MATHEMATICS OF COMPUTATION
VOLUME 42, NUMBER 165
JANUARY 1984, PAGES 25-40

On Optimal Shooting Intervals

By R. M. M. Mattheij and G. W. M. Staarink

Abstract. We develop an adaptive multiple shooting strategy, which is nearly optimal with respect to cpu time. Since the costs of integration are the most important components in this, we investigate in some detail how the gridpoints are chosen by an adaptive integration routine. We use this information to find out where the shooting points have to be selected. We also show that our final strategy is stable in the sense that rounding errors can be kept below a given tolerance. Finally we pay attention to the question how the need for memory can be minimized.

1. Introduction. Consider the ODE

$$\frac{dx(t)}{dt} = L(t)x(t) + r(t), \qquad \alpha \leqslant t \leqslant \beta, \tag{1.1}$$

where $L(\cdot)$ is an $n \times n$ matrix function and $r(\cdot)$ an n vector function. Assume that the following *boundary condition* (BC) is given for x

$$M_\alpha x(\alpha) + M_\beta x(\beta) = b, \tag{1.2}$$

where M_α and M_β are $n \times n$ matrices. Most often both forward and backward integration is unstable due to the presence of rapidly increasing and rapidly decreasing solutions. This is the reason why a *multiple shooting* (M.S.) *technique* is preferred. In such a method one selects a set of shooting points t_0 ($= \alpha$), $t_1, \ldots, t_k$ ($= \beta$) and restarts the integration on each interval (t_i, t_{i+1}) for $i = 0, \ldots, k-1$. In this way one computes for each i an approximate particular solution (if the problem is inhomogeneous), $\bar{f}_i(t)$ say, and an approximate fundamental solution, $\bar{\Phi}_i(t)$ say, on some grid $\subset [t_i, t_{i+1}]$. Then an approximant, $\bar{x}(t)$ say, of the desired solution can be written as

$$\bar{x}(t) = \bar{\Phi}_i(t)v_i + \bar{f}_i(t), \tag{1.3}$$

where v_i is some fixed vector.

By matching the relations (1.3) at the shooting points, we obtain a relation for the v_i,

$$A_i v_i - B_i v_{i+1} = F_i, \qquad i = 0, \ldots, k-2, \tag{1.4}$$

where

$$\begin{aligned} &\text{(a)} \quad A_i = \bar{\Phi}_i(t_{i+1}), \\ &\text{(b)} \quad B_i = \bar{\Phi}_{i+1}(t_{i+1}), \\ &\text{(c)} \quad F_i = \bar{f}_{i+1}(t_{i+1}) - \bar{f}_i(t_{i+1}), \qquad 0 \leqslant i \leqslant k-2. \end{aligned} \tag{1.5}$$

Received August 19, 1981; revised February 23, 1983.
1980 *Mathematics Subject Classification*. Primary 65L10.
Key words and phrases. Multiple shooting, adaptive codes.

The relations (1.4) together with a transformed version of the BC (1.2) give the following multiple shooting equations that determine the v_i (and hence $\bar{x}$):

(1.6) $PV = F$, where

$$P = \begin{bmatrix} A_0 & -B_0 & & & & \\ & A_1 & -B_1 & & \varnothing & \\ & & \ddots & \ddots & & \\ & \varnothing & & \ddots & \ddots & \\ & & & A_{k-2} & -B_{k-2} & \\ M_\alpha \bar{\Phi}_0(t_0) & & & & M_\beta \bar{\Phi}_{k-1}(t_N) & \end{bmatrix}, \tag{1.7}$$

$V = (v_0^T, \ldots, v_{k-1}^T)^T$ and $F = (F_0^T, \ldots, F_{k-1}^T)^T$; for $0 \leqslant i \leqslant k-2$ the F_i are defined by (1.5) and moreover $F_{k-1} := b - M_\alpha \bar{f}_0(t_0) - M_\beta \bar{f}_{N-1}(t_N)$. This description of multiple shooting gets the general framework. There exist many variants which may be considered as modifications regarding one or more of the following aspects: the choice of the shooting points, the number of basis solutions of the homogeneous system (i.e. the number of columns in the Φ_i) and, finally, the way the desired solution is computed from the matching recursion and the BC (cf. (1.7)). The last question is dealt with either by using a special block LU solver or by using special recursion techniques for solving (unstable) recursions, cf. [2], [8], [12], [16], [18]. The problem of how many basis solutions should be integrated arises in so-called *(partially) separated boundary conditions* (cf. [4], [13], [16]): By limiting the number of columns in the Φ_i to l say, the efficiency of the method is expected to increase. If $l < n$, one can reorder the matrix P such that one effectively deals with $l \times l$ blocks only (cf. [13]). For our present discussion we are mainly interested in the question of how the shooting points t_i should be chosen. The simplest way for doing this is to give them in advance (cf. [2], [3], [18]). However, as adaptivity is one of the most appealing aspects of M.S., one rather would like to have a device that decides for itself when the integration should be restarted (cf. [4], [13], [16]). There is still much discussion about the optimal choice (cf. [1, p. 159ff.]). We can think of three optimality criteria: The first one is stability. Originally this was one of the ideas behind M.S. to improve single shooting by making the interval of integration sufficiently small in order to limit growth of (rounding) errors. The second criterion is the amount of work. In using an automatic integrator and a (rough) stability check, we may be far away from optimally distributed grids, as described e.g. in [7], [14], [15], and therefore have an inefficient algorithm. Another criterion is the memory space that is required. Apart from situations where some shooting points may be given in advance (cf. [2], [3]) or where the solution has to be computed on a sufficiently dense grid, it quite often happens that one is merely interested in having approximations at just some points; if more information is needed, one may restart the algorithm on some smaller shooting interval. In the last situation it makes sense to ask for the minimal number of shooting points $k_{\min}$ say; for the smaller $k_{\min}$, the less matrices A_i and B_i have to be stored. This paper is intended to contribute to the discussion about optimality regarding these three criteria, viz. optimal stability, minimal complexity (or rather computing time) and minimal storage. For this we will study in some detail a model problem from which we can draw a number of conclusions. The results we will derive are likely to hold for more general BVP of mildly stiff ODE. It should be clear, however, that we cannot be too ambitious and may hope for a suitable divide and conquer strategy at most. For example, single shooting is an optimal strategy from a storage point of view but is most often unstable.

An outline of this paper is as follows: In Section 2 we first consider optimal shooting intervals in relation to the stability (or rather propagated rounding errors) and derive a practical criterion for an automatic check. In Section 3 we introduce a model problem and indicate why the adaptive integrator should be used only once. In Section 4 we give an estimation theory for the number of gridpoints required by an automatic integrator with local step size control. This is applied in Section 5, where a bound is given for the total operational cost of a multiple shooting algorithm; here we also derive an almost optimal strategy regarding the complexity. In Section 6 we show how we may condense the multiple shooting equations in order to minimize the storage. Several examples sustain the theory.

2. Shooting Intervals and Error Amplification. The most popular variants of multiple shooting for linear problems use some reorthogonalization technique at the shooting points; cf. [4], [8], [12], [13], [16]. In this the matrices $\overline{\Phi}_i(t_{i+1})$ are factorized into an orthogonal and an upper triangular matrix. The orthogonal matrix appearing in this factorization is then used as an initial value for $\overline{\Phi}_{i+1}(t_{i+1})$. The information contained in the upper triangular matrix may be used for selecting the shooting points (cf. [4], [13], [16]). In our opinion the most appropriate criterion to select a shooting point in order to limit the error growth is the one that is also mentioned in [13], viz. a check on the incremental matrices $A_iB_{i-1}^{-1}$; in fact this criterion may also be used if no orthogonalizations are used. Although a detailed description of stability aspects is outside the scope of this paper, we certainly have to justify this, at least qualitatively. (For analysis cf. [2], [10], [13], [18].)

For a proper understanding of the global error, it is useful to distinguish between rounding errors and discretization errors. For the latter type of errors one can show that the global effect of a local discretization error, with bound δ say, is estimated by $\sim \kappa N\delta$; here κ is some constant, that is $O(1)$ if the BVP is well conditioned, and N is the number of gridpoints (cf. [9], [10]). Of course this makes sense only in exact arithmetic. The rounding errors, however, deal only with the *discrete* problem and the way the computations are actually carried out. In particular for multiple shooting we should reckon with rounding errors made in the two main steps. In the first step we in fact solve k unstable initial value problems on the intervals (t_i, t_{i+1}). If ξ denotes *the relative machine accuracy*, then we can expect propagated rounding errors at t_{i+1} with an error bound $\leqslant \|A_iB_{i-1}^{-1}\|\xi$. The second step is the solution of the multiple shooting equations. Assuming that P is well conditioned (cf. [9]), we may deduce that errors made at this step are only moderate. (This argument is still meaningful if the matrix P is not formed explicitly like in [16].) Hence we find that the global rounding error mainly arises from errors in integrating the k unstable initial value problems. From the general result given in [10], we therefore conclude that a global error bound is given by

$$\sim \kappa k \max_i \|A_iB_{i-1}^{-1}\|\xi, \tag{2.1}$$

where κ is a conditioning constant as above. (N.B. the factor k does not appear if we have an exponential dichotomy, cf. [9, Section 5].) Having accepted (2.1) as a bound for the global rounding errors, a practical check on allowed error amplification is then given by

$$\|A_i B_{i-1}^{-1}\| \leq \frac{\varepsilon}{\kappa k \xi}, \tag{2.2}$$

where ε is the *required accuracy* for the solution of the problem. Notice that (2.2) is certainly an improvement of criteria, as e.g. mentioned in [4], [13], [16], since the latter do not reckon with the required global accuracy at all.

Example 2.3. Consider the ODE

$$\frac{dx}{dt} = \begin{bmatrix} 1 - 19\cos 2t & 0 & 1 + 19\sin 2t \\ 0 & 19 & 0 \\ -1 + 19\sin 2t & 0 & 1 + 19\cos 2t \end{bmatrix} x + r(t),$$

where $r(t)$ is chosen such that

$$x \equiv \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

is a solution. The homogeneous ODE has solutions growing like $\sim e^{20t}, e^{19t}, e^{-18t}$.

Let $M_\alpha = M_\beta = I$ and $[\alpha, \beta] = [0, \pi]$. It is not difficult to check that $\kappa \approx 1$ (cf. [9]). Since x should be integrated exactly, we are mainly confronted with errors due to rounding. To solve this problem we used an M.S. code as described in [12], with various choices for the increments $\|A_i B_{i-1}^{-1}\|$, on an IBM 370/165. Table 2.1 gives the errors in $\|\cdot\|_\infty$ for single precision ($\xi \approx 4.7\ 10^{-7}$) and Table 2.2 the errors in double precision ($\xi \approx 1.1\ 10^{-16}$). The influence of the incremental value is as predicted indeed. (N.B. the factor k does not appear in this case, cf. [9, Example 6.3].) Here $\mathcal{G}$ denotes the maximal increment, *error* the actually found maximal error, and *nos* the number of shooting points.

TABLE 2.1 ($\varepsilon = 10^{-6}$)

$\mathcal{G}$	10	100	1000
error	$9.2\ 10^{-6}$	$4.6\ 10^{-5}$	$7.5\ 10^{-4}$
nos	28	14	10

TABLE 2.2 ($\varepsilon = 10^{-8}$)

$\mathcal{G}$	10^3	10^4	10^5	10^6
error	$1.1\ 10^{-13}$	$1.4\ 10^{-12}$	$3.3\ 10^{-11}$	$2.6\ 10^{-10}$
nos	10	7	6	5

3. The Model Problem. The question of optimal complexity that will be discussed in Sections 4, 5 is closely related to the optimal integration strategy. In this paper we assume that we are using an adaptive integrator. Such an integrator will always produce overhead, mainly due to rejecting step size choices. Since we have to compute several solutions on each interval, we basically have several times this overhead. However, in general, a solution will always contain a dominant mode, and it is this mode that is actually responsible for the step size choice. Therefore it seems sensible to use the adaptive feature of the integrator only once on each interval for a certain solution and to use the integrator as such for computing the remaining solutions on the same grid. The savings thus obtained are considerable as one may check e.g. from Table 3.1, where we have given in column two the cpu time (in millisecs) that was needed to compute the particular solutions $\bar{f}_i$ in Example 2.3 by the adaptive method RKF45 (cf. Example 4.7). The number of gridpoints found by this integrator (N) is given in column three. Finally in column four we have given the cpu time for the computation of the fundamental solutions $\bar{\Phi}_i$, only using the fourth order Runge-Kutta method on this grid.

TABLE 3.1

ε	total cpu part. sol.	N	total cpu fund. sol.
10^{-3}	126	121	129
10^{-4}	150	175	190
10^{-5}	224	269	283
10^{-6}	337	418	439

In order to get an idea of how large the interval must be, or how many points per interval should be chosen to have an optimal computing time, it makes sense to consider this problem for a simplified model first.

The model is based on the following ideas: First, it is assumed that the adaptivity feature is applied to the inhomogeneous problem only (see however Remark 3.4). In agreement to what has been said above, the fundamental solution therefore is just integrated on the thus computed grid. Second, we assume that there exists some smooth particular solution (even for boundary layer problems this is quite often the case). We can then always scale the solutions of (1.1) such that this particular solution is of order 1 (this is only for reasons of simplicity), and we assume this has been done. In the third place, by absence, in general, of other information, we assume that the particular solution on each interval (t_i, t_{i+1}) has been given the initial value 0. As a consequence, this particular solution is expected to have an $O(1)$ component of the fast mode at $t = t_i$. Finally, we assume that $L(t)$ does not vary much on $[\alpha, \beta]$. For this kind of problems, the following scalar ODE gives a qualitatively satisfactory description regarding the growth behavior of the solutions:

$$\frac{dx(t)}{dt} = \lambda x - \lambda g(t), \qquad \lambda > 0, \tag{3.1}$$

where $g(\cdot)$ is a smooth scalar function with $\forall_t g(t) \approx 1$. On each of the intervals we then compute an approximant of the (exact) particular solution $q_i(\cdot)$ say, defined by

$$q_i(t) = \lambda \int_{t_i}^{t} g(\tau) e^{\lambda(t-\tau)}\, d\tau, \qquad t_i \leqslant t \leqslant t_{i+1}. \tag{3.2}$$

Hence

$$q_i(t) \approx g(t) e^{\lambda(t-t_i)} - g(t_i). \tag{3.3}$$

Remark 3.4. For homogeneous problems we should take just some homogeneous solution rather than an inhomogeneous solution for approximation by the adaptive integrator. The initial value should then be some normalized value as is common practice in M.S. algorithms. Therefore a qualitative description of this integration is properly simulated by considering the homogeneous part of (3.1) only. The solution that is actually approximated by the adaptive method can then be assumed to be $s_i(t) := e^{\lambda(t-t_i)}$. The smoothness properties of $s_i(\cdot)$ and $q_i(\cdot)$ (cf. (3.3)) are quite similar. On account of this observation, we shall not treat the homogeneous problem separately.

4. Estimates for the Number of Integration Steps on an Interval. In order to get an idea of the computational cost of the integration, we first investigate the number of gridpoints that is needed for an adaptive solver to approximate a solution on a given interval within a certain accuracy. We do this for the model problem that was defined in Section 3.

Most adaptive methods are based on the use of local error estimates (cf. [17], [2, p. 19]). Let *est* be such an estimate at a certain point t_{old}. Then a new step size, h_{new}, is found from an old step size, h_{old}, using a formula like

$$h_{\text{new}} = 0.9h_{\text{old}}\left|\frac{\varepsilon}{est}\right|^{1/p}. \tag{4.1}$$

Here p is the *local order* of the method, ε the *prescribed accuracy* and 0.9 a safety factor. In particular in our model problem, *est* is determined using approximants for (derivatives of) $q_i(t)$ on the interval (t_i, t_{i+1}). Ideally we must have (cf. (3.3))

$$est \approx \zeta(h_{\text{old}})^p\lambda^p\exp(\lambda t_{\text{old}})g(t_i), \tag{4.2}$$

where ζ is a kind of *error constant* (depending on the integration method). So we find

$$h_{\text{new}} \approx 0.9\frac{1}{\lambda}\left|\frac{\varepsilon}{\zeta g(t_i)}\right|^{1/p}\exp(-\lambda t_{\text{old}}/p). \tag{4.3}$$

Therefore, by virtue of (4.2), we can in principle determine the number of gridpoints that we ideally need on (t_i, t_{i+1}), $\mathcal{N}(t_i, t_{i+1})$ say. We will already be satisfied with a reasonable estimate of $\mathcal{N}(t_i, t_{i+1})$:

***Property* 4.4.** *$\mathcal{N}(t_i, t_{i+1})$, the number of gridpoints* on (t_i, t_{i+1}), that is found by the adaptive integrator is approximately equal to

$$\mathcal{S}(t_i, t_{i+1}) := \frac{10}{9}\left|\frac{\zeta g(t_i)}{\varepsilon}\right|^{1/p} p\{\exp(\lambda[t_{i+1} - t_i]/p) - 1\}.$$

Proof. Let i be fixed. Let us denote the step sizes as found from the method by $s_1, s_2, \ldots, s_N$, i.e. we have gridpoints $t_i, t_i + s_1, \ldots, t_i + \sum_{j=1}^N s_j = t_{j+1}$ (the last step size must be chosen such that we arrive at t_{i+1}). From (4.3) we then find

$$s_{l+1} = \mu\exp\left(-\nu\sum_{j=1}^{l} s_j\right), \quad \text{where } \mu = 0.9\frac{1}{\lambda}\left|\frac{\varepsilon}{\zeta g(t_i)}\right|^{1/p} \text{ and } \nu = \frac{\lambda}{p}. \tag{a}$$

Now let $s(t)$ be a differentiable function, for which

$$s(t) = \mu\exp\left(-\nu\int_1^t s(\tau)\,d\tau\right) \quad \text{for } 1 \leqslant t \leqslant N, \tag{b^1}$$

and

$$(\mathrm{b}^2) \qquad s(1) = s_1 = \mu.$$

Then $s(l)$ can be expected to be a good estimate for s_l. Differencing once in (b^1) yields the Riccati equation

$$(\mathrm{c}) \qquad \frac{d}{dt}s(t) = -\nu s^2(t).$$

Hence

$$s(t) = \frac{1}{\nu}\frac{1}{t+c},$$

where c can be found from (b^2). We obtain

$$(\mathrm{d}) \qquad s(l) = \frac{\mu}{\nu\mu(l-1)+1}.$$

In order to estimate N we use $s(l)$ in (d) as an estimate for s_l and moreover $\Sigma_1^N s_j \approx t_{i+1} - t_i$. We thus obtain

$$N \approx \frac{1}{\mu\nu}\{\exp \nu[t_{i+1} - t_i] - 1\}. \quad \square$$

Remark 4.5. For fixed t_i, the function $S(t) = \mathcal{S}(t_i, t)$ is monotonically increasing for $t > t_i$. By expanding it in powers we moreover see that it is increasing more than proportional with $(t - t_i)$.

Remark 4.6. A most interesting feature of the estimate in Property 4.4 is that it shows the dependence of $\mathcal{S}(t_i, t_{i+1})$ in terms of $\lambda(t_{i+1} - t_i)$ or rather of the incremental growth $\exp(\lambda[t_{i+1} - t_i])$. This is important for the generalization later on. Note that we also found this incremental growth as the important quantity to determine the length of a shooting interval in Section 2.

Although the result of Property 4.4 is based on first order estimates it is remarkably sharp as can be seen from

Example 4.7. A very nice and easy to handle integration code is provided by RKF45 (cf. [5], [17]). Effectively it uses local error estimates with $p = 5$ and $\zeta = 1/1040$. We have applied RKF45 to a model problem, where we have chosen $g \equiv 1$. (Note that the results are independent of i now.) The solution was approximated for several interval lengths, which were determined by prescribing certain *incremental growth values*, $\mathcal{G}$ say. We have also tried several values for ε (the tolerance), between 10^{-3} and 10^{-6} and λ, between 1 and 20. As was to be expected, for a given value of $\mathcal{G}$, the number of points was independent of λ. In Table 4.1 we have listed the actual number of gridpoints, "$\mathcal{N}$" values, followed by their estimates, "$\mathcal{S}$" values, between brackets.

TABLE 4.1

ε \ $\mathcal{G}$	1000	100	10	e	2
10^{-3}	16 (16.4)	8 (8.3)	3 (3.2)	1 (1.2)	1 (0.9)
10^{-4}	25 (26.0)	13 (13.2)	5 (5.1)	2 (1.9)	2 (1.4)
10^{-5}	41 (41.2)	21 (20.9)	8 (8.1)	3 (3.0)	2 (2.3)
10^{-6}	67 (65.4)	34 (33.1)	13 (12.8)	5 (4.5)	4 (3.6)

Note that the estimates are relatively sharper the larger $\mathcal{N}$ is.

5. Optimal Shooting Intervals and Complexity. Suppose we divide the interval (α, β) into k subintervals. For the model problem we saw in Remark 4.5 that, for a given tolerance ε, the number of gridpoints per subinterval increases more than proportional with the interval length. Hence if the function $g(\cdot)$ is smooth we can expect the optimal distribution of the shooting points to be such that each interval contains the same number of gridpoints (at least if k is not too small). This can be thought of as *equidistributing* the shooting points (cf. [7], [14]).

In order to optimize for k we first consider ODE's for which the model problem with $g \equiv 1$ gives an appropriate description regarding the adaptive integration. In this case equidistributed shooting points are also equidistant. From Property 4.4 we therefore find that the total number of gridpoints is estimated by

$$(5.1) \qquad \mathcal{T}(k) := \sum_{i=0}^{k-1} \mathcal{S}(t_i, t_{i+1}) = k \cdot \frac{10}{9} \left|\frac{\zeta}{\varepsilon}\right|^{1/p} p \left\{ \exp\left(\frac{\lambda(\beta - \alpha)}{kp} \right) - 1 \right\}.$$

The computational complexity of the integration can now be estimated by using (5.1). In order to make our discussion as general as possible we include methods like these in [13], [16], where the Φ_i consist of l columns ($l \leqslant n$). We obtain

Property 5.2. Let the computational cost of one integration step be $\kappa_1 n^2$ flops, for some constant κ_1. Let Φ_i consist of l columns for all i. Then the total cost of integrating the fundamental solution is estimated by $\kappa_1 \mathcal{T}(k) n^3$ flops. (A flop is one multiplication plus one addition.)

The constant κ_1 strongly depends on the problem, i.e. κ_1 will be larger if the evaluation of L is more costly.

The cost of the adaptive integrator is more complicated. Of course, in general there are more integrations than actually used gridpoints. In particular restarting the integration may cause quite a bit of overhead. A realistic model is given by

Assumption 5.3. Let the overhead of the adaptive integrator on a subinterval be estimated by $\kappa_1[\gamma_1 + \gamma_2 \mathcal{S}(t_i, t_i + (\beta - \alpha)/k)] n^2$ flops, where κ_1 is as in Property 5.2 and γ_1 and γ_2 are positive constants, independent of k.

Combining Property 5.2 and Assumption 5.3, we find

COROLLARY 5.4. *The total cost of integration is estimated by*

$$\kappa_1 \{ [\mathcal{T}(k)(l + 1) + \gamma_2] + k\gamma_1 \} n^2$$

flops.

There are two other possible sources of computational labor. The first one is the solution of the linear system $PX = F$ (cf. (1.6)). A solution of (1.6) can e.g. be found using one of the special solvers (cf. [8]) or by direct recursion (cf. [2], [12], [16]). Therefore the following makes sense.

Assumption 5.5. Let the computational cost of solving (1.6) or its related subsystem be given by $\kappa_2 k l^3$. If orthonormalization is used, then this has a cost of $\kappa_3 k l n^2$. Here $\kappa_2 \lesssim 2$ and $\kappa_3 \lesssim 1$.

From this we now find

Property 5.6. The total computational cost of the M.S. algorithm is estimated by

$$\left\{ (l + 1 + \gamma_2)\kappa_1 \mathcal{T}(k) + \left(\kappa_1\gamma_1 + \kappa_2 \frac{l^2}{n^2} + \kappa_2 l \right) k \right\} n^2$$

flops. The number of shooting intervals is bounded above by

$$k_{\max} = \lim_{s \to \infty} \mathcal{T}(s) = \left\lceil \frac{10}{9} \left| \frac{\zeta}{\varepsilon} \right|^{1/p} \lambda(\beta - \alpha) \right\rceil$$

(here $\lceil a \rceil$ denotes the smallest integer greater than or equal to a). For the optimal k, k_{opt}, i.e. the number of intervals with minimal cost, there holds

$$k_{\text{opt}} \approx k_{\max}, \qquad \text{if } (l + 1 + \gamma_2) \gtrsim \frac{l^3}{n^2} \frac{\kappa_3}{\kappa_1} + l \frac{\kappa_3}{\kappa_1} + \gamma_1,$$

$$k_{\text{opt}} \approx \left\lceil \frac{10}{9} \left| \frac{\zeta}{\varepsilon} \right|^{1/p} \lambda(\beta - \alpha) \frac{(l + 1 + \gamma_2)\kappa_1}{\kappa_2 l^3/n^2 + \kappa_3 l + \gamma_1} \right\rceil, \quad \text{otherwise.}$$

Proof. The complexity estimate is a consequence of Corollary 5.4 and Assumption 5.5. Furthermore we observe that $\mathcal{T}(s)$ is monotonically decreasing (for $s \geqslant 1$) and approaches

$$\lim_{s \to \infty} \mathcal{T}(s) = \frac{10}{9} \left| \frac{\zeta}{\varepsilon} \right|^{1/p} \lambda(\beta - \alpha);$$

note that $k_{\max}$ should be an integer. Since $\mathcal{S}(\cdot)$ is an estimate for the number of points/interval, we see that

$$\mathcal{T}(k) = \sum_{i=0}^{k-1} \mathcal{S}\left(t_i, t_i + \frac{\beta - \alpha}{k}\right) \geqslant k.$$

Hence, if

$$(l + 1 + \gamma_2) \gtrsim \frac{l^3}{n^2} \frac{\kappa_2}{\kappa_1} + l \frac{\kappa_3}{\kappa_1} + \gamma_1,$$

we have that $k_{\text{opt}} \approx k_{\max}$. Finally, if this is not the case, the approximate optimal value follows from solving

$$(l + 1 + \gamma_2)\kappa_1 \mathcal{T}(k_{\text{opt}}) = \left(\kappa_2 \frac{l^3}{n^2} + \kappa_3 l + \gamma_1 \kappa_1\right) k_{\text{opt}}. \quad \square$$

COROLLARY 5.7. *For the optimal number of points per interval we have*

(i) $$\mathcal{S}\left(t_i, t_i + \frac{\beta - \alpha}{k_{\text{opt}}}\right) \approx 1, \quad \textit{if } (l + 1 + \gamma_2) \gtrsim \frac{l^3}{n^2} \frac{\kappa_2}{\kappa_1} + l \frac{\kappa_3}{\kappa_1} + \gamma_1,$$

(ii) $$\mathcal{S}\left(t_i, t_i + \frac{\beta - \alpha}{k_{\text{opt}}}\right) \approx \frac{\kappa_2 l^3/n^2 + l\kappa_3 + \gamma_1 \kappa_1}{(l + 1 + \gamma_2)\kappa_1}, \quad \textit{otherwise}.$$

The preceding estimates mainly give a good *qualitative* insight in the optimal complexity. On account of the fact, however, that $\mathcal{S}(t_i, t_i + (\beta - \alpha)/k)$ is a relatively more inaccurate estimate of $\mathcal{N}(t_i, t_i(\beta - \alpha)/k)$ the closer

$$\mathcal{N}(t_i, t_i + (\beta - \alpha)/k)$$

is to 1 and at the same time $\mathcal{T}(k)$ (the dominant factor in Property 5.6) is almost independent of k for larger k, it seems advisable to take for $\mathcal{N}(t_i, t_i + (\beta - \alpha)/k)$ values that are not too small integers.

Remark 5.8. If $l < n$, the result of 5.7(ii) seems to indicate that the optimal number of points might be slightly smaller than in case $l = n$. However, as a whole, the qualitative conclusion, viz. about a fixed number of points, is similar.

Having adopted the strategy of a fixed number of gridpoints per interval, for the model problem with $g \equiv 1$, it seems quite reasonable also to use this for more general model problems. Indeed, as observed before, for larger k (i.e. a smaller number of points per interval) an optimal strategy (i.e. equidistribution of shooting points) would require the same number of gridpoints in each interval. Although an analogue to the function $\mathcal{T}(k)$ now is more complicated, it seems very likely that the qualitative conclusions will be the same. We can then generalize further: if it is the equidistribution that counts, then the complexity arguments are likely to be valid for cases where the homogeneous solutions rather grow like $\exp(\int_\alpha^t \lambda(\sigma)\,d\sigma)$ for some nonconstant function $\lambda(\sigma)$.

In order to give the algorithm as much flexibility as possible, we therefore propose to choose $\mathcal{N}$ (= steps/interval) such that it is close to the optimal value, but not too far from 1. From our experiments, 5 was found to be fairly satisfactory when RKF45 was used.

Below we give some examples which illustrate all this.

Example 5.9. Consider the ODE in Example 2.3, now with $r(t)$ such that

$$x(t) = e^t\begin{pmatrix}1\\1\\1\end{pmatrix}.$$

We used an M.S. algorithm that picked a new shooting point either if a certain incremental growth was detected or if the number of gridpoints on the subinterval was equal to a given input parameter. The initial value for the particular solution was chosen to be 0. In Figures 5.1 and 5.2 we have indicated the ***total number of gridpoints*** $\mathcal{N}_T$ as a function of the parameter $\mathcal{G}$ that sets the maximal incremental growth per interval.

As can be expected this number is monotonically increasing for large $\mathcal{G}$. For $\mathcal{G}$ close to 1 this is not the case; this last effect is caused by inefficiencies if the shooting interval is too small (N.B. one step per interval is minimal). In Figures 5.3

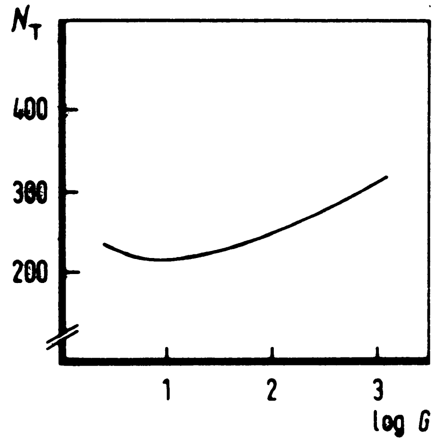

FIGURE 5.1. $\varepsilon = 10^{-4}$

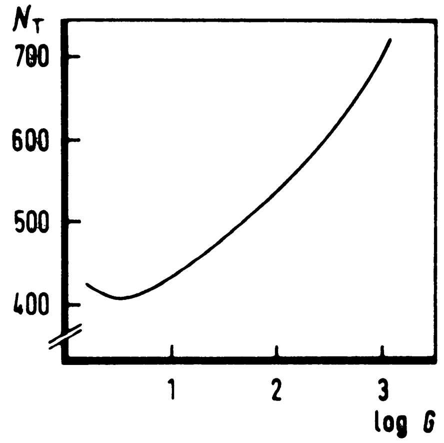

FIGURE 5.2. $\varepsilon = 10^{-6}$

and 5.4 we have indicated the cpu time (in milliseconds) that was needed to compute the desired solution in double precision. Of course, the actual magnitudes of the figures are less important than their qualitative meaning: For $\varepsilon = 10^{-4}$ we again see that too small increments are less efficient; however, for $\varepsilon = 10^{-6}$ this is no longer really to be seen from such a picture. The reason is that for smaller tolerances the number of points per subinterval is larger and therefore the overhead is less felt for $\varepsilon = 10^{-6}$.

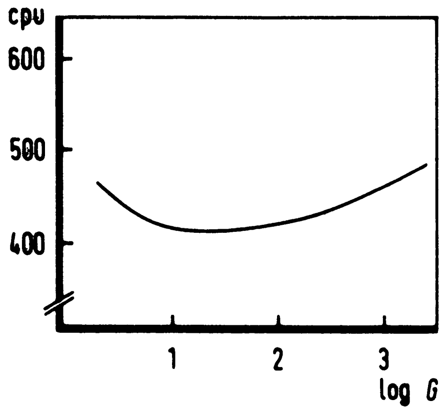

FIGURE 5.3. $\varepsilon = 10^{-4}$

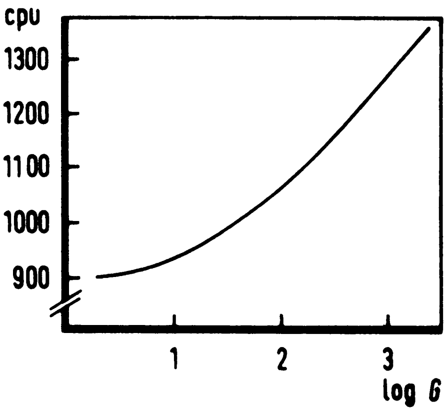

FIGURE 5.4. $\varepsilon = 10^{-6}$

We also have tried the M.S. algorithm with shooting intervals consisting of a fixed number of points. In Table 5.1 we have given the total cpu time (in millisecs) and the number of points (indicated between the brackets) for several values of the tolerance ε and the fixed number of gridpoints per interval $\mathfrak{N}$.

TABLE 5.1

ε \ $\mathfrak{N}$	3	4	5	6	7	8
10^{-3}	317 (108)	308 (113)	296 (116)	290 (121)	289 (125)	290 (130)
10^{-4}	443 (162)	399 (165)	385 (170)	384 (175)	380 (181)	380 (186)
10^{-5}	684 (256)	625 (260)	590 (264)	580 (269)	580 (274)	575 (280)

In all these computations at least 90% of the cpu time was needed for integration.

Example 5.10. As a first major deviation from the model problem, we choose $r(t)$ in 2.3 such that a particular solution is given by

$$\sin 30t \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

This means that this solution will dominate the "activity" of a general solution. In Table 5.2 we have given cpu time and number of grid points (in brackets) for several tolerances and maximal growth per interval $\mathcal{G}$. In Table 5.3 we did the same but now with the number of points per interval fixed (cf. Table 5.1).

TABLE 5.2

ε \ $\mathcal{G}$	100	200	1000	10,000	100,000
10^{-3}	255 (103)	254 (110)	295 (134)	375 (179)	491 (252)
10^{-5}	520 (268)	472 (283)	670 (337)	887 (467)	1215 (653)
10^{-8}	1931 (1100)	2212 (1196)	2619 (1399)	3147 (1690)	3614 (1940)

TABLE 5.3

ε \ $\mathcal{N}$	3	4	5	6	7	8
10^{-3}	254 (90)	242 (87)	235 (89)	237 (90)	257 (101)	255 (100)
10^{-5}	610 (207)	524 (207)	482 (206)	462 (197)	472 (209)	488 (217)
10^{-8}	224 (824)	2031 (824)	1935 (825)	1855 (826)	1805 (828)	1815 (827)

Qualitatively these tables give a result similar to that in the previous example, in favor of choosing $\mathcal{N}$ fixed. Since the activity of $\sin 30t$ is more or less dominating the integration step, we may expect that the total number of gridpoints is fairly independent of $\mathcal{N}$ (= points/interval). Indeed, it turned out that for $\varepsilon = 10^{-8}$, $\mathcal{T} \approx 825$ for all larger values of $\mathcal{N}$, with natural deviations roughly equal to $\mathcal{T} - (\lceil \mathcal{T}/\mathcal{N} \rceil - 1)\mathcal{N}$.

Example 5.11. Consider the ODE

$$\frac{dx}{dt} = \begin{pmatrix} \psi(t) & 0 \\ 2\psi(t) & -\psi(t) \end{pmatrix} x + \begin{pmatrix} (1-\psi(t))e^t \\ 2e^t \end{pmatrix}, \qquad \psi(t) = 20\sin t + 20t\cos t.$$

Choosing the boundary condition $x(0) + x(2) = \binom{1+e^2}{2(1+e^2)}$, one can easily check that $x = \binom{1}{2}e^t$. A fundamental solution is given by

$$F(t) = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} e^{\phi(t)} & 0 \\ 0 & e^{-\phi(t)} \end{pmatrix}, \qquad \phi(t) = 20t\sin t.$$

Apparently the basis solutions in this problem have a significantly varying growth behavior on $[0, 2]$. Tables like 5.2 and 5.3 are given below, viz. Table 5.4 and Table 5.5.

TABLE 5.4

ε \ $\mathcal{G}$	100	200	1000	10,000	100,000
10^{-3}	126 (71)	92 (77)	155 (91)	170 (118)	208 (156)
10^{-5}	256 (180)	264 (194)	325 (236)	399 (304)	504 (392)
10^{-8}	929 (717)	960 (749)	1021 (819)	1089 (871)	1137 (904)

TABLE 5.5

ε \ $\mathfrak{N}$	3	4	5	6	7	8
10^{-3}	129 (61)	127 (63)	126 (65)	127 (68)	125 (70)	129 (73)
10^{-5}	263 (133)	246 (136)	234 (139)	233 (142)	232 (145)	236 (149)
10^{-8}	1014 (526)	971 (529)	926 (532)	879 (536)	871 (539)	868 (542)

Example 5.12. Consider the ODE

$$\frac{dx}{dt} = \begin{pmatrix} t(1-\cos 2t) & 1+t\sin 2t \\ -1+t\sin 2t & t(1+\cos 2t) \end{pmatrix} x + r(t), \qquad 0 \leqslant t \leqslant 4,$$

where $r(t)$ is chosen such that

$$x = \begin{pmatrix} 1+\cos t \\ 1-\sin t \end{pmatrix}.$$

A fundamental solution is given by

$$F(t) = \begin{pmatrix} \cos t & \sin t \\ -\sin t & \cos t \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & e^{t^2} \end{pmatrix}.$$

In this example all solutions are varying with time, and moreover one of the basis solutions has a very rapidly changing growth behavior (as compared to a solution growing like $e^{\lambda t}$, λ constant). Some results are given in Tables 5.6 and 5.7.

TABLE 5.6

ε \ $\mathcal{G}$	100	200	1000	10,000	100,000
10^{-3}	52 (27)	51 (25)	60 (34)	66 (42)	81 (55)
10^{-5}	96 (66)	99 (65)	133 (88)	151 (106)	184 (137)
10^{-8}	348 (264)	327 (250)	394 (309)	420 (335)	435 (343)

TABLE 5.7

ε \ $\mathfrak{N}$	3	4	5	6	7	8
10^{-3}	50 (20)	47 (20)	48 (22)	51 (24)	52 (25)	52 (26)
10^{-5}	97 (46)	91 (47)	87 (48)	86 (48)	86 (50)	87 (50)
10^{-8}	341 (180)	304 (177)	391 (177)	271 (178)	269 (178)	268 (175)

Example 5.13. Our final example is the well-known artificial layer problem (cf. [1, p. 52], [3, p. 455], [16, p. 60]). Consider the ODE

$$x'' + \frac{3\mu}{(\mu+t^2)^2} x = 0, \quad \text{with } x(0.1) = -x(-0.1) = \frac{0.1}{(\mu+0.01)^{1/2}}.$$

For small μ, there is only activity in a region of thickness $\sqrt{\mu}$ around $x = 0$. Hence a reasonable code should choose almost all points in this region. Below, in Tables 5.8 and 5.9, we have given cpu time and number of gridpoints for $\mu = 10^{-6}$.

TABLE 5.8 $(\mu = 10^{-6})$

$\varepsilon \backslash \mathcal{G}$	2	4	6	8	100	1000	10,000
10^{-3}	72 (21)	42 (26)	38 (26)	38 (26)	28 (18)	46 (47)	44 (47)
10^{-5}	58 (40)	54 (48)	54 (49)	54 (47)	52 (56)	58 (62)	64 (95)
10^{-8}	114 (144)	112 (159)	112 (163)	110 (155)	122 (187)	128 (190)	175 (290)

TABLE 5.9 $(\mu = 10^{-6})$

$\varepsilon \backslash \mathcal{N}$	3	4	5	6	7	8
10^{-3}	39 (28)	33 (26)	30 (26)	33 (29)	31 (28)	32 (31)
10^{-5}	60 (52)	56 (54)	54 (54)	49 (53)	49 (53)	47 (54)
10^{-8}	181 (182)	150 (181)	145 (181)	129 (181)	130 (181)	120 (181)

Since the basis solutions do not grow substantially on this interval (viz. a factor 500-1500, depending on the accuracy of the computation), a strategy with $\mathcal{G}$ fixed means that a choice of $\mathcal{G} > 1500$ will result in single shooting. It appears that the integrator needs a number of steps before it has relaxed the step size, after having passed the high activity region. Because of the starting strategy of RKF45, a new shooting point outside of this high activity region will automatically relax the step size. This explains why the total number of steps $\mathcal{N}_T$ is independent of the $\mathcal{N}$. The results are also in agreement (perhaps better, even taking into account that we used the adaptivity feature for one solution only) with the results in [16], where e.g. for $\varepsilon = 10^{-8}$, $\mathcal{N}_T = 190$. The numbers given in [3] do not lend themselves to comparison. However, since they are computed by a code, which is designed to handle *nonlinear* problems, they involve the computation of 5 basis solutions, instead of 2 like the code in [16] does; hence we do not expect the results in [3] to be competitive.

Conclusion. The use of an adaptive integrator is undoubtedly one of the best features of M.S. as e.g. Example 5.13 confirms (see also the conclusion in [16]). If one chooses to determine the shooting points by checking some kind of dependence of solution vectors (cf. [4], [16]), then one does effectively the same as checking the growth of solutions (cf. [13]), as this growth generates errors, which in turn cause the dependence. However, as can be seen, both from the examples and from the theory, a reliable growth tolerance does not seem obvious. In fact, since rounding errors of order ξ (the machine precision) are often considerably smaller than (global!) discretization errors of order ε (the tolerance), an acceptable bound for the growth (from a stability point of view) would be $O(\varepsilon/\xi)$, see also Sections 2 and 6. Note that in our examples this bound would be at least as big as 10^8, which would lead to extremely inefficient strategies of course. In [2], [3], [18] the M.S. codes require the user to specify the shooting points. Only if a good knowledge of the growth of solutions is available this may work well. After what we have found above, however, it goes without saying that such a strategy may be far from optimal, regarding the complexity. In that respect the results in [3] should be interpreted in a relative sense, i.e. only in relation to that special implementaion. Concluding we may say that the fixed number of integration points per interval option seems to work equally well or better than the others. Moreover it is based on a theoretical model which lends itself to a more extensive analysis.

6. Minimization of Storage. As we noted in Section 1, a third optimality criterion is given by the requirement to minimize the storage for the matrices A_i and B_i and the vectors F_i, i.e. to minimize the order of the matrix P (see (1.6)). We shall now show that we can reduce this order without really affecting the complexity and the stability, the latter at least to a certain extent. Such a reduction of the order is also important in order to make the previously obtained complexity strategy practical; indeed, following the recommendations of Section 5, we may end up with a very large value for k (number of shooting intervals), which might make the proposed strategy less attractive, if not impractical.

The solution to those problems is very simple though. Indeed, all we have to do is to assemble as much of the recurrence relations (1.4), as is allowed by stability considerations, in order to build a condensed "major shooting step". Below we describe this condensation.

If for a given t_j, the point t_{j+l} is such that

$$\|\Phi_{j+l-1}(t_{j+l})[\Phi_{j+l-1}(t_{j+l-1})]^{-1}\cdots\Phi_j(t_{j+1})[\Phi_j(t_j)]^{-1}\|\leqslant\frac{\varepsilon}{\xi\kappa k},\tag{6.1}$$

then it follows, from what has been observed in Section 2 (cf. (2.1)) that forward recursion on (t_j, t_{j+l}) gives global rounding errors of maximal order ε. Hence rather than (1.4) for $i = j,\dots, j + l$, we use the major recursion step

$$C_{j+l-1}v_j - B_{j+l-1}v_{j+l} = G_{j+l-1},\tag{6.2}$$

where C_{j+l-1} and G_{j+l-1} are recursively defined by (6.3) and (6.4), respectively:

$$C_j = A_j;\quad C_{j+s+1} = A_{j+s+1}B_{j+s}^{-1}C_{j+s},\qquad s = 0,\dots, l-1,\tag{6.3}$$

$$G_j = F_j;\quad G_{j+s+1} = A_{j+s+1}B_{j+s}^{-1}G_{j+s} + F_{j+s+1},\qquad s = 0,\dots, l-1.\tag{6.4}$$

It goes without saying that the condensed relations like (6.2) together with the BC give a significantly lower order M.S. system if the ratio ε/ξ is large. Moreover, condensing the system plus solving it has approximately the same complexity as solving the uncondensed one; for this one should note that each recursion step approximately requires l^3 flops (cf. Example 6.6).

Remark 6.5. The recursive formulations (6.3) and (6.4) show that the matrices C_{j+l-1} and G_{j+l-1} can be found from updating C_{j+s} and G_{j+s} at each minor shooting step (proceeding from t_j to t_{j+l}). As a consequence one only needs some limited work space, but no extra memory.

The threshold criterion (6.1) may be implemented in the following way. Assuming the problem is well conditioned, we just choose some constant η, equal to 10 say, to serve as an upper bound for κ. A good guess for k is given by $(\beta - \alpha)/(t_{j+l} - t_j)$.

Hence, in practice we suggest to use

$$\frac{\varepsilon}{\xi\kappa k} = \frac{\varepsilon}{\xi}\frac{1}{\eta}\frac{t_{j+l}-t_j}{\beta-\alpha}. \tag{6.5}$$

Example 6.6. We performed this condensation strategy for the problem given in Example 2.3, for which we used an M.S. code based on RKF45 and 5 integration steps per (smaller) shooting interval. The major shooting intervals were found by requiring that the maximal increment should not exceed some preset value $\mathcal{G}$. Table 6.1 now confirms our expectation that the total *cpu* time (in millisecs) is practically independent of this value of $\mathcal{G}$ indeed.

TABLE 6.1. $\varepsilon = 10^{-4}$

$\mathcal{G}$	10^3	10^4	10^6	10^8	10^{10}	10^{12}	10^{14}	10^{16}
k	10	7	5	4	3	3	3	2
cpu	415	412	412	416	408	407	410	414

Mathematisch Instituut
Katholieke Universiteit
Toernooiveld
6525 ED Nijmegen, The Netherlands

1. B. CHILDS (ed.), *Codes for Boundary-Value Problems in Ordinary Equations*, Lecture Notes in Comput. Sci., Vol. 76, Springer-Verlag, Berlin, 1978.

2. P. DEUFLHARD, "Recent advances in multiple shooting techniques," In: *Computational Techniques for Ordinary Differential Equations* (Gladwell, Sayers, eds.), Academic Press, London, 1980, pp. 217–272.

3. H.-J. DIEKHOFF, P. LORY, H. J. OBERLE, H.-J. PESCH, P. RENTROP & R. SEYDEL, "Comparing routines for the numerical solution of initial value problems for ordinary differential equations in multiple shooting," *Numer. Math.*, v. 27, 1977, pp. 449–469.

4. S. D. CONTE, "The numerical solution of linear boundary value problems," *SIAM Rev.*, v. 8, 1966, pp. 309–321.

5. G. E. FORSYTH, M. A. MALCOLM & C. B. MOLER, *Computer Methods for Mathematical Computations*, Prentice-Hall, Englewood Cliffs, N.J., 1977.

6. J. H. GEORGE & R. W. GUNDERSON, "Conditioning of linear boundary value problems," *BIT*, v. 12, 1972, pp. 172–181.

7. J. KAUTSKY & N. NICHOLS, "Equidistributing meshes with constraints," *SIAM J. Sci. Statist. Comput.*, v. 1, 1980, pp. 499–511.

8. H. B. KELLER, *Numerical Solution of Two Point Boundary Value Problems*, CBMS series 24, SIAM, Philadelphia, Pa., 1976.

9. R. M. M. MATTHEIJ, "The conditioning of linear boundary value problems," *SIAM J. Numer. Anal.*, v. 19, 1982, pp. 963–978.

10. R. M. M. MATTHEIJ, "Estimates for the errors in the solutions of linear boundary value problems, due to perturbations," *Computing*, v. 27, 1981, pp. 299–318.

11. R. M. M. MATTHEIJ, "Accurate estimates for the fundamental solutions of discrete boundary value problems," *J. Math. Anal. Appl.* (To appear.)

12. R. M. M. MATTHEIJ & G. W. M. STAARINK, "An efficient algorithm for solving general linear two point boundary value problems," *SIAM J. Sci. Statist. Comput.*, v. 4, 1983.

13. M. R. OSBORNE, "The stabilized march is stable," *SIAM J. Numer. Anal.*, v. 16, 1979, pp. 923–933.

14. V. PEREYRA & E. G. SEWELL, "Mesh selection for discrete solution of boundary value problems in ordinary differential equations," *Numer. Math.*, v. 23, 1975, pp. 261–268.

15. R. D. RUSSELL & J. CHRISTIANSEN, "Adaptive mesh selection strategies for solving boundary value problems," *SIAM J. Numer. Anal.*, v. 15, 1978, pp. 59–80.

16. M. R. SCOTT & H. A. WATTS, "Computational solution of linear two point boundary value problems via orthonormalization," *SIAM J. Numer. Anal.*, v. 14, 1977, pp. 40–70.

17. L. F. SHAMPINE & H. A. WATTS, "Solving non stiff ordinary differential equations, the state of the art," *SIAM Rev.*, v. 18, 1976, pp. 376–411.

18. J. STOER & R. BULIRSCH, *Einführung in die numerische Mathematik* II, Springer-Verlag, Berlin, 1973.