduced a set of mesh selection that uses κ_y, γ_y and σ_y instead of κ_1, γ_1 and σ. Moreover the new monitor function $\phi_s(x)$ is a scaled linear combination of the following monitor functions $\phi_1(x_i) = \|\mathbf{y}'_i\|_1 + \|\mathbf{y}'_{i-1}\|_1$ and $\phi_2(x_i) = |\|\mathbf{y}_i\|_\infty - \|\mathbf{y}_{i-1}\|_\infty|$, where $\mathbf{y}'$ is an approximation of the derivative of the approximated solution at the mesh points of the same order p . It is interesting to observe that, if we have computed a good approximation of the solution, we are going to add more points when its derivative is large, and these are the regions of rapid variation. We have implemented the same variants of the mesh selection based on conditioning, just considering these new monitor functions. The new conditioning parameters based on the computed solution are also used to decide if the parameters are stabilized and we can use the error in the mesh selection. The resulting mesh selection strategies are denoted respectively as HSSE, MSSE and NSSE if we use the global error. A preliminary version of MSSE has been used in [14] for the solution of optimal control problems. The HSSR, MSSR, NSSR, HSST, MSST, NSST mesh selection strategies use respectively the residual and the truncation error in the mesh selection.

Algorithm 1 Mesh selection

$r = 0$
$s^{(r)}$ approximated initial solution,
π_r initial mesh, TMF = string with the chosen monitor function, Z = (R or T or E).
while the maximum estimated error $\max(E^Z)$ is higher than the input tolerances **do**
  compute the values of the new approximate solution in π_r by solving the linear problem (21) using $s^{(r)}$
  compute an approximation of the componentwise mixed relative approximation of the error E^Z
  compute an estimate of the conditioning parameters $\kappa, \kappa_1, \kappa_2, \gamma_1, \sigma$ (optional for TMF= *SS* or TMF= NS*)
  compute $\kappa_y, \gamma_y, \sigma_y$
  compute an estimation of the condition number of M_p.
  if $\max(E^Z)$ is higher than the input tolerances **then**
    compute the monitor function ϕ (ϕ_c or ϕ_s)
    if the error is used in the monitor function **then**
      compute the monitor function ψ
      ($\psi = (E^Z)^{1/o}$ ($o = p + 1$ for R, $p + 1/2$ for T and p for E))
      find a new mesh by equidistributing a linear combination of ϕ and ψ moving the mesh points;
      add and remove points on the new mesh using ψ;
    else
      Equidistribuite ϕ moving the mesh points;
      if the new mesh is similar to the previous one, or the conditioning parameters are stabilized **then**
        add and remove points on the new mesh using ϕ
      end if
    end if
    if the new mesh is not quasi uniform add mesh points such that $0.5 \leq \frac{h_i}{h_{i+1}} \leq 2$
    r=r+1;
  end if
end while

The last three mesh selections, denoted NSE, NST and NSR, are not hybrid. In this case we only use the global error, the truncation error or the residual estimations as monitor function. In summary the available mesh selections are:

- HSCE, HSCR, HSCT, HSSE, HSSR, HSST: hybrid mesh selections based either on conditioning or on an approximation of the conditioning parameters for high stiff problems.
- MSCE, MSCR, MSCT, MSSE, MSSR, MSST: hybrid mesh selections based either on conditioning or on an approximation of the conditioning parameters for middly stiff problems.
- NSCE, NSCR, NSCT, NSSE, NSSR, NSST: hybrid mesh selections based either on conditioning or on an approximation of the conditioning parameters for non stiff problems.
- NSE, NSR, NST: mesh selections for nonstiff problems.

The mixed relative error (15) is approximated using either the global error E^E (last letter E), the scaled residual E^R (last letter R) or the scaled local truncation error E^T (last letter T).

The algorithm for the hybrid mesh selection starts using as monitor function ϕ_c or ϕ_s. The monitor function is firstly used to move the mesh points and after (if needed) to add or remove points. If the estimation of error is considered in the mesh selection,

the monitor function used to move the mesh points is a linear combination of the two, the one used to add and/or remove points is only based on the error. The algorithm for linear problems is briefly described in Algorithm 1. The empirical parameters used to decide how to add and remove points are omitted, they depend on the values of the conditioning parameters.

4 Nonlinear iteration

For linear problems the most important step is the mesh selection, once we have a good information on the error we need just to move mesh points, or to add and remove them, in order to have the error less than the input tolerances. In the nonlinear case we have another big issue: the convergence of the method used for the solution of the nonlinear equations. By the author knowledge usually general purposes codes implement the Newton iteration or some of its variants that enlarge the region of convergence. For difficult problems, like singularly perturbed one, a continuation strategy is considered. Some attempt to make the continuation strategy automatic has been made in the codes colmod and ACDC [26]. In general, the mesh is adapted only if the non linear method is able to compute an approximated solution. The same holds for the quasi-linearization strategy implemented in colsys, colnew [25, 35, 36], the mesh is the same for all the quasi-linearization steps. The code TOM implements the quasi-linearization algorithm described in [32], solving at each step a linear differential boundary value problem, and moving to the next linear problem when some properties are satisfied. This is the most important decision, because solving the linear problem with high accuracy is not required, but if the linear problem is not sufficiently accurate the convergence is not assured in the discrete case. Given $y^{(n)}(x)$ an approximation of the continuous linear problem at step n, the linear continuous problem that we solve at the next step is given by:

$$\begin{cases} (y^{(n+1)}(x))' = f_l(x, y^{(n+1)}(x), y^{(n)}(x)) = J_f(x, y^{(n)}(x))y^{(n+1)}(x) + v_n(x), \\ \quad x \in [a\,,\, b], \\ C_a^n y^{(n+1)}(a) + C_b^n y^{(n+1)}(b) = g_n, \end{cases} \tag{21}$$

where

$$J_f(x, y^{(n)}(x)) := \frac{\partial f(x, y^{(n)}(x))}{\partial y^{(n)}},$$

$$C_a^n := \frac{\partial g(y^{(n)}(a), \mathbf{y}^{(n)}(b))}{\partial y^{(n)}(a)}, \qquad C_b^n := \frac{\partial g(y^{(n)}(a), y^{(n)}(b))}{\partial y^{(n)}(b)},$$

$$\begin{cases} v_n(x) := f(x, y^{(n)}(x)) - J_f(x, y^{(n)}(x))y^{(n)}(x), \\ g_n \quad := C_a^n y^{(n)}(a) + C_b^n y^{(n)}(b) - g(y^{(n)}(a), y^{(n)}(b)). \end{cases}$$

The discrete problem obtained after the discretization with a BVM is the following:

$$M_p\,(\mathbf{y}^{n+1} - \mathbf{y}^n) \;=\; \mathbf{e}_1 \otimes g(y^{(n)}(a), y^{(n)}(b)) \;+\; \mathcal{B}_p\mathbf{f}, \tag{22}$$

where $\mathbf{f} = (f(x_0, \mathbf{y}_0^n)), \cdots, f(x_N, \mathbf{y}_N^n))$. In this way, if we use a fixed mesh for each iteration, the algorithm is equivalent to the Newton method applied to the discretized equations. If we change the mesh, the vector $\mathbf{y}^n$ is computed by evaluating the spline quasi-interpolant of the previous computed solution, that is $\mathbf{y}_i^n = s^{(n)}(x_i)$.

A difference with respect to the implementation described in [32] is given by the criteria to decide when to start the next step. Given $s^{(n+1)}$ the spline quasi-interpolant associated to the current approximation the computed approximation of the residual in the mesh points is approximated as:

$$r^{(n+1)}(x_i) := (s^{(n+1)}(x_i))' - J_f(x_i, s^{(n)}(x_i))\bar{\mathbf{y}}_i^{n+1} - \tag{23}$$

$$(f(x_i, s^{(n)}(x_i)) - J_f(x_i, s^{(n)}(x_i))s^{(n)}(x_i)). \tag{24}$$

Following [37, 38] we check that the relative residual

$$\rho_{n+1} := \frac{\|\mathbf{r}^{(n+1)}\|_{L^2}}{\|(s^{(n+1)})' - f(x, \bar{\mathbf{y}}^{(n+1)})\|_{L^2}} \leq \eta_{n+1}, \tag{25}$$

choosing $\eta_{n+1} < \min(0.01, \|(s^{(n+1)})' - f(x, s^{(n+1)})\|_{L^2})$. This assures the quadratic local convergence of Newton method for linear boundary conditions. To make the procedure more robust we add more checking before to accept the solution:

- the relative error E^Z must be less than 1 using as tolerances for the linear problem:
  $rtol_{lin} := \min(0.05, \max(rtol, \|f_l(x, \mathbf{y}^{(n+1)}(x)) - f(x, \bar{\mathbf{y}}^{(n+1)})\|_{L^2}))$,
  $atol_{lin} := \min(10^{-3}, \max(atol, \|f_l(x, \mathbf{y}^{(n+1)}(x)) - f(x, \bar{\mathbf{y}}^{(n+1)})\|_{L^2}))$;
- the conditioning parameters $\kappa_y, \gamma_y, \sigma_y$ (and if available $\kappa_1, \gamma_1, \sigma$) are stabilized;

The nonlinear iteration is described by Algorithm 2.

Algorithm 2 nonlinear iteration

$n = 0, r = 0$ $s^{(n)}$ approximate initial solution π_r initial mesh
while the error of the nonlinear problem is higher than the input tolerance **do**
  compute the values of the new approximate solution in π_r by solving the linear problem (21) using $s^{(n)}$
  compute an approximation of the error (E^E or E^R or E^T) of the linear problem and the conditioning parameters
  if the linear problem satisfies the stopping criteria **then**
    compute the spline extension $s^{(n+1)}$, $n = n + 1, \pi_0 = \pi_r, r = 0$
  else change the mesh using Algorithm 1, $r = r + 1$
  end if
end while

It is interesting to observe that the mesh selection and the nonlinear iteration are strictly connected. An important role in checking if the numerical solution has been approximated in the correct way is given by the evaluation of the conditioning parameters, in fact if they are stabilized we can suppose that we have reached a good approximation of the continuous solution of the current linear problem.

5 Numerical examples

In this section we present some numerical examples to show that the appropriate mesh selection is important to have a code that works in difficult situations and that the use of conditioning and the mesh selection embedded in the nonlinear iteration can help the convergence of the nonlinear method. We also observe that if the problem is not stiff, the use of the conditioning parameters is not necessary and the method for solving the nonlinear problem usually converges without any issue.

In the following we compare the code TOM with the Matlab codes bvp4c, bvp5c, bvptwp. Even if in the code TOM it is possible to choose the order and the method, we use for all the examples the orders 4 and 6 and the BS method. In summary the codes used in the experiments are:

- TOM_XXX_p: the code TOM selecting as input the BS method of order p and the mesh selection XXX where XXX could be NSE, NST, NSR;
- TOM_XXXX_p: the code TOM selecting as input the BS method of order p and the hybrid mesh selection XXXX where XXXX could be one of the mesh selections defined in section 3 (HSCE, HSCT, HSCR, ..., NSSE, NSST, NSSR);
- bvp4c: three-stage Lobatto IIIa formula of order 4 [39];
- bvp5c: four-stage Lobatto IIIa formula of order 5 [40];
- twpbvp_l (twpbvp_m): the code bvptwp selecting as input the Lobatto IIIa formulas (MIRK methods) with standard mesh selection. The code uses variable orders 4, 6 and 8 [5];
- twpbvpc_l (twpbvpc_m): the code bvptwp selecting as input the Lobatto IIIa formulas (MIRK methods) with hybrid mesh selection based on conditioning. The code uses variable orders 4, 6 and 8 [5].

For all the compared codes we choose as maximum number of mesh points N=20000. All the experiments have been performed using an Intel Core I9 10 core 3.6 GHz with Matlab R2021b for MacOS. The work precision diagrams are computed choosing the tolerances $atol = rtol$ ranging from 10^{-3} to 10^{-8} and by estimating the mixed error as

$$\max_{0\le i\le N} \left\| \frac{|\mathbf{y}_i^p - \mathbf{y}_i^*|}{\max(1, \min(|\mathbf{y}_i^p|, |\mathbf{y}_i^*|))} \right\|_\infty$$

by computing a more precise solution $\mathbf{y}^*$ with the code twpbvp_m with a doubled number of fixed points and a relative tolerance 10 times smaller.

We choose for comparison some linear and nonlinear examples available in the website testset for BVP solvers [27], that contains a list of singularly perturbed test problems in Fortran and in Matlab. The Matlab version has been used to check the behaviour of the code bvptwp in [5].

The first example is the problem bvpT1, a linear second order boundary value problem, written as a system of first order equations. This problem has a boundary layer in a, but if we choose $\lambda = 10^{-2}$ the solution does not have any rapid variations. In this case it is not useful the use of the conditioning in the mesh selection and the code is more efficient if we use the mesh selection for nonstiff problems. Figure 1

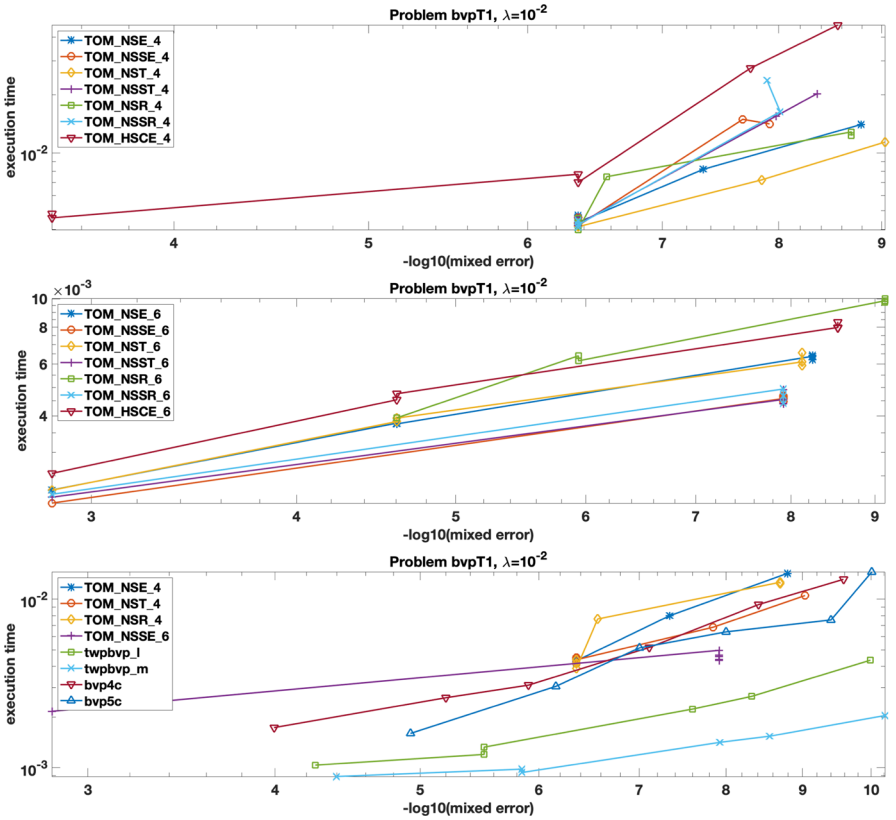

Fig. 1 Work precision diagrams for the linear problem bvpT1, comparison using different mesh selections order 4(top) and order 6 (middle), comparison with the other matlab solvers (bottom)

shows that the NSE, NSR, NST, NSSE, NSST and NSSR mesh selections give similar results, so they can be used indifferently, HSCE gives the highest computational time. Moreover we observe that the behaviour of the compared methods is mainly dictated by the order of the numerical method used and not by the mesh selection. In fact the methods with a smaller execution time are the ones with variable orders 4, 6 and 8 (twpbvp_m and twpbvp_l).

The second test problem is the problem bvpT6, a linear problem with a turning point. We choose $\lambda = 10^{-2}$ and we observe that the behaviour of the codes is similar to the one of the first example, the mesh selection HSCE being the most expensive (see Fig. 2). This is confirmed by the values of the stiffness parameter reported in Table 1, that shows that the problem is nonstiff.

For nonlinear problems the situation is more complicated, because the convergence of the methods used for its solution is not assured. Figure 3 shows the results for the nonlinear problem bvpT24. For $\lambda = 10^{-2}$ all the codes behave in a similar way. For $\lambda = 10^{-3}$ (Fig. 3, bottom) only the codes twpbvp_l, twpbvpc_l and TOM are able to give a solution. The comparison of some of the mesh selections shows that the

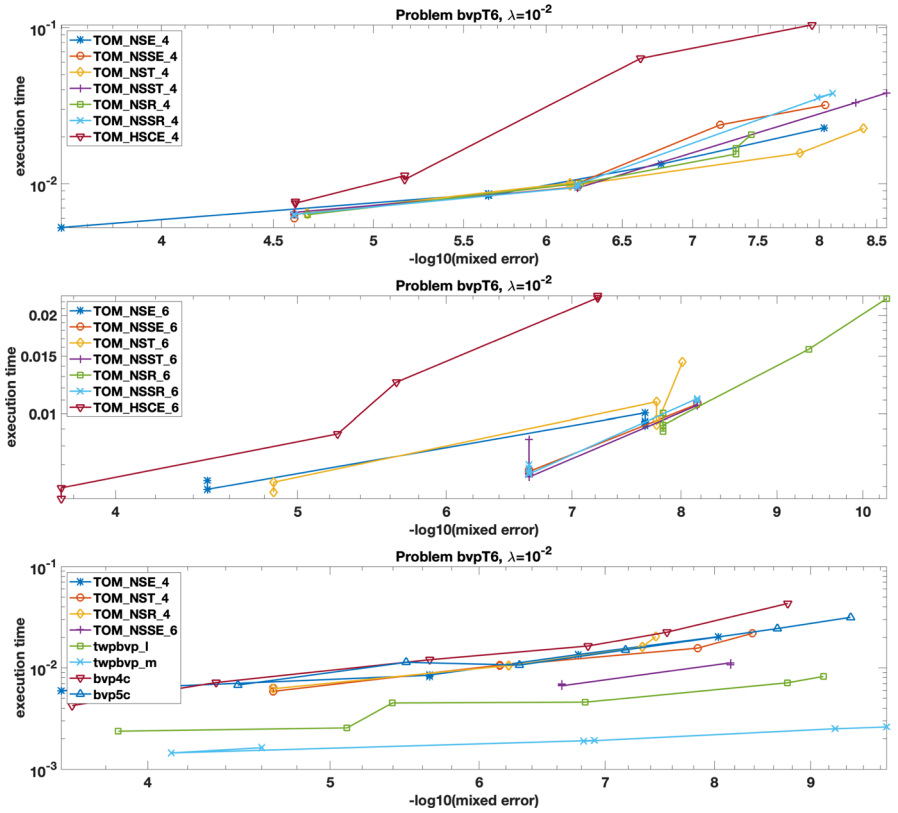

Fig. 2 Work precision diagrams for the linear problem bvpT6. Comparison using different mesh selections order 4(top) and order 6 (middle), comparison with the other matlab solvers (bottom)

Table 1 Values of the computed conditioning parameters using the code TOM (tol=10^{-6}, ord=6, MSCE)

Example	λ	κ	κ_1	κ_y	γ_1	γ_y	σ	σ_y
bvpT1	10^{-2}	$1.11 \cdot 10^{1}$	$1.00 \cdot 10^{1}$	$1.10 \cdot 10^{1}$	$2.21 \cdot 10^{0}$	$1.06 \cdot 10^{0}$	$4.51 \cdot 10^{0}$	$9.45 \cdot 10^{0}$
bvpT6	10^{-2}	$1.62 \cdot 10^{1}$	$7.96 \cdot 10^{0}$	$8.97 \cdot 10^{0}$	$0.96 \cdot 10^{0}$	$2.91 \cdot 10^{0}$	$7.48 \cdot 10^{0}$	$2.75 \cdot 10^{0}$
bvpT24	10^{-2}	$2.33 \cdot 10^{3}$	$1.76 \cdot 10^{3}$	$1.44 \cdot 10^{1}$	$1.01 \cdot 10^{2}$	$1.35 \cdot 10^{0}$	$1.65 \cdot 10^{1}$	$9.84 \cdot 10^{0}$
bvpT24	10^{-3}	$2.33 \cdot 10^{5}$	$1.76 \cdot 10^{5}$	$1.38 \cdot 10^{2}$	$1.00 \cdot 10^{3}$	$1.47 \cdot 10^{0}$	$1.77 \cdot 10^{2}$	$9.37 \cdot 10^{1}$
bvpT27	10^{-8}	$1.00 \cdot 10^{8}$	$1.00 \cdot 10^{8}$	$5.00 \cdot 10^{7}$	$2.11 \cdot 10^{0}$	$1.35 \cdot 10^{0}$	$4.75 \cdot 10^{7}$	$3.70 \cdot 10^{7}$
bvpT33	10^{-3}	$1.32 \cdot 10^{4}$	$1.26 \cdot 10^{4}$	$5.28 \cdot 10^{2}$	$7.76 \cdot 10^{2}$	$3.55 \cdot 10^{1}$	$1.49 \cdot 10^{1}$	$1.41 \cdot 10^{1}$

mesh selections based on the conditioning (HSCE) are much more expensive than the one for nonstiff problems. This means that the troubles are mainly due to the method used for the solution of the nonlinear equations. The conditioning is instead important for the nonlinear problem bvpT27 with $\lambda = 10^{-8}$ as shown in Fig. 4: in this case the code TOM does not work if we use a mesh selection that is not based on the conditioning parameters, the same applies for the codes twpbvp_l and twpbvp_m, that

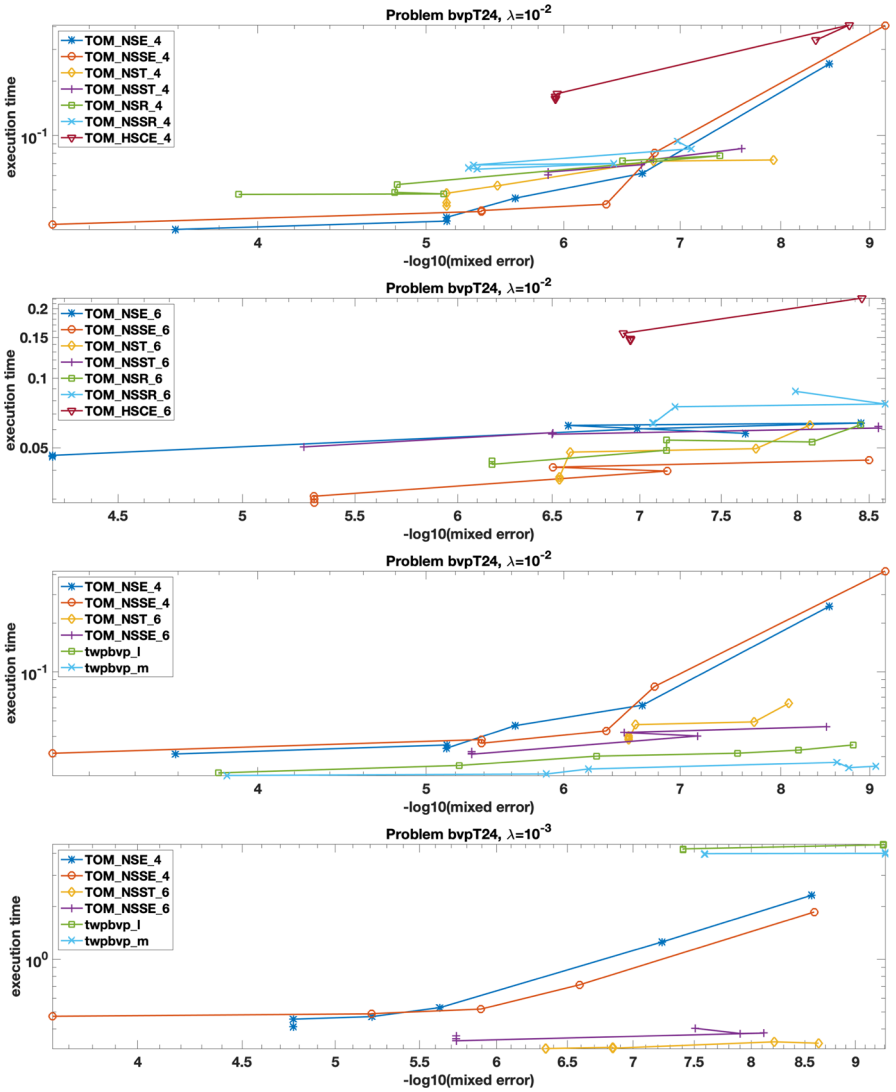

Fig. 3 Work precision diagrams for the nonlinear problem bvpT24. Comparison using different mesh selections for $\lambda = 10^{-2}$ with order 4(top) and order 6 (middle top), comparison with the other matlab solvers ($\lambda = 10^{-2}$ middle bottom, $\lambda = 10^{-3}$ bottom)

are not able to end the computation and stopped with an error. Figure 4 shows that the mesh selections for high stiff problems are, in general, the most efficient. For smaller values of the parameter λ only the code TOM with the mesh selection for high stiff problems is able to end the computation without errors.

We show as last example the nonlinear problem bvpT33 in Fig. 5. For $\lambda = 10^{-3}$ the problem is nonstiff as confirmed by the stiffness constant in Table 1. The codes twpbvp_m, twpbvp_l require a shorter computational time, due to the higher order

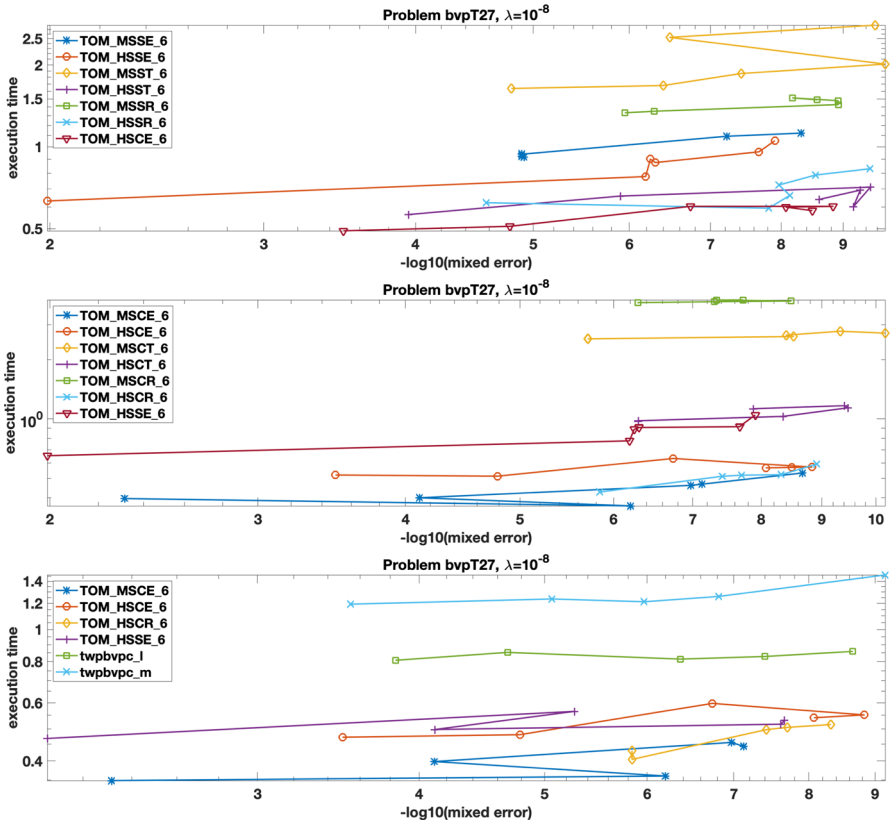

Fig. 4 Work precision diagrams for the nonlinear problem bvpT27. Comparison using different mesh selections (top,middle), comparison with the other matlab solvers (bottom)

and, as expected, twpbvp_m is the most efficient. The orginal mesh selection HSCE, as showed in Fig. 5, bottom and middle pictures, shows the highest computational time indipendently on the order used, for this problem the lower computational time is reached using the new mesh selection strategies.

Information about the approximation of the conditioning constant of the chosen examples is reported in Tables 1–2. It is possible to see that the values of κ and κ_1 approximated using the three different numerical methods have at least two significant corrected digits, confirming the theoretical results of Theorem 1. The approximation of the other conditioning parameters is more sensible to the change of the numerical method. The stiffness constant has the same magnitude order in all the used methods and it is very well approximated by the parameter σ_y.

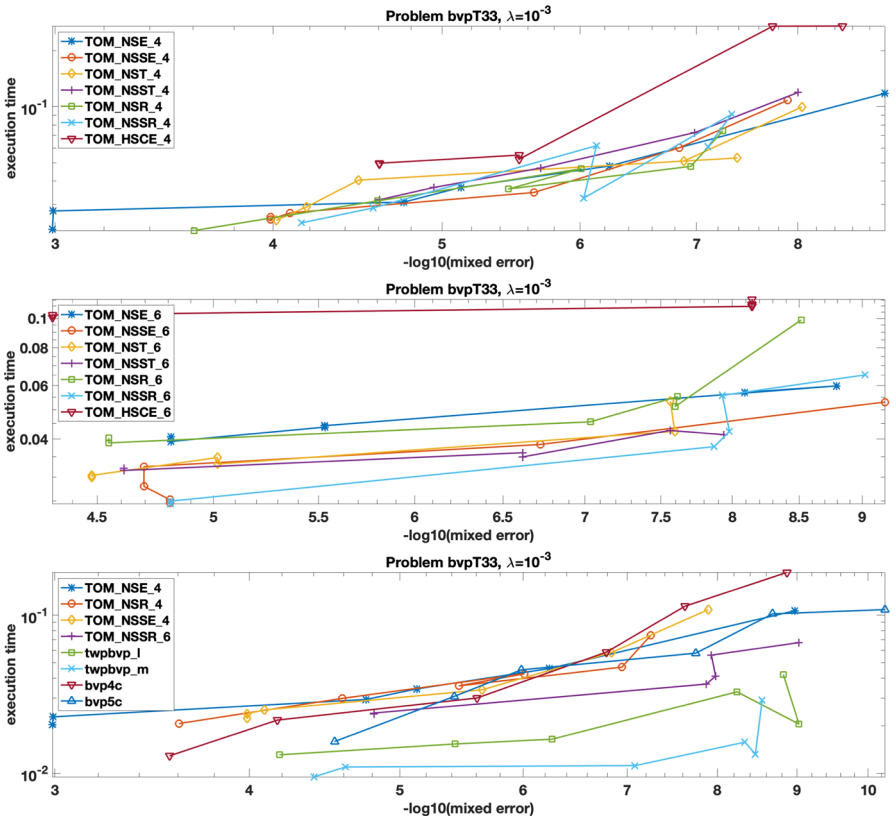

Fig. 5 Work precision diagrams for the non lienar problem bvpT33. Comparison using different mesh selections order 4(top) and order 6 (middle), comparison with the other matlab solvers (bottom)

Table 2 Values of the computed conditioning parameters using the codes bvptwp with solvers twpbvpc_l and twpbvpc_m (tol=10^{-6})

		twpbvpc_l/twpbvpc_m		twpbvpc_l		twpbvpc_m	
Example	λ	κ	κ_1	γ_1	σ	γ_1	σ
bvpT1	10^{-2}	$1.11 \cdot 10^{1}$	$1.00 \cdot 10^{1}$	$2.84 \cdot 10^{0}$	$7.86 \cdot 10^{0}$	$2.84 \cdot 10^{0}$	$7.86 \cdot 10^{0}$
bvpT6	10^{-2}	$1.62 \cdot 10^{1}$	$7.98 \cdot 10^{0}$	$1.86 \cdot 10^{0}$	$4.22 \cdot 10^{0}$	$1.93 \cdot 10^{0}$	$4.07 \cdot 10^{0}$
bvpT24	10^{-2}	$2.31 \cdot 10^{3}$	$1.75 \cdot 10^{3}$	$1.12 \cdot 10^{2}$	$1.58 \cdot 10^{1}$	$1.10 \cdot 10^{2}$	$1.57 \cdot 10^{1}$
bvpT24	10^{-3}	$2.33 \cdot 10^{5}$	$1.76 \cdot 10^{5}$	$1.02 \cdot 10^{3}$	$1.73 \cdot 10^{2}$	$1.03 \cdot 10^{3}$	$1.71 \cdot 10^{2}$
bvpT27	10^{-8}	$1.00 \cdot 10^{8}$	$1.00 \cdot 10^{8}$	$2.52 \cdot 10^{0}$	$9.11 \cdot 10^{7}$	$2.53 \cdot 10^{1}$	$9.10 \cdot 10^{7}$
bvpT33	10^{-3}	$1.32 \cdot 10^{4}$	$1.26 \cdot 10^{4}$	$9.50 \cdot 10^{2}$	$3.25 \cdot 10^{1}$	$8.77 \cdot 10^{2}$	$3.54 \cdot 10^{1}$

6 Conclusion

In this paper we have presented the new mesh selection strategies implemented in the code TOM, together with a proper approximation of the conditioning constants of the discretized problem when the underlying method is not one step. The results show that for linear problems the hybrid mesh selections are efficient only for the stiff case, this justifies the introduction of standard mesh selection for the solution of nonstiff problems. Moreover, for nonlinear problems, the conditioning parameters, or their approximations, allow to implement a method for their solution that is efficient for some problems for which the standard Newton type iteration methods do not converge using a fixed mesh. The proposed quasi-linearization strategy for the solution of non linear problems needs some more theoretical studies in order to have a robust technique for choosing the acceptance of the solution of an intermediate linear system. Moreover it would be interesting to see how the generation of a final smooth grid using the algorithm presented in [13] would impact on the performance of the code.

Acknowledgements The author wishes to thank Antonella Falini and Felice Iavernaro for useful comments and suggestions and the anonymous referees for their careful reading and useful remarks. The research has been founded by the INdAM-GNCS Research Project "Numerical algorithms in optimization, ODEs, and applications" CUP_E55F22000270001. The author is member of the INdAM Research group GNCS.

Funding Open access funding provided by Università degli Studi di Bari Aldo Moro within the CRUI-CARE Agreement. INdAM-GNCS Project, CUP_E55F22000270001.

Declarations

Conflict of interest The author declares no conflict of interest.

References

1. Boisvert, J.J., Muir, P.H., Spiteri, R.J.: A Runge-Kutta BVODE solver with global error and defect control. ACM Trans. Math. Softw. **39**(2), 1–2 (2013)
2. Mazzia, F., Trigiante, D.: A hybrid mesh selection strategy based on conditioning for boundary value ODE problems. Numer. Algorithms **36**(2), 169–187 (2004)
3. Cash, J.R., Mazzia, F.: A new mesh selection algorithm, based on conditioning, for two-point boundary value codes. J. Comput. Appl. Math. **184**(2), 362–381 (2005)
4. Cash, J.R., Mazzia, F.: Hybrid mesh selection algorithms based on conditioning for two-point boundary value problems. JNAIAM J. Numer. Anal. Ind. Appl. Math. **1**(1), 81–90 (2006)
5. Cash, J.R., Hollevoet, D., Mazzia, F., Nagy, A.: Algorithm 927: the matlab code bvptwp.m for the numerical solution of two point boundary value problems. ACM Trans. Math. Softw. (TOMS) **39**(2), 1–12 (2013)

6. Shishkin, G.I.: A difference scheme for a singularly perturbed equation of parabolic type with discontinuous boundary conditions. USSR Comput. Math. Math. Phys. **28**(6), 32–41 (1988). https://doi.org/10.1016/0041-5553(88)90039-0
7. Bakhvalov, N.S.: The optimization of methods of solving boundary value problems with a boundary layer. USSR Comput. Math. Math. Phys. **9**(4), 139–166 (1969). https://doi.org/10.1016/0041-5553(69)90038-X
8. Podila, P.C., Gupta, T., Vigo-Aguiar, J.: A numerical scheme for a weakly coupled system of singularly perturbed delay differential equations on an adaptive mesh. Comput. Math. Methods **3**(3), 1104 (2021). https://doi.org/10.1002/cmm4.1104
9. Vulanović, R., Nhan, T.A.: Robust hybrid schemes of higher order for singularly perturbed convection-diffusion problems. Appl. Math. Comput. **386**, 125495 (2020)
10. Pulverer, G., Söderlind, G., Weinmüller, E.: Automatic grid control in adaptive bvp solvers. Numerical Algorithms **56**(1), 61–92 (2011). https://doi.org/10.1007/s11075-010-9374-0
11. Kitzhofer, G., Koch, O., Pulverer, G., Simon, C., Weinmüller, E.B.: The new matlab code bvpsuite for the solution of singular implicit bvps. J. Numer. Anal. Industrial Appl. Math. **5**(1–2), 113–134 (2010)
12. Auzinger, W., Kneisl, G., Koch, O., Weinmüller, E.: A collocation code for singular boundary value problems in ordinary differential equations. Numer. Algorithms **33**(1–4), 27–39 (2003). https://doi.org/10.1023/A:1025531130904
13. Söderlind, G., Yadaw, A.S.: The impact of smooth W-grids in the numerical solution of singular perturbation two-point boundary value problems. Appl. Math. Comput. **218**(10), 6045–6055 (2012). https://doi.org/10.1016/j.amc.2011.11.086
14. Mazzia, F., Settanni, G.: Bvps codes for solving optimal control problems. Math. **9**(20), 2618 (2021). https://doi.org/10.3390/math9202618
15. De Marinis, A., Iavernaro, F., Mazzia, F.: A minimum-time obstacle-avoidance path planning algorithm for unmanned aerial vehicles. Numer. Algorithms **89**(4), 1639–1661 (2022). https://doi.org/10.1007/s11075-021-01167-w
16. Li, Y., Wang, Z.: An Adaptive Cross Approximation-based Method for Robust Nonlinear Feedback Control Problems, vol. 2018-June, pp. 3460–3465 (2018). https://doi.org/10.23919/ACC.2018.8431842
17. Putkaradze, V., Rogers, S.: Constraint control of nonholonomic mechanical systems. J. Nonlinear Sci. **28**(1), 193–234 (2018). https://doi.org/10.1007/s00332-017-9406-1
18. Putkaradze, V., Rogers, S.: Numerical simulations of a rolling ball robot actuated by internal point masses. Numer. Algebra Control Optim. **11**(2), 143–207 (2021). https://doi.org/10.3934/naco.2020021
19. Giordano, D., Amodio, P., Iavernaro, F., Labianca, A., Lazzo, M., Mazzia, F., Pisani, L.: Fluid statics of a self-gravitating perfect-gas isothermal sphere. European J. Mechanics, B/Fluids **78**, 62–87 (2019). https://doi.org/10.1016/j.euromechflu.2019.05.013
20. Gazzola, F., Pavani, R.: The impact of nonlinear restoring forces acting on hinged elastic beams. Bulletin Belgian Math. Soc. - Simon Stevin **22**(4), 559–578 (2015). https://doi.org/10.36045/bbms/1447856059
21. Uecker, H.: Continuation and bifurcation in nonlinear pdes - algorithms, applications, and experiments. Jahresber. Deutsch. Math.-Verein. **124**(1), 43–80 (2022). https://doi.org/10.1365/s13291-021-00241-5
22. Uecker, H.: Hopf bifurcation and time periodic orbits with pde2path - algorithms and applications. Commun. Comput. Phys. **25**(3), 812–852 (2019). https://doi.org/10.4208/CICP.OA-2017-0181
23. Soetaert, K., Cash, J., Mazzia, F.: Solving Differential Equations in R. Springer, Berlin, Heidelberg (2012)
24. Mazzia, F., Cash, J.R., Soetaert, K.: Solving boundary value problems in the open source software R: Package bvpsolve. Opuscula Math. **34**(2), 387–403 (2014)
25. Ascher, U.M., Mattheij, R.M., Russell, R.D.: Numerical Solution of Boundary Value Problems for Ordinary Differential Equations. SIAM, Philadelphia (1995)
26. Cash, J.R., Moore, G., Wright, R.: An automatic continuation strategy for the solution of singularly perturbed nonlinear boundary value problems. ACM Trans. Math. Software **27**(2), 245–266 (2001)
27. Mazzia, F., Cash, J.R.: A fortran test set for boundary value problem solvers. AIP Conference Proceedings **1648**(1), 020009 (2015). https://doi.org/10.1063/1.4912313. https://archimede.uniba.it/~bvpsolvers/testsetbvpsolvers/
28. Brugnano, L., Trigiante, D.: A new mesh selection strategy for ODEs. Appl. Numer. Math. **24**(1), 1–21 (1997)

29. Cash, J.R., Mazzia, F.: Conditioning and hybrid mesh selection algorithms for two-point boundary value problems. Scalable Comput. Practice Experience **10**(4), 347–361 (2009)
30. Capper, S., Cash, J., Mazzia, F.: On the development of effective algorithms for the numerical solution of singularly perturbed two-point boundary value problems. Int. J. Comput. Sci. Math. **1**(1), 42–57 (2007)
31. Brugnano, L., Trigiante, D.: Solving Differential Problems by Multistep Initial and Boundary Value Methods. Gordon & Breach, Amsterdam (1998)
32. Mazzia, F., Sestini, A., Trigiante, D.: The continuous extension of the B-spline linear multistep methods for BVPs on non-uniform meshes. Appl. Numer. Math. **59**(3–4), 723–738 (2009). https://doi.org/10.1016/j.apnum.2008.03.036
33. Mazzia, F., Sestini, A.: The bs class of hermite spline quasi-interpolants on nonuniform knot distributions. BIT Numer. Math. **49**(3), 611–628 (2009). https://doi.org/10.1007/s10543-009-0229-9
34. Bertolazzi, E., Falini, A., Mazzia, F.: The Object Oriented C++ library QIBSH++ for Hermite spline Quasi Interpolation. arXiv:2208.03260 (2022). https://doi.org/10.48550/arXiv.2208.03260
35. Ascher, U., Christiansen, J., Russell, R.D.: Collocation software for boundary-value odes. ACM Trans. Math. Softw. **7**, 209–222 (1981)
36. Bader, G., Ascher, U.: A new basis implementation for a mixed order boundary value ode solver. Siam J. Scient. Stat. Comput. **8**(4), 483–500 (1987)
37. Dean, E.J.: An inexact newton method for nonlinear two-point boundary-value problems. J. Optim. Theory Appl. **75**(3), 471–486 (1992)
38. Dembo R.S., Eisenstat S.C., Steihaug T. Inexact Newton methods. SIAM J. Numer. Anal. **19**, 400–408 (1982)
39. Shampine, L.F., Kierzenka, J.: A bvp solver based on residual control and the matlab pse. ACM Trans. Math. Softw. **27**(3), 299–317 (2002)
40. Shampine, L.F., Kierzenka., J.: A bvp solver that controls residual and error. J. Numer. Anal. Ind. Appl. Math.. 3(1-2), 27–41 (2008)