Advances in Applied Mathematics and Mechanics
Adv. Appl. Math. Mech., Vol. **5**, No. 3, pp. 269-308

DOI: 10.4208/aamm.12-m1211
June 2013

A Matrix-Vector Operation-Based Numerical Solution Method for Linear m-th Order Ordinary Differential Equations: Application to Engineering Problems

M. Aminbaghai[1], M. Dorn[1,2], J. Eberhardsteiner[1] and B. Pichler[1,*]

[1] *Institute for Mechanics of Materials and Structures, Vienna University of Technology (TU Wien), Karlsplatz 13/202, A-1040 Vienna, Austria*
[2] *Linnaeus University, Department of Building and Energy Technology, Lückligs Plats 1, S-35195 Växjö, Sweden*

Received 17 February 2012; Accepted (in revised version) 8 February 2013

Available online 30 April 2013

Abstract. Many problems in engineering sciences can be described by linear, inhomogeneous, m-th order ordinary differential equations (ODEs) with variable coefficients. For this wide class of problems, we here present a new, simple, flexible, and robust solution method, based on piecewise exact integration of local approximation polynomials as well as on averaging local integrals. The method is designed for modern mathematical software providing efficient environments for numerical matrix-vector operation-based calculus. Based on cubic approximation polynomials, the presented method can be expected to perform (i) similar to the Runge-Kutta method, when applied to stiff initial value problems, and (ii) significantly better than the finite difference method, when applied to boundary value problems. Therefore, we use the presented method for the analysis of engineering problems including the oscillation of a modulated torsional spring pendulum, steady-state heat transfer through a cooling web, and the structural analysis of a slender tower based on second-order beam theory. Related convergence studies provide insight into the satisfying characteristics of the proposed solution scheme.

AMS subject classifications: 34A30, 34B60

Key words: Numerical integration, polynomial approximation, ODE, variable coefficients, initial conditions, boundary conditions, stiff equation.

1 Introduction

Many problems in engineering sciences can be described by linear, inhomogeneous, m-th order, ordinary differential equations (ODEs) with variable coefficients

*Corresponding author.
Email: mehdi.aminbaghai@tuwien.ac.at (M. Aminbaghai), michael.dorn@tuwien.ac.at (M. Dorn), josef.eberhardsteiner@tuwien.ac.at (J. Eberhardsteiner), bernhard.pichler@tuwien.ac.at (B. Pichler)

$$\sum_{i=0}^{m} a_i(x)y^{(i)}(x) = r(x), \qquad y^{(i)}(x) \overset{\text{def}}{=} \frac{d^i y(x)}{dx^i}. \tag{1.1}$$

This includes problems such as oscillations, one-dimensional heat transfer, or beam bending. Classical numerical solution methods for related initial value problems and boundary value problems, respectively, include the backward Euler method, the Runge-Kutta method, Adams-Bashforth-Moulton methods [1–3], backward differentiation formulas (BDFs) [4], Rosenbrock methods [5, 6], as well as the finite difference method.

During the last decades, quite a number of new methods have been presented for specific classes of problems. For stiff first-order initial value problems, numerical differentiation formulas (NDFs) and a modified Rosenbrock method were presented in [7]. Meshless local Petrov-Galerkin methods [8, 9] have become popular for solving thin beam problems in the framework of the first-order theory. Discontinuous Galerkin methods [10] were shown to be efficient for second-order ODEs with periodic coefficient functions and periodic perturbation functions. Piecewise linearized methods [11] have turned out to be useful for non-stiff initial value problems. The differential quadrature method [12, 13] has gained interest of researchers dealing with dynamic problems, including three-dimensional vibration of functionally graded circular plates analyzed based on the one-dimensional differential quadrature method [14]. Also integration schemes based on approximations provided by radial basis functions have become popular [15, 16].

When selecting a specific method out of the variety of available approaches capable to solve problems subsumable under (1.1), there is typically a trade-off between numerical efficiency and versatility. There exist highly efficient methods which are designed for specific classes of problems, i.e., for ODEs with a specific order of differentiation m, for specific types of coefficient functions $a_i(x)$, for specific perturbation functions $r(x)$, and for specific types of boundary conditions; but such specialized methods frequently exhibit a limited utilizability for other specific problems. Methods designed for initial value problems, for instance, may lack user friendliness when applying them to general boundary value problems which include boundary conditions referring to the end of the integration interval. Highly robust and flexible methods, in turn, which are generally applicable to virtually all problems subsumable under (1.1), commonly cannot compete with the aforementioned, highly efficient methods, when a specific problem at hand has to be solved.

This is the motivation to present a new, simple, flexible, and robust method for the solution of ordinary differential equations such as (1.1), based on piecewise exact integration of local approximation polynomials as well as on averaging local integrals. The method is designed to comply with the following requirement profile:

- The method should be applicable to any linear, inhomogeneous, m-th order ODE with variable coefficients – either associated with an initial value problem, or with a

boundary value problem – without raising the need to adapt the method for specific properties of the underlying differential equation.

- The method should be easy to implement into modern mathematical software providing efficient environments for numerical matrix-vector operation-based calculus.
- Extensions of the method towards higher-order integration schemes with increased order of accuracy should be straightforward.

The paper is structured as follows. Inspired by [17], Section 2 presents a numerical integration method for smooth functions, based on piecewise exact integration of cubic approximation polynomials. Section 3 builds on this method to derive a numerical solution method for linear m-th order ODEs with smooth perturbation function and smooth coefficient functions. After comparing the proposed method with classical methods, by applying them to a stiff initial value problem and to a benchmark boundary value problem (Section 4), we use the presented method to analyze engineering problems including the oscillation of a modulated torsional spring pendulum (Section 5), steady-state heat transfer through a cooling web (Section 6), and structural analysis of a slender tower based on second-order beam theory (Section 7). In Section 8 we discuss the results, leading to conclusions and to a future outlook.

2 Approximation polynomial-based numerical integration of smooth functions

Herein, we first briefly recall and then extend the numerical integration method used in [17]. This method allows for integrating smooth functions $f(x)$, based on piecewise exact integration of local quadratic approximation polynomials as well as on averaging such local integrals. To this end, the domain of interest $x_a \leq x \leq x_b$ is subdivided into $(n-1)$ intervals of identical size

$$\lambda = \frac{x_b - x_a}{n-1}. \tag{2.1}$$

At the n interval boundaries $x_a = x_1$, x_2, x_3, $\cdots$, x_{n-1}, and $x_n = x_b$, the function $f(x)$ is evaluated, delivering the function values $f_1 = f(x_1)$, $f_2 = f(x_2)$, $\cdots$, $f_{n-1} = f(x_{n-1})$, and $f_n = f(x_n)$. In the next step, quadratic polynomials are used to approximate the function $f(x)$ in all subdomains consisting of two neighboring intervals. This results in $(n-2)$ local approximation polynomials. In more detail, in the first interval $[x_1;x_2]$, the function $f(x)$ is approximated by the quadratic polynomial which is defined in the subdomain $[x_1;x_3]$, and which exactly reproduces the function values f_1, f_2, and f_3. Similarly, in the last interval $[x_{n-1};x_n]$, the function $f(x)$ is approximated by the quadratic polynomial which is defined in the subdomain $[x_{n-2};x_n]$, exactly reproduces the function values f_{n-2}, f_{n-1}, and f_n. In all other intervals in between, two local approximation polynomials are available, e.g., in the second interval $[x_2;x_3]$, the function $f(x)$ is approximated

by the aforementioned quadratic polynomial defined in subdomain $[x_1;x_3]$ and exactly reproducing the function values f_1, f_2, and f_3, as well as, independently, by the quadratic polynomial defined in subdomain $[x_2;x_4]$ and exactly reproducing the function values f_2, f_3, and f_4. Next, all approximation polynomials are integrated exactly over the two intervals in which they were designed to approximate $f(x)$. This implies that, in the first and in the last interval, $f(x)$ is approximately integrated only once, while in all other intervals, $f(x)$ is approximately integrated twice, and the values of these two integrals are averaged in each of these intervals. This strategy allows for expressing the approximate integration of $f(x)$ in the interval from x_1 to x_k as simple as [17]

$$\int_{x_1}^{x_k} f(x)\,dx \approx \sum_{s=1}^{k} A_{ks}^{1} f_s \quad \text{with} \quad A_{ks}^{1} \overset{\text{def}}{=} \frac{\lambda}{24} A_{ks}, \tag{2.2}$$

where A_{ks} is a matrix of real numbers, which reads, e.g., for $\max k = \max s = 6$, as [17]

$$A_{ks} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 10 & 16 & -2 & 0 & 0 & 0 \\ 8 & 32 & 8 & 0 & 0 & 0 \\ 9 & 27 & 27 & 9 & 0 & 0 \\ 9 & 28 & 22 & 28 & 9 & 0 \\ 9 & 28 & 23 & 23 & 28 & 9 \end{bmatrix}. \tag{2.3}$$

For larger values of $\max k = \max s$, additional components of A_{ks} can be found easily, considering the following rules

$$\forall k \le 7, \qquad \forall s \ge 7: \quad A_{ks} = 0, \tag{2.4a}$$

$$\forall k \ge 7: \qquad A_{k1} = A_{kk} = 9, \quad A_{k2} = A_{kk-1} = 28, \quad A_{k3} = A_{kk-2} = 23,$$
$$A_{k4} = \cdots = A_{kk-3} = 24, \quad A_{kk+1} = A_{kk+2} = \cdots = A_{kn} = 0. \tag{2.4b}$$

It will turn out to be useful for the ODE solution method proposed in Section 3, to extend the definition of the integration matrix introduced in [17], see (2.2), towards

$$A_{ks}^{i} \overset{\text{def}}{=} \begin{cases} \delta_{ks}, & i = 0, \\ \dfrac{\lambda}{24} A_{ks}, & i = 1, \\ \displaystyle\sum_{t=1}^{k} A_{kt}^{1} A_{ts}^{i-1}, & i \ge 2, \end{cases} \tag{2.5}$$

where δ_{ks} stands for the Kronecker delta, i.e., $\delta_{ks}=1$ for $k=s$ and $\delta_{ks}=0$ for $k\neq s$. Integration matrices A_{ks}^{i}, namely, allow for straightforward i-fold numerical integration of a smooth function $f(x)$ as

$$\underbrace{\int_{x_1}^{x_k} \int \cdots \int}_{i \text{ integrals}} f(x) dx^i \approx \sum_{s=0}^{n} A_{ks}^{i} f_s. \tag{2.6}$$

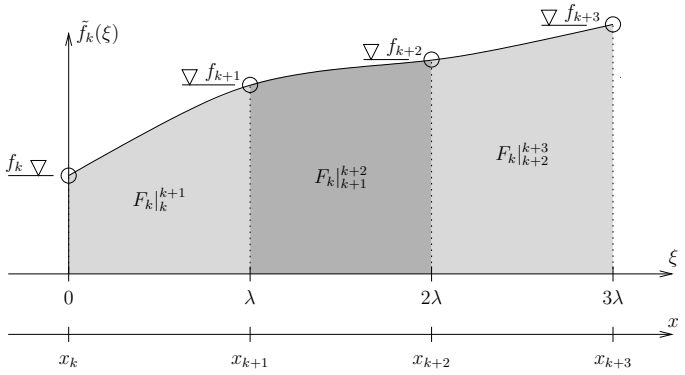


Figure 1: Cubic polynomial $\tilde{f}_k(\xi)$ approximating the function $f(x)$ in the interval $[x_k,x_{k+3}]$.

As for (2.6), we note that all the integration constants, which are to be introduced when performing the $(i-1)$ indefinite integrations, were set equal to zero, for the sake of simplicity. Finally, we mention a formal similarity between the *integration* matrices introduced in (2.5) and *differentiation* matrices used in the framework of differential quadrature methods [18].

In order to provide more insight into the philosophy of the described integration method, and in order to further increase the accuracy of integration, we now extend the approach towards piecewise exact integration of cubic approximation polynomials. To this end, the domain of interest $x_a \le x \le x_b$ is again subdivided into $(n+1)$ intervals of identical size λ, see (2.1). Within each subdomain consisting of three neighboring intervals, the function $f(x)$ is approximated by a cubic polynomial. Focusing on three neighboring intervals covering the domain $[x_k;x_k+3\lambda]$, we introduce a local coordinate ξ with origin at x_k, reading as: $\xi = x-x_k$ (Fig. 1). The cubic approximation polynomial $\tilde{f}_k(\xi)$ satisfying $\tilde{f}_k(\xi=0)=f_k$, $\tilde{f}_k(\xi=\lambda)=f_{k+1}$, $\tilde{f}_k(\xi=2\lambda)=f_{k+2}$, and $\tilde{f}_k(\xi=3\lambda)=f_{k+3}$ reads as:

$$\tilde{f}_k(\xi) = -\Big(\frac{f_k-3f_{k+1}+3f_{k+2}-f_{k+3}}{6\lambda^3}\Big)\xi^3+\Big(\frac{2f_k-5f_{k+1}+4f_{k+2}-f_{k+3}}{2\lambda^2}\Big)\xi^2 - \Big(\frac{11f_k-18f_{k+1}+9f_{k+2}-2f_{k+3}}{6\lambda}\Big)\xi+f_k. \tag{2.7}$$

Integrating the cubic approximation polynomial $\tilde{f}_k(\xi)$ exactly from x_k to x_{k+1}, from x_{k+1} to x_{k+2}, and from x_{k+2} to x_{k+3}, respectively, yields

$$F_k|_k^{k+1} = \int_{x_k}^{x_{k+1}} \tilde{f}_k(x)dx = \int_0^{\lambda} \tilde{f}_k(\xi)d\xi = \frac{\lambda}{24}(9f_k+19f_{k+1}-5f_{k+2}+f_{k+3}), \tag{2.8a}$$

$$F_k|_{k+1}^{k+2} = \int_{x_{k+1}}^{x_{k+2}} \tilde{f}_k(x)dx = \int_{\lambda}^{2\lambda} \tilde{f}_k(\xi)d\xi = \frac{\lambda}{24}(-f_k+13f_{k+1}+13f_{k+2}-f_{k+3}), \tag{2.8b}$$

$$F_k|_{k+2}^{k+3} = \int_{x_{k+3}}^{x_{k+3}} \tilde{f}_k(x)dx = \int_{2\lambda}^{3\lambda} \tilde{f}_k(\xi)d\xi = \frac{\lambda}{24}(f_k-5f_{k+1}+19f_{k+2}+9f_{k+3}), \tag{2.8c}$$

see also Fig. 1. Based on (2.8), integration of $f(x)$ from $x=x_1$ to $x_{k=2,3,4,\cdots,n}$ is expressed as

$$\int_{x_1}^{x_k} f(x)dx \approx \begin{cases} F_1|_1^2, & \text{for } k=2, \\ F_1|_1^2+F_1|_2^3, & \text{for } k=3, \\ F_1|_1^2+F_1|_2^3+F_1|_3^4, & \text{for } k=4, \\ F_1|_1^2+\frac{1}{2}(F_1|_2^3+F_2|_2^3)+\frac{1}{2}(F_1|_3^4+F_2|_3^4)+F_2|_4^5, & \text{for } k=5, \\ F_1|_1^2+\frac{1}{2}(F_1|_2^3+F_2|_2^3)+\frac{1}{3}(F_1|_3^4+F_2|_3^4+F_3|_3^4) & \\ \quad +\frac{1}{2}(F_2|_4^5+F_3|_4^5)+F_3|_5^6, & \text{for } k=6, \\ F_1|_1^2+\frac{1}{2}(F_1|_2^3+F_2|_2^3)+\frac{1}{3}(F_1|_3^4+F_2|_3^4+F_3|_3^4) & \\ \quad +\frac{1}{3}(F_2|_4^5+F_3|_4^5+F_4|_4^5)+\frac{1}{2}(F_3|_5^6+F_4|_5^6)+F_4|_6^7, & \text{for } k=7, \\ F_1|_1^2+\frac{1}{2}(F_1|_2^3+F_2|_2^3)+\frac{1}{3}\sum_{s=1}^{k-5}\left(F_s|_{s+2}^{s+3}+F_{s+1}|_{s+2}^{s+3}+F_{s+2}|_{s+2}^{s+3}\right) & \\ \quad +\frac{1}{2}(F_{k-4}|_{k-2}^{k-1}F_{k-3}|_{k-2}^{k-1})+F_{k-3}|_{k-1}^{k}, & \text{for } k\geq 6. \end{cases} \tag{2.9}$$

Eq. (2.9) implies that if $f(x)$ is to be integrated from x_1 to x_2, from x_1 to x_3, or from x_1 to x_4, respectively, then the method sums up exact integrals related to the approximation polynomial defined in subdomain $[x_1;x_4]$, see the definitions for $k=2$, $k=3$, and $k=4$ in (2.9). Next we consider that $f(x)$ is to be integrated from x_1 to x_5. In this case, exact integrals of the two approximation polynomials defined in the subdomains $[x_1;x_4]$ and $[x_2;x_5]$ are considered. The second and third interval ($[x_2;x_3]$ and $[x_3;x_4]$) are both within the domain of definition of the two approximation polynomials. The related integrals are averaged in these two intervals, see the terms $(F_1|_2^3+F_2|_2^3)/2$ and $(F_1|_3^4+F_2|_3^4)/3$ in definition for $k=5$ in (2.9). Integration in the first and the fourth interval ($[x_1;x_2]$ and $[x_4;x_5]$), in turn, is based on first and the second approximation polynomial, respectively, see the terms $F_1|_1^2$ and $F_2|_4^5$ in the definition for $k=5$ in (2.9). In the general case, where $f(x)$ is to be integrated over the first five (or more) intervals, three (or more) approximation polynomials are considered, whereby two intervals are lying within the domain of two approximation polynomials, and one (or more) intervals is (are) lying within the domain of definition of three approximation polynomials. In these intervals, the exact integrals of the individual approximation polynomials are again averaged, see definitions for $k=6$, $k=7$, and $k\geq 6$ in (2.9).

In order to provide more detailed insight into the properties of the described integration method, we consider the following illustrative example which refers to integration of the function

$$y(x)=\sin(x)\exp\left(-\frac{x}{10}\right) \tag{2.10}$$

in the interval $[x_a,x_b]=[0,3\pi]$ based on seven grid points ($n=7$) with coordinates

$$x_k=(k-1)\frac{\pi}{2}, \quad k=\{1,2,3,4,5,6,7\}. \tag{2.11}$$

Denoting the primitive of $y(x)$ as $Y(x)$, and using the initial condition $Y(x=0)$, we estimate the function values of $Y(x)$ at the grid point positions (2.11) using the described integration method, see (2.2),

$$Y_k = \int_{x_1}^{x_k} y(x)dx \approx \sum_{s=1}^{k} \frac{\lambda}{24} A_{ks} y_s, \quad k=\{1,2,3,4,5,6,7\}. \tag{2.12}$$

The grid point distance follows from (2.1) as $\lambda=\pi/2$, the values of the integration matrix A_{ks} from (2.3) and (2.4), and the function values of (2.10) at all grid points positions (2.11) read as

$$y_1=0, \quad y_2=0.8546, \quad y_3=0, \quad y_4=-0.6242, \quad y_5=0, \quad y_6=0.4559, \quad y_7=0. \tag{2.13}$$

According to (2.12), estimates for the sought function values of $Y(x)$ follow from the following simple matrix-vector multiplication:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \\ Y_6 \\ Y_7 \end{bmatrix} \approx \frac{\pi/2}{24} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 10 & 16 & -2 & 0 & 0 & 0 & 0 \\ 8 & 32 & 8 & 0 & 0 & 0 & 0 \\ 9 & 27 & 27 & 9 & 0 & 0 & 0 \\ 9 & 28 & 22 & 28 & 9 & 0 & 0 \\ 9 & 28 & 23 & 23 & 28 & 9 & 0 \\ 9 & 28 & 23 & 24 & 23 & 28 & 9 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0.8546 \\ 0 \\ -0.6242 \\ 0 \\ 0.4559 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.8950 \\ 1.7899 \\ 1.1426 \\ 0.4222 \\ 0.8951 \\ 1.4212 \end{bmatrix}. \tag{2.14}$$

The obtained results (2.14) turn out to be of satisfactory quality, when comparing with the available analytical solution, see Fig. 2, and considering the very coarse discretization of the integration interval.

The integration method turns out to be surprisingly simple, see (2.14), given the rather advanced underlying integration concept which is illustrated next. The given function

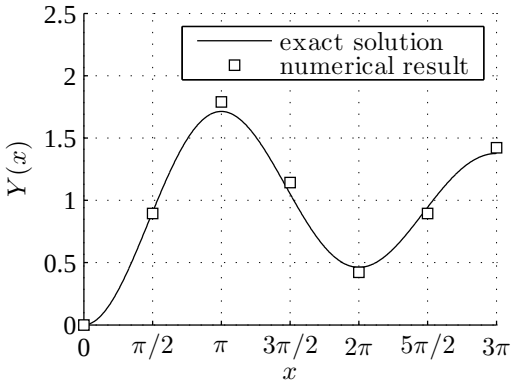


Figure 2: Comparison of function values Y_k with $k=\{1,2,3,4,5,6,7\}$ obtained with the described integration method, see (2.10) to (2.14) and points marked with squares, with the analytical result $Y(x)=\frac{100}{101}-\left[\frac{100}{101}\cos(x)+\frac{10}{101}\sin(x)\right]\exp\left(-\frac{x}{10}\right)$ illustrated by a solid line.

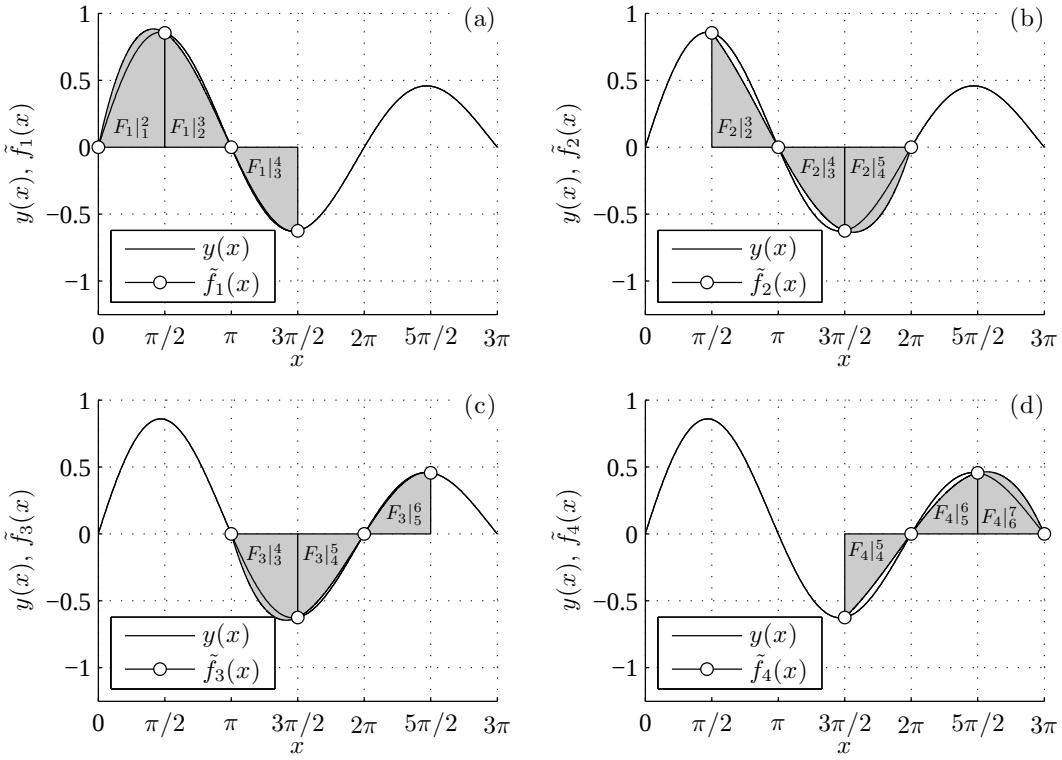


Figure 3: Concept of the described integration method: the function $y(x)$ is approximated, in all subintervals containing four neighboring grid points, by means of cubic polynomials, see (a) for approximation polynomial $\tilde{f}_1(x)$, (b) for $\tilde{f}_2(x)$, (c) for $\tilde{f}_3(x)$, and (d) for $\tilde{f}_4(x)$.

$y(x)$ is approximated, in all subintervals containing four neighboring grid points, by means of cubic polynomials, see Fig. 3. The approximation polynomials are analytically integrated between the grid points, see shaded areas in Fig. 3. Since the subintervals are overlapping each other, also the computed integrals refer to overlapping domains and, therefore, they are averaged arithmetically, according to (2.9).

Based on exactly integrating local approximation polynomials and on averaging such local integrals, see (2.7) to (2.9), an integration method for i-fold numerical integration of a smooth function $f(x)$ can be expressed by analogy to (2.5) and (2.6) as

$$\underbrace{\int_{x_1}^{x_k}\int\cdots\int}_{i \text{ integrals}} f(x)dx^i \approx \sum_{s=0}^{k} A_{ks}^i f_s, \qquad A_{ks}^i = \begin{cases} \delta_{ks}, & i=0, \\ \dfrac{\lambda}{144}A_{ks}, & i=1, \\ \displaystyle\sum_{t=0}^{n} A_{kt}^1 A_{ts}^{i-1}, & i\geq 2. \end{cases} \tag{2.15}$$

As for (2.15), we note that all the integration constants, which are to be introduced when performing the $(i-1)$ indefinite integrations, were set equal to zero, for the sake of sim-

plicity. A_{ks} in (2.15) is a matrix of real numbers, which follows from specifying (2.9) for (2.8) and from setting the result equal to (2.15) specified for $i=1$; e.g., for $\max k=\max s=10$, A_{ks} reads as

$$A_{ks}=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 54 & 114 & -30 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 48 & 192 & 48 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 54 & 162 & 162 & 54 & 0 & 0 & 0 & 0 & 0 & 0 \\ 54 & 168 & 132 & 168 & 54 & 0 & 0 & 0 & 0 & 0 \\ 53 & 171 & 136 & 136 & 171 & 53 & 0 & 0 & 0 & 0 \\ 53 & 170 & 139 & 140 & 139 & 170 & 53 & 0 & 0 & 0 \\ 53 & 170 & 138 & 143 & 143 & 138 & 170 & 53 & 0 & 0 \\ 53 & 170 & 138 & 142 & 146 & 142 & 138 & 170 & 53 & 0 \\ 53 & 170 & 138 & 142 & 145 & 145 & 142 & 138 & 170 & 53 \end{bmatrix}. \tag{2.16}$$

For larger values of $\max k=\max s$, additional entries of A_{ks} can be found easily, considering the rules

$$\forall k\leq 11, \qquad \forall s\geq 11: \quad A_{k,s}=0, \tag{2.17a}$$

$$\begin{aligned} \forall k\geq 11: \qquad & A_{k,1}=A_{k,k}=53, \quad A_{k,2}=A_{k,k-1}=170, \quad A_{k,3}=A_{k,k-2}=138, \\ & A_{k,4}=A_{k,k-3}=142, \quad A_{k,5}=A_{k,k-4}=145, \\ & A_{k,6}=\cdots=A_{k,k-5}=144, \quad A_{k,k+1}=\cdots=A_{k,n}=0. \end{aligned} \tag{2.17b}$$

Finally, we compare the method based on quadratic approximation polynomials with the method based on cubic approximation polynomials, and we note that single and multiple integration of a pointwisely defined function is *formally* based on (2.5), independent of the used order of local approximation, compare (2.5) with (2.15). The integration matrices A^i_{ks} depend on the degree of the used approximation polynomials; for quadratic polynomials, see (2.3)-(2.5), and for cubic polynomials, see (2.15)-(2.17). Further extensions of the integration matrices A^i_{ks} towards locally quartic or even higher-order approximation polynomials is straightforward and follows the line described above.

3 Numerical solution of linear m-th order ODEs with smooth perturbation function and smooth coefficient functions

For the sake of clarity of presentation, we start with index notation, followed by the transition to matrix-vector operation-based calculus, which allows for straightforward implementation of the solution method into contemporary software for numerical mathematical computations. The proposed solution method consists (i) of discretizing the ODE in the domain of interest, (ii) of integrating it numerically, which involves introduction of m integration constants, and (iii) of identifying these m integration constants.

3.1 Derivation of the proposed method in index notation

As for presentation in index notation, it is useful to rewrite the ordinary differential equation (1.1) as

$$\sum_{i=0}^{m} a^{[i]}(x)y^{(i)}(x)=r(x), \quad y^{(i)}(x)\overset{\text{def}}{=}\frac{d^i y(x)}{dx^i}, \tag{3.1}$$

such that superscripts without brackets represent exponents, superscripts in round brackets indicate derivatives, and superscripts in squared brackets stand simply for indices.

Assuming that both the perturbation function $r(x)$ and the coefficient functions $a^{[i]}(x)$, $i=0,1,2,\cdots,m$, are smooth functions in the integration interval $x_a\le x\le x_b$, we subdivide the latter into $(n-1)$ intervals of identical size λ, see (2.1). The governing differential equation (3.1) is evaluated at all n interval boundaries $x_a=x_1$, x_2, x_3, $\cdots$, x_{n-1}, and $x_n=x_b$, delivering n equations of the form

$$\sum_{i=0}^{m} a_k^{[i]}y_k^{(i)}=r_k, \quad k=1,2,\cdots,n, \tag{3.2}$$

where

$$a_k^{[i]}\overset{\text{def}}{=}a^{[i]}(x_k), \quad y_k^{(i)}\overset{\text{def}}{=}\frac{d^i y(x)}{dx^i}\Big|_{x=x_k}, \quad r_k\overset{\text{def}}{=}r(x_k). \tag{3.3}$$

Eqs. (3.2) contain $(m+1)\times n$ unknowns $y_k^{(i)}$ with $i=0,1,2,\cdots,m$ and $k=1,2,\cdots,n$.

In order to reduce the number of unknowns in the discretized version of the ODE (3.2), we start with formal integration of the unknown function $y^{(m)}(x)$ representing the highest-order derivative of the sought function $y(x)$. Integrating $y^{(m)}(x)$ first symbolically from $x=x_1(=x_a)$ to $x=x_k$, applying then also the integration method (2.6) to this problem, and setting the two resulting expressions equal to each other yields

$$\int_{x_a}^{x_k} y^{(m)}(x)dx=y^{(m-1)}(x)\Big|_{x_a}^{x_k}=y_k^{(m-1)}-y_a^{(m-1)}=\sum_{s=1}^{k}A_{ks}^1 y_s^{(m)}. \tag{3.4}$$

Rearranging terms allows for expressing $y_k^{(m-1)}$ as

$$y_k^{(m-1)}=y_a^{(m-1)}+\sum_{s=1}^{k}A_{ks}^1 y_s^{(m)}, \tag{3.5}$$

where $y_a^{(m-1)}$ may be interpreted as an integration constant which is equal to the function value of $y^{(m-1)}(x)$ at the left boundary of the integration interval, i.e., at $x=x_a$. Double and triple integration of $y^{(m)}(x)$ from $x=x_a$ to $x=x_k$ yield by analogy to (3.4) and (3.5):

$$y_k^{(m-2)}=y_a^{(m-2)}+y_a^{(m-1)}(x_k-x_a)+\sum_{s=1}^{k}A_{ks}^2 y_s^{(m)}, \tag{3.6a}$$

$$y_k^{(m-3)}=y_a^{(m-3)}+y_a^{(m-2)}(x_k-x_a)+y_a^{(m-1)}\frac{(x_k-x_a)^2}{2!}+\sum_{s=1}^{k}A_{ks}^3 y_s^{(m)}. \tag{3.6b}$$

Eqs. (3.5) and (3.6) imply that function values of all lower-order derivatives can be expressed as

$$y_k^{(m-i)} = \sum_{j=0}^{i-1} x_k^{[j]} y_a^{(m-i+j)} + \sum_{s=1}^{k} A_{ks}^i y_s^{(m)}, \quad i=1,2,\cdots,m, \quad k=1,2,\cdots,n, \tag{3.7}$$

where

$$x_k^{[j]} \overset{\text{def}}{=} \frac{(x_k - x_a)^j}{j!}. \tag{3.8}$$

Insertion of (3.7) into the discretized version of the ODE (3.2) results in the sought reduction of unknowns. Rearranging terms in the obtained expression can be shown to deliver the following system of n linear, algebraic equations with n unknown function values $y_s^{(m)}$ $(s=1,2,\cdots,n)$ and m integration constants $y_a^{(m-i)}$ $(i=1,2\cdots,m)$:

$$\sum_{s=0}^{k} D_{ks} y_s^{(m)} + \sum_{i=0}^{m-1} d_k^{[i]} y_a^{(i)} = r_k, \quad k=1,2,\cdots,n. \tag{3.9}$$

In (3.9), D_{ks} and $d_k^{[i]}$, respectively, denote the a-priori computable coefficients of one $n \times n$ matrix D and m different vectors $\vec{d}^{[i]}$, with $i=0,1,\cdots,m-1$, reading as:

$$D_{ks} = \sum_{i=0}^{m} a_k^{[m-i]} A_{ks}^i \quad \text{and} \quad d_k^{[i]} = \sum_{j=0}^{i} a_k^{[j]} x_k^{[i-j]}. \tag{3.10}$$

Definitions (3.10) and the quadratic approximation polynomial-related Eqs. (2.5) and (2.3) imply D_{ks} to be a lower triangular matrix, except for coefficient D_{23} which is also not equal to zero, unless $a_2^{[i]} = 0$ for all $i=0,1,\cdots,m$. If integration is based on piecewise exact integration of cubic approximation polynomials, see (2.15), then D_{ks} is a lower triangular matrix, except for coefficients D_{23} and D_{24}, which are both not equal to zero.

Solving the discretized version of the ODE (3.9) for the unknown function values $y_k^{(m)}$, with $k=1,2,\cdots,n$ involves inversion of the D matrix, i.e., D has to have full rank. This requires, in particular, that the first coefficient of the D matrix, D_{11}, is not equal to zero. Definitions (3.10) and (2.5) together with (2.3) imply that $D_{11} \neq 0$ corresponds to the condition $a_1^{[m]} \neq 0$, i.e., $a^{[m]}(x_a) \neq 0$. Restricting our considerations to cases where this condition is satisfied, the solution of (3.9) for the unknown function values $y_k^{(m)}$ reads as

$$y_k^{(m)} = w_k + \sum_{i=1}^{m} v_k^{[m-i]} y_a^{(m-i)}, \quad k=1,2,\cdots,n, \tag{3.11}$$

where the vector $\vec{w}$ and the m different vectors $\vec{v}^{[m-i]}$ with $i=1,2,\cdots,m$ contain again a-priori computable coefficients defined as

$$w_k = \sum_{s=1}^{k} D_{ks}^{-1} r_s \quad \text{and} \quad v_k^{[m-i]} = -\sum_{s=1}^{k} D_{ks}^{-1} d_s^{[m-i]}. \tag{3.12}$$

(3.11) provides access to function values of the sought function $y^{(m)}(x)$. Back-substitution of (3.11) into (3.7) results in expressions for the function values of the lower-order derivatives $y^{(m-i)}(x)$, i.e., expressions for $y_k^{(m-i)}$ with $i=1,2,\cdots,m$. We are left with determination of the m integration constants $y_a^{(m-i)}$ with $i=1,2,\cdots,m$. This is described next.

The m integration constants $y_a^{(m-i)}$ with $i=1,2,\cdots,m$ are identified from m boundary conditions. Considering *initial value problems*, all m boundary conditions are formulated in the sought integration constants, providing direct access to them. Considering general *boundary value problems*, however, the boundary conditions refer to both boundaries of the interval of interest. The conditions referring to x_a provide direct access to sought integration constants, and the remaining constants have to be identified from the boundary conditions referring to x_b. To come up with a related system of algebraic equations, we specify (3.7) first for (3.11) and then for $k=n$, which delivers, when considering $x_n=x_b$ as well as $y_n^{(m-i)}=y_b^{(m-i)}$,

$$y_b^{(m-i)}=\sum_{j=0}^{i-1}x_b^{[j]}y_a^{(m-i+j)}+\sum_{s=1}^{n}A_{ns}^{i}\Big[w_s+\sum_{j=1}^{m}v_s^{[m-j]}y_a^{(m-j)}\Big],\quad i=1,2,\cdots,m. \tag{3.13}$$

Eq. (3.13) together with the m boundary conditions represents m equations for the m integration constants $y_a^{(m-i)}$ with $i=1,2,\cdots,m$.

3.2 Matrix-vector operation-based implementation

Herein, we describe the transition from the index notation to the matrix-vector operation-based outline of the proposed solution method. The structogram of the proposed solution method reads as:

1. Specify x_a and x_b, the two boundaries of the integration interval.
2. Specify n, the number of grid points within the integration interval.
3. Compute λ, the grid point distance, as $\lambda=(x_b-x_a)/(n-1)$.
4. Compute coordinates of grid points: x_k $(k=1,2,\cdots,n)$
$$x_k=x_a+(k-1)\lambda. \tag{3.14}$$
5. Compute vectors $\vec{x}^{[i]}$ $(i=0,1,\cdots,m-1)$, with coordinates defined as
$$x_k^{[i]}=\frac{(x_k-x_a)^i}{i!}. \tag{3.15}$$
6. Since Eqs. (3.10) cannot be directly related to standard matrix-vector operations, the coefficient functions $a^{[i]}(x)$ $(i=0,1,\cdots,m)$ are evaluated at the n grid point positions, and the obtained values are put into the main diagonals of corresponding $n\times n$ diagonal matrices $M^{[i]}$ $(i=0,1,\cdots,m)$ such that the components of the latter matrices read as
$$M_{kk}^{[i]}=a_k^{[i]},\quad k=1,2,\cdots,n;\quad \text{all other}\quad M_{ks}^{[i]}=0. \tag{3.16}$$

7. Evaluate perturbation function $r(x)$ at all n grid points, in order to assemble vector $\vec{r}$ with coordinates $r_k = r(x_k)$.

8. Assemble matrices A, A^0, and A^1 according to Eqs. (2.3), (2.4) and (2.5), or according to Eqs. (2.16), (2.17) and (2.15), respectively. Compute matrices A^i $(i=2,3,\cdots,m)$ based on the following matrix multiplications:
$$A^i = A^1 A^{i-1}. \tag{3.17}$$

9. Compute matrix D and vectors $\vec{d}^{[i]}$ $(i=0,1,\cdots,m-1)$, using matrix multiplications and matrix-vector multiplications, respectively, based on the following matrix-vector representations of Eqs. (3.10)
$$D = \sum_{i=0}^{m} M^{[m-i]} A^i \quad \text{and} \quad \vec{d}^{[i]} = \sum_{j=0}^{i} M^{[j]} \vec{x}^{[i-j]}. \tag{3.18}$$

10. Invert matrix D and compute vectors $\vec{w}$ and $\vec{v}^{[i]}$ $(i=0,1,\cdots,m-1)$:
$$\vec{w} = D^{-1}\vec{r}, \quad \vec{v}^{[i]} = -D^{-1}\vec{d}^{[i]}. \tag{3.19}$$

11. *Only for boundary value problems*: Determine unknown integration constants (= state variables at x_a) from boundary conditions formulated at the end of the integration interval, using the following system of algebraic equations:
$$y_b^{(m-i)} = \sum_{j=0}^{i-1} x_b^{[j]} y_a^{(m-i+j)} + (\vec{u}^{[i]})^T \left[\vec{w} + \sum_{j=1}^{m} \vec{v}^{[m-j]} y_a^{(m-j)} \right], \quad i=1,2,\cdots,m, \tag{3.20}$$
where the row vector $(\vec{u}^{[i]})^T$ stands for the transpose of the column vector $\vec{u}^{[i]}$ with coordinates
$$u_k^{[i]} = A_{nk}^i. \tag{3.21}$$

12. Compute solution vectors
$$\vec{y}^{(m)} = \vec{w} + \sum_{i=1}^{m} \vec{v}^{[m-i]} y_a^{(m-i)}, \quad \vec{y}^{(m-i)} = \sum_{j=0}^{i-1} \vec{x}^{[j]} y_a^{(m-i+j)} + A^i \vec{y}^{(m)}. \tag{3.22}$$

4 Comparison of the presented method with other methods

4.1 First-order initial value problem: comparison with backward Euler scheme and Runge-Kutta method

Herein, we solve the initial value problem
$$\frac{dy(x)}{dx} + 15y(x) = 0, \quad y(x=0) = 1 \tag{4.1}$$
by means of (i) the backward Euler method (”bEM”) see Appendix B.1, of (ii) the Runge-Kutta method (”RK4”), see Appendix B.2, and of (iii) the presented solution method

based on piecewise exact integration of quadratic approximation polynomials and on piecewise exact integration of cubic approximation polynomials, respectively, see Section 2. Solutions, computed in the interval $x \in [0,1]$, are compared with the analytical solution of (4.1), reading as

$$y(x) = \exp(-15x). \tag{4.2}$$

Since (4.2) is a strongly decaying exponential function, (4.1) is mathematically classified as a "stiff" initial value problem.

We investigate the accuracy of numerical results as a function of the chosen method and of the number of chosen grid points n, i.e., we repeatedly solve problem (4.1), based on many different choices for grid point number n. In this context, the following prediction error is evaluated at every grid point position and for every numerical solution scheme mentioned above:

$$E_k = |y_k - y(x_k)|, \quad k = 1,2,\cdots,n, \tag{4.3}$$

where y_k stands for numerical results referring to position x_k and where $y(x_k)$ stands for the analytical solution (4.2) evaluated at $x = x_k$. To compare the four solution methods, we focus on the maximum prediction error $E_{\max}$, on the mean prediction error E_{mean}, and on the prediction error at the end of the integration interval E_b, defined as

$$E_{\max} = \max_k E_k, \quad E_{mean} = \frac{1}{n}\sum_{k=1}^{n} E_k, \quad E_b = E_n, \tag{4.4}$$

respectively.

Properties of the proposed method observed with small discretization densities ($n \leq 30$) are discussed next. Numerical results at the end of the integration interval, obtained with increasing grid point number n, "oscillate" with decreasing amplitude around the analytical solution, see Fig. 4.

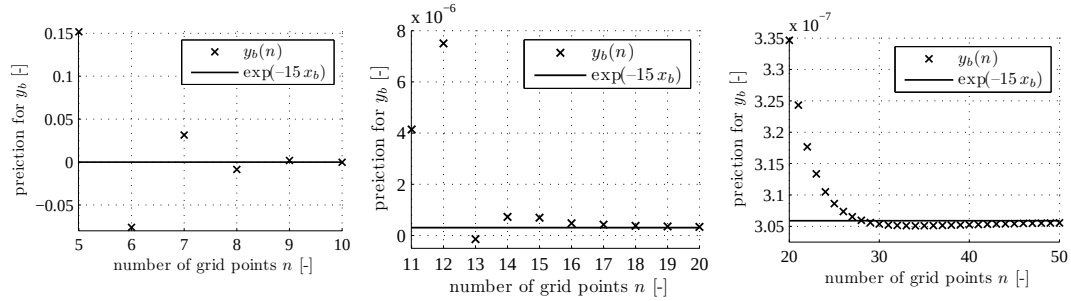


Figure 4: Convergence study for stiff initial value problem (4.1): function value at the end of the integration interval ($x_b = 1$) obtained with the presented solution method based on piecewise exact integration of cubic approximation polynomials (crosses) and exact solution (solid line), see (4.2); notably, the ordinates of (a), (b), and (c) refer to different intervals.

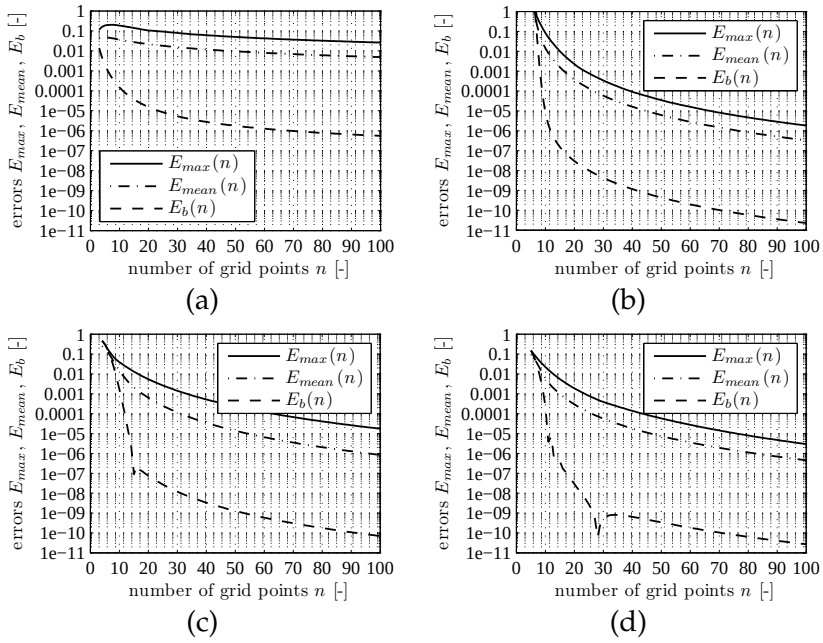


Figure 5: Convergence study for stiff initial value problem (4.1): errors (4.4) as a function of the grid point number n: (a) backward Euler method ("bEM") see B.1, (b) classical Runge-Kutta method ("RK4"), see B.2, (c) presented solution method based on piecewise exact integration of quadratic approximation polynomials, and (d) presented solution method based on piecewise exact integration of cubic approximation polynomials.

This results in non-smooth shapes of $E_b(n)$, see, e.g., the kink at $n=28$ in Fig. 5(d), resulting from the fact that numerical results at the end of the integration interval are (i) larger than the analytical solution for $14 \leq n \leq 28$, but (ii) smaller than the analytical solution for $n \leq 29$, see Fig. 4.

The proposed method performs significantly better than the backward Euler method ("bEM"), independent of the chosen grid point number, see Fig. 5. For grid point numbers around 10, the proposed scheme using quadratic approximation polynomials performs similar to the Runge-Kutta method ("RK4"), while the proposed scheme using cubic approximation polynomials performs even better than the classical RK4 method. For grid point numbers around 100, the proposed scheme using cubic approximation polynomials performs similar to the Runge-Kutta method, while the proposed scheme using quadratic approximation polynomial results in smaller accuracies.

4.2 Second-order boundary value problem: comparison with finite difference method

Herein, we solve the benchmark boundary value problem of [19]

$$\frac{d^2y(x)}{dx^2} - \frac{dy(x)}{dx} = -\exp(x-1) - 1, \quad \begin{cases} y(x=0)=0, \\ y(x=1)=0, \end{cases} \tag{4.5}$$

by means of (i) the finite difference method, see Appendix B.3 and of (ii) the presented solution method based on piecewise exact integration of quadratic approximation polynomials and on piecewise exact integration of cubic approximation polynomials, respectively, see Section 2. Solutions, computed in the interval $x \in [0,1]$, are compared with the analytical solution of (4.5), reading as

$$y(x) = x\big[1-\exp(x-1)\big]. \tag{4.6}$$

When solving the problem at hand according to the outline of the presented solution method described in Subsection 3.2, two integration constants are introduced: $y_a^{(0)}$ and $y_a^{(1)}$. This provides the motivation to reformulate the boundary conditions given in (4.5) using the notation of Subsection 3.2; this yields

$$y_a^{(0)} = 0, \quad y_b^{(0)} = 0. \tag{4.7}$$

While the first boundary condition (4.7) provides direct access to the first integration constant, the second integration constant $y_a^{(1)}$ is to be identified from the boundary condition at the end of the integration interval, see the second condition (4.7). We are left with establishing a relation between the boundary value $y_b^{(0)}$ appearing in (4.7), and the integration constant $y_a^{(1)}$. To this end, we focus on point 11 of the listing in Subsection 3.2, where relations between boundary values of $y(x)$ and its derivatives with respect to x are established. Evaluating $y_b^{(0)}$ according to (3.20) yields in the present context:

$$y_b^{(0)} = (\vec{u}^{[2]})^T \vec{w} + \Big[x_b^{[0]} + (\vec{u}^{[2]})^T \vec{v}^{[0]}\Big] y_a^{(0)} + \Big[x_b^{[1]} + (\vec{u}^{[2]})^T \vec{v}^{[1]}\Big] y_a^{(1)}. \tag{4.8}$$

$y_a^{(1)}$ is the only unknown on the right-hand side of Eq. (4.8), i.e., all other quantities can be computed after having completed the first 10 points of the listing of Subsection 3.2. We now specify Eq. (4.8) for the first equation (4.7), and we insert the resulting expression for $y_b^{(0)}$ into the second boundary condition (4.7) which results in an algebraic equation for the sought integration constant $y_a^{(1)}$, with the solution:

$$y_a^{(1)} = -\frac{(\vec{u}^{[2]})^T \vec{w}}{x_b^{[1]} + (\vec{u}^{[2]})^T \vec{v}^{[1]}}. \tag{4.9}$$

Now, all integration constants are identified (see the first equation (4.7) and (4.9)), such that the further evaluation of the solution is the same as the one described for initial value problems, see point 12 of the listing of Subsection 3.2, providing access to $y(x)$ and its derivatives with respect to x.

We investigate the accuracy of numerical results as a function of the chosen method and the number of chosen grid points n, i.e., we repeatedly solve problem (4.1), based on many different choices for grid point number n. In this context, prediction error (4.3)

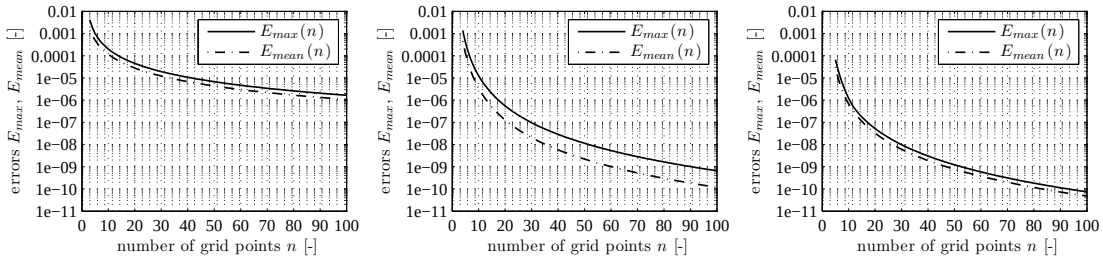


Figure 6: Convergence study for boundary value problem (4.5): errors (4.4) as a function of the grid point number n: (a) finite difference method, see Appendix B.3, (b) presented solution method based on piecewise exact integration of quadratic approximation polynomials, and (c) presented solution method based on piecewise exact integration of cubic approximation polynomials.

is evaluated at every grid point position and for every numerical solution scheme. To compare the three solution methods, we focus on the maximum prediction error $E_{\max}$ and on the mean prediction error E_{mean} defined in (4.4). The proposed methods performs better than the finite difference method, see Fig. 6. For a fixed grid point number, cubic approximation polynomials provide more accurate results than quadratic approximation polynomials.

5 Oscillation of a modulated torsional spring pendulum

5.1 Statement of the problem

In order to check the performance of the proposed method in the framework of a challenging initial value problem, we consider a linear mechanical system comprising a massless, straight, and rigid rod which is supported in its middle such that the rod can rotate around an axis which is normal to the rod axis, see Fig. 7.

The rod is fixed to one end of a linear torsional spring (with spring stiffness $=\bar{c}$), and the other end of the spring is fixed in space. Two identical point masses M_p are mounted to the rod in an axisymmetric fashion with respect to the axis of rotation. They are forced to move simultaneously along the rod in opposite directions such that at any time t the distance of both masses from the center of rotation is the same and given through the following mathematical expression

$$\ell(t)=\ell_0(1+m_d\sin(\omega t)). \tag{5.1}$$

In Eq. (5.1), ℓ_0, m_d, and ω, respectively, denote the mean distance of the point masses from the center of rotation, the modulation depth [20], and the angular frequency of the modulation. As the masses move closer to the center of rotation, the moment of inertia of the system, J, decreases, while its eigenfrequency increases, and vice versa. In mathematical terms, the moment of inertia reads at time t as [20]

$$J(t)=2M_p\ell^2(t). \tag{5.2}$$

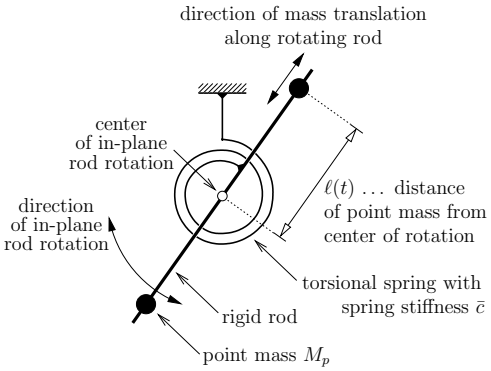


Figure 7: Modulated torsional spring pendulum analyzed by Butikov [20].

Considering a friction-free system, and formulating the dynamic torque equilibrium for the oscillating pendulum, delivers the following governing differential equation:

$$\frac{d}{dt}\left[J(t)\dot{\varphi}(t)\right] = -\bar{c}\varphi(t). \tag{5.3}$$

In Eq. (5.3), $J(t)\dot{\varphi}(t)$ denotes the angular momentum of the rotating structure, and the right-hand side of (5.3) implies that the restoring torque acting from the spring on the rod is proportional to the angular displacement φ of the rod, where φ is measured relative to the natural (stress-free) position of the spring. Following Butikov [20] who considered that the oscillation is damped proportionally to the angular velocity of the rod, i.e., adding $-4M_p\ell_0^2\gamma\dot{\varphi}$ to the right-hand side of Eq. (5.3), defining $\omega_0 \overset{\text{def}}{=} \sqrt{\bar{c}/(2M_p\ell_0^2)}$, combining Eqs. (5.1) to (5.3), and simplifying yields the following second-order ODE for $\varphi(t)$

$$(1+m_d\sin(\omega t))\ddot{\varphi}(t) + \left[2m_d\omega\cos(\omega t)(1+m_d\sin(\omega t))+2\gamma\right]\dot{\varphi}(t)+\omega_0^2\varphi(t)=0, \tag{5.4}$$

see Fig. 8 for an illustration of the involved non-linear coefficient functions.

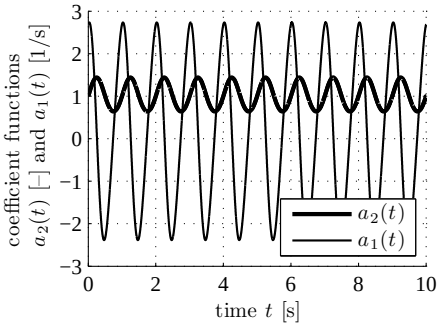


Figure 8: Coefficient functions $a_2(t)=1+m_d\sin(\omega t)$ and $a_1(t)=2m_d\omega\cos(\omega t)(1+m_d\sin(\omega t))+2\gamma$, evaluated for the input values listed in (5.5).

5.2 Solution by means of the presented method

Because no analytical solution exists for the linear and homogeneous ODE with variable coefficients, (5.4), we solve it numerically within the time interval ranging from $t=0$s to $t=10$s, considering the following numerical inputs

$$m_d=0.2, \quad \omega=2\pi[\mathrm{s}^{-1}], \quad \gamma=\frac{\pi}{36}[\mathrm{s}^{-1}], \quad \omega_0=\pi[\mathrm{s}^{-2}], \tag{5.5}$$

and the initial conditions

$$\varphi(0)=10^\circ, \quad \dot{\varphi}(0)=0. \tag{5.6}$$

This problem was analyzed by Butikov [20, pp. 538]. We note for later reference that the size of the integration interval $t_b-t_a=10$ s together with the angular frequency of the modulation $\omega=2\pi$ (corresponding to an modulation period $T=1$s) imply that the time interval of interest comprises ten sinusoidal modulation cycles. The numerical solution, obtained with a grid point number $n=2001$ (2000 intervals) and with the integration method based on piecewise exact integration of quadratic approximation polynomials, satisfactorily reproduces the benchmark solution described in [20], i.e., the obtained functions $\varphi(t)$, $\dot{\varphi}(t)$, and $\ddot{\varphi}(t)$ exhibit pronounced nonlinearities, see Fig. 9.

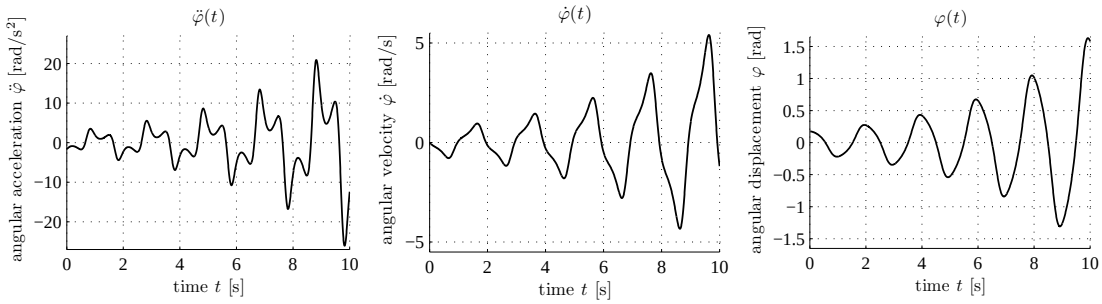


Figure 9: Solution for ODE (5.4) specified for input values (5.5), with initial conditions (5.6), computed based on the proposed solution method with grid point number $n=2001$ and piecewise exact integration of quadratic approximation polynomials, see Section 2; for details on numerical results see also Table 1.

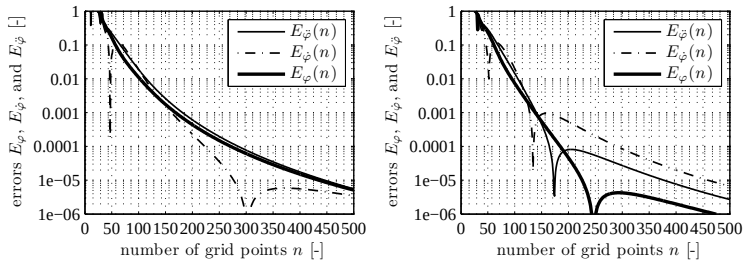
In order to gain more insight into the characteristics of the proposed solution scheme, we investigate the accuracy of our results as a function of the number of chosen grid points n, i.e., we repeatedly solve the ODE (5.4), based on many different choices for grid point number n. In the context of defining meaningful prediction errors for the computed state variables $\varphi(t)$, $\dot{\varphi}(t)$, and $\ddot{\varphi}(t)$, we recall that no analytical solution exists for the problem at hand. Hence, we treat the results obtained with $n=2001$ (see Fig. 9) as the reference solution, and we compare the computed state variables $\varphi(t)$, $\dot{\varphi}(t)$, and $\ddot{\varphi}(t)$ at the *end* of the integration interval ($t=10$s) with the corresponding values of the reference solution, based on the following definitions for the prediction errors:

$$E_\alpha=\left|\frac{\alpha(t=10\mathrm{s};n)-\alpha(t=10\mathrm{s};n=2001)}{\alpha(t=10\mathrm{s};n=2001)}\right|, \quad \alpha=\varphi,\dot{\varphi},\ddot{\varphi}. \tag{5.7}$$

Table 1: Numerical results for pendulum problem obtained with grid point number $n=2001$ and piecewise exact integration of quadratic approximation polynomials, see also Fig. 9.

t [s]	$\varphi(t)$ [rad]	$\dot{\varphi}(t)$ [rad/s]	$\ddot{\varphi}(t)$ [rad/s^2]
0	+0.17453293	±0.00000000	−1.7225709
1	−0.21779263	+0.07666806	+1.9434582
2	+0.27161429	−0.14664899	−2.2865614
3	−0.33862976	+0.21682313	+2.7593630
4	+0.42210886	−0.29292147	−3.3787311
5	−0.52611988	+0.38019098	+4.1707151
6	+0.65572848	−0.48390759	−5.1711305
7	−0.81724486	+0.60980455	+6.4268465
8	+1.01853127	−0.76446596	−7.9977637
9	−1.26938494	+0.95572338	+9.9595272
10	+1.58201503	−1.19308784	−12.4070726

Errors (5.7) are evaluated for both integration methods: quadratic and cubic approximation polynomials. Numerical results at the end of the integration interval, obtained with increasing grid point number n, "oscillate" with decreasing amplitude around the reference solution, see the kinks in Figs. 10 (a) and (b). In addition, the convergence study implies that both approaches have a similar quantitative performance in the range of engineering relevance, compare Figs. 10 (a) and (b).



	(a)	(b)
errors $< 10^{-1}$	$n \geq 70$	$n \geq 74$
errors $< 10^{-2}$	$n \geq 115$	$n \geq 109$
errors $< 10^{-3}$	$n \geq 182$	$n \geq 141$
errors $< 10^{-4}$	$n \geq 286$	$n \geq 291$
errors $< 10^{-5}$	$n \geq 447$	$n \geq 460$

Figure 10: Convergence study for pendulum problem: errors (5.7) as a function of the number of used grid points n: (a) for piecewise exact integration of quadratic approximation polynomials, and (b) for piecewise exact integration of cubic approximation polynomials.

For errors smaller than 1%, both approaches require practically 120 grid points. While 141 grid points allow for reducing prediction errors down to 0.1%, if integration is based on piecewise cubic approximation polynomials, the same accuracy requires 182 grid points, if quadratic approximation polynomials are used (Fig. 10). For even more accurate results in terms of prediction errors amounting to less than 0.01%, however, quadratic approximation polynomials turn out to perform a little more efficient that their cubic counterparts.

5.3 Multi-subinterval solution by means of the presented method

Inversion of the D-matrix defined in (3.18) is – for large values of n – the most time-consuming part of the solution scheme, since the time required for numerical inversion of D increases overlinearly with increasing grid point number n. Reduction of computation time requires reduction of the grid point number. In order to maintain a high accuracy level, it is beneficial to subdivide the integration domain into several subintervals, and to apply the proposed solution method each of these subintervals separately. To obtain results with errors as small as the ones given for $n=401$ in Fig. 10, for instance, one could use 10 integration subintervals covering time periods of one second each (one subinterval per modulation cycle), and discretize each subinterval by 41 grid points only. After having applied the presented solution method to the first subinterval $t\in[0\text{s},1\text{s}]$, the state variables obtained at the end of this first subinterval, i.e., $y_b^{(0)}=\varphi(t=1\text{s})$ and $y_b^{(1)}=\dot{\varphi}(t=1\text{s})$ would be available and serve as initial values for solving the problem in the second subinterval $t\in[1\text{s},2\text{s}]$, and so on. This way, Fig. 10 implies that only 30 grid points per modulation cycle are sufficient to reduce prediction errors to values smaller than 0.01%.

6 Steady-state heat transfer through a cooling web

6.1 Statement of the problem

In order to describe the performance of the proposed method in the framework of an engineering second-order boundary value problem involving boundary conditions which are more complex that the ones given in Subsection 4.2, we study stationary heat transfer through a cooling web with width t, length ℓ, and a rectangular cross-section, see Fig. 11.

Since the surfaces at the top and the bottom are only slightly inclined with respect to the mid-plane of the cooling web ($\tan\beta\ll 1$, see Fig. 11), we follow [21] and assume a one-dimensional heat conduction problem in x-direction, governed by

$$q(x)=-\eta\frac{dT(x)}{dx}, \tag{6.1}$$

where $q(x)$ is the heat flux in x-direction per unit cross-sectional area of the cooling web, η is the thermal conductivity of the cooling web material, and $T(x)$ is the temperature distribution.

In order to derive the governing differential equation for $T(x)$, we deal with the energy balance of an elementary cooling web volume (see Fig. 11(b)). Since there are no heat sources or heat sinks within a cooling web, and because we envision a stationary (time-independent) problem, conservation of energy implies simply that the heat conducted into the elementary volume is equal to the heat conducted out of the element. The heat entering the element through the cross-section at x (see Fig. 11(a)) follows from

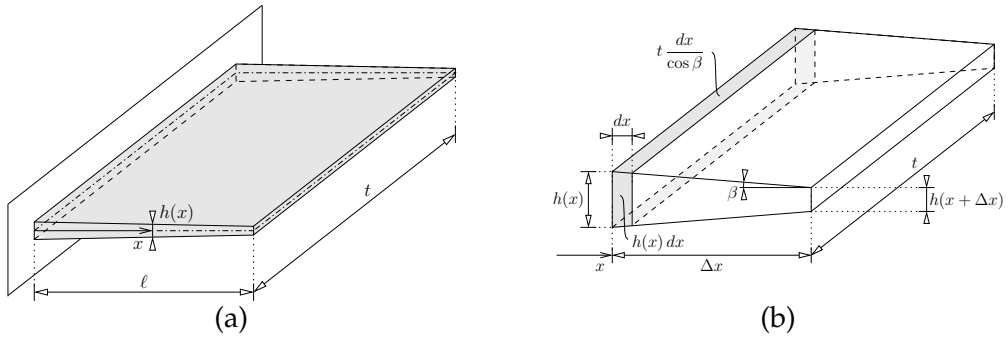


Figure 11: Cooling web typically used in electronic devices [21] (a) geometric dimensions, (b) elementary volume.

multiplying (6.1) with the cross-sectional area $A(x)=th(x)$ as

$$Q(x)=-\eta\frac{dT(x)}{dx}\Big|_x th(x). \tag{6.2}$$

By analogy, the heat leaving the element through the cross-section at $x+\Delta x$ (see Fig. 11(b)) follows as

$$Q(x+\Delta x)=-\eta\frac{dT(x)}{dx}\Big|_{x+\Delta x} th(x+\Delta x). \tag{6.3}$$

The heat loss through the four lateral (ℓat) surfaces of the element, $Q_{\ell at}$, is governed by the heat transfer coefficient α_h, the temperature difference between these surfaces and the surrounding air, and the area of the solid-air interface. Integrating over the latter (see Fig. 11(b)), we find

$$Q_{\ell at}=\int_x^{x+\Delta x}\alpha_h\Big[T(\xi)-T_{air}\Big]\,2\Big[\frac{t}{\cos\beta}+h(\xi)\Big]d\xi. \tag{6.4}$$

Formulating, based on (6.2), (6.3), and (6.4), the elementary energy balance condition as $Q(x)=Q(x+\Delta x)+Q_{\ell at}$, rearranging terms, dividing the resulting expression by $\Delta x=x_j-x_i$, and taking the limit $\Delta x\to 0$, delivers[†]

$$\eta t\frac{d}{dx}\Big[\frac{dT(x)}{dx}h(x)\Big]-\alpha_h\Big[T(x)-T_{air}\Big]2\Big[\frac{t}{\cos\beta}+h(x)\Big]=0. \tag{6.5}$$

Rearranging terms in (6.5) finally yields the following second-order ODE for the temperature distribution $T(x)$:

$$\eta th(x)\frac{d^2T(x)}{dx^2}+\eta t\frac{dh(x)}{dx}\frac{dT(x)}{dx}-2\alpha_h\Big[\frac{t}{\cos\beta}+h(x)\Big]T(x)=-2\alpha_h\Big[\frac{t}{\cos\beta}+h(x)\Big]T_{air}. \tag{6.6}$$

[†]Use is made of $\lim_{\Delta x\to 0}\frac{1}{\Delta x}\int_x^{x+\Delta x}f(\xi)d\xi=f(x)$.

As boundary conditions, a constant temperature is prescribed at the position $x=0$, where the cooling web is fixed to the medium which is to be cooled [21]

$$T(x=0)=T_a=\text{const} \tag{6.7}$$

and at the other end $(x=\ell)$, a free-surface condition applies

$$-\eta\frac{dT(x)}{dx}\Big|_{x=\ell} th(x=\ell)=\alpha_h\Big[T(x=\ell)-T_{air}\Big]th(x=\ell). \tag{6.8}$$

Eq. (6.8) is an energy balance condition formulated for the rectangular surface at $x=\ell$, expressing that the heat which is transported inside the cooling web to this surface is equal to the energy lost from that surface towards the ambient air.

6.2 Solution by means of the presented method

The governing differential equation (6.6), together with the boundary conditions (6.7) and (6.8), is ready to be solved by the presented solution method, provided that function $h(x)$ and constants t, ℓ, α_h, η, T_a, and T_{air} are specified numerically. The height of the cooling web $h(x)$ is considered to decrease linearly with increasing distance from the medium which is to be cooled (see Fig. 11(a)), and geometric dimensions typical for cooling webs in electronic equipment are adopted [21]:

$$h(x)=h_a+(h_b-h_a)\frac{x}{\ell} \quad \text{with} \quad \begin{cases} h_a=2\text{mm}, \\ h_b=1\text{mm}, \end{cases} \quad \text{and} \quad \begin{cases} \ell=40\text{mm}, \\ t=200\text{mm}. \end{cases} \tag{6.9}$$

Furthermore, we consider the cooling web to consist of aluminum, the surrounding air to exhibit room temperature, and the body to be cooled to exhibit 50°C, i.e.,

$$\begin{cases} \eta=200\text{W m}^{-1\circ}\text{C}^{-1}, \\ \alpha_h=15\text{W m}^{-2\circ}\text{C}^{-1}, \end{cases} \quad \begin{cases} T_a=50^\circ\text{C}, \\ T_{air}=25^\circ\text{C}. \end{cases} \tag{6.10}$$

When solving the problem at hand according to the outline of the presented solution method described in Subsection 3.2, the temperature distribution $T(x)$ is denoted as $y(x)$, and two integration constants are introduced: $y_a^{(0)}$ and $y_a^{(1)}$. This provides the motivation to reformulate the boundary conditions (6.7) and (6.8) using the notation of Subsection 3.2; this yields

$$y_a^{(0)}=T_a, \quad -\eta y_b^{(1)}=\alpha_h\Big[y_b^{(0)}-T_{air}\Big]. \tag{6.11}$$

While the first boundary condition (6.11) together with numerical input (6.10) provides direct access to the first integration constant as $y_a^{(0)}=50^\circ$C, the second integration constant $y_a^{(1)}$ is to be identified from the boundary condition at the end of the cooling web, see the second condition (6.11). We are left with establishing a relation between the boundary

values $y_b^{(1)}$ and $y_b^{(0)}$ appearing in (6.11), and the integration constant $y_a^{(1)}$. To this end, we focus on point 11 of the listing in Subsection 3.2, where relations between boundary values of $y(x)$ and its derivatives with respect to x are established. Evaluating $y_b^{(0)}$ and $y_b^{(1)}$ according to (3.20) yields in the present context:

$$y_b^{(0)} = (\vec{u}^{[2]})^T\vec{w} + \left[x_b^{[0]} + (\vec{u}^{[2]})^T\vec{v}^{[0]}\right] y_a^{(0)} + \left[x_b^{[1]} + (\vec{u}^{[2]})^T\vec{v}^{[1]}\right] y_a^{(1)}, \tag{6.12a}$$

$$y_b^{(1)} = (\vec{u}^{[1]})^T\vec{w} + (\vec{u}^{[1]})^T\vec{v}^{[0]} y_a^{(0)} + \left[x_b^{[0]} + (\vec{u}^{[1]})^T\vec{v}^{[1]}\right] y_a^{(1)}. \tag{6.12b}$$

$y_a^{(1)}$ is the only unknown on the right-hand sides of Eqs. (6.12), i.e., all other quantities are either specified by means of boundary conditions (see $(6.11)_1$) or they can be computed after having completed the first 10 points of the listing of Subsection 3.2. We now insert Eqs. (6.12) into the second boundary condition (6.11) which results in an algebraic equation for the sought integration constant $y_a^{(1)}$, with the solution:

$$y_a^{(1)} = \frac{\alpha_h\left\{T_{air} - (\vec{u}^{[2]})^T\vec{w} - \left[x_b^{[0]} + (\vec{u}^{[2]})^T\vec{v}^{[0]}\right] T_a\right\} - \eta\left[(\vec{u}^{[1]})^T\vec{w} + (\vec{u}^{[1]})^T\vec{v}^{[0]} T_a\right]}{\alpha_h\left[x_b^{[1]} + (\vec{u}^{[2]})^T\vec{v}^{[1]}\right] + \eta\left[x_b^{[0]} + (\vec{u}^{[1]})^T\vec{v}^{[1]}\right]}. \tag{6.13}$$

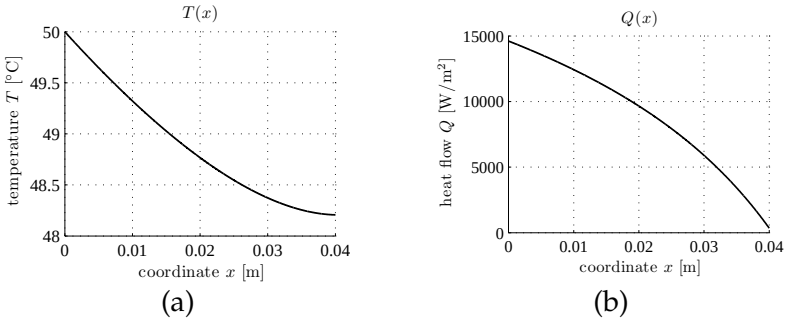
Now, all integration constants are identified (see the first equation (6.11) and (6.13)), such that the further evaluation of the solution is the same as the one described for initial value problems, see point 12 of the listing of Subsection 3.2, providing access to $y(x)$ and its derivatives with respect to x, i.e., to the temperature field $T(x)$ and its derivatives with respect to x.

In order to investigate the accuracy of our results as a function of the number of chosen grid points n, we repeatedly solve the ODE (6.6), based on many different choices for grid point number n. In the context of defining meaningful prediction errors, we set our focus on boundary values which were not specified through boundary conditions, i.e., on the heat fluxes at the integration boundaries Q_a and Q_b, as well as on the temperature at the free end of the cooling web, T_b. We treat the results obtained with $n = 2001$ (see Fig. 12) as the reference solution, and we compare the computed state variables Q_a, Q_b, and T_b with the corresponding values of the reference solution, based on the following definitions for the prediction errors:

$$E_\alpha = \left|\frac{\alpha(n) - \alpha(n=2001)}{\alpha(n=2001)}\right|, \quad \alpha = Q_a, Q_b, T_b. \tag{6.14}$$

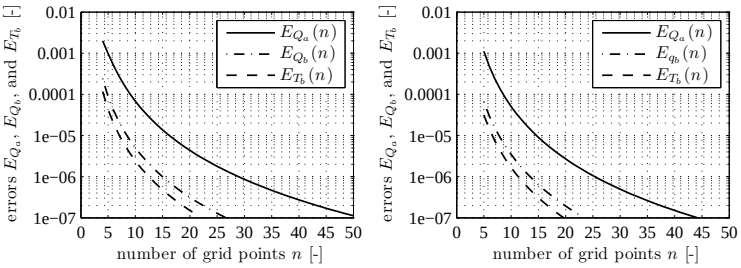
Errors (6.14) are evaluated for both integration methods: quadratic and cubic approximation polynomials (see Fig. 13).

The convergence study implies that piecewise exact integration of cubic approximation polynomials based on a specific grid point number results in a slightly higher precisions than piecewise exact integration of quadratic approximation polynomials and the



(a) (b)

Figure 12: Solution for the cooling web problem (6.6) specified for input values (6.9) and (6.10), with boundary conditions (6.7) and (6.8), computed based on the proposed solution method with grid point number $n=2001$ and piecewise exact integration of quadratic approximation polynomials, see Section 2: (a) solution function $T(x)$, (b) heat flow distribution $Q(x)$ according to (6.1); for details on numerical results see also Table 2.



(a) (b)

	(a)	(b)
errors $< 10^{-3}$	$n \geq 5$	$n \geq 6$
errors $< 10^{-4}$	$n \geq 10$	$n \geq 9$
errors $< 10^{-5}$	$n \geq 17$	$n \geq 15$
errors $< 10^{-6}$	$n \geq 29$	$n \geq 26$

Figure 13: Convergence study for cooling web problem: errors (6.14) as a function of the number of used grid points n: (a) for piecewise exact integration of quadratic approximation polynomials, and (b) for piecewise exact integration of cubic approximation polynomials.

same grid point number. With the former integration method, 5, 10, and 20 grid point deliver results with errors (6.14) smaller than 1.12×10^{-3}, 5.08×10^{-5}, and 2.70×10^{-6}, respectively.

Table 2: Numerical results for cooling web problem obtained with grid point number $n=2001$ and piecewise exact integration of quadratic approximation polynomials, see also Fig. 12.

x [mm]	$T(x)$ [°C]	$Q(x)$ [W/m^2]
0	50.000000	14608.945
4	49.715876	13792.518
8	49.448774	12904.759
12	49.200253	11931.795
16	48.972186	10856.156
20	48.766839	9655.5509
24	48.586987	8301.1055
28	48.436071	6754.7769
32	48.318412	4965.4568
36	48.239535	2862.9208
40	48.206634	348.09951

7 Structural analysis of a slender tower

7.1 Statement of the problem

Next, the presented solution method is applied to a more complex boundary value problem associated with a fourth-order ODE, namely the determination of the internal forces and of the displacements of a slender tower made up of reinforced concrete. The tower exhibits a height of ℓ=150 m and an annular cross-section (see Fig. 14), and it is loaded, at the top, by two forces acting in axial and transversal direction, as well as by a distributed transversal loading $q(x)$ along the entire height, and by dead and life load acting in axial direction (not illustrated in Fig. 14 since force vectors would coincide with the tower axis).

The structural analysis is carried out within the framework of second-order beam theory, by solving the following set of differential equations: two equilibrium conditions

$$\frac{dR(x)}{dx}=-q(x), \qquad \frac{dM(x)}{dx}=R(x)-N(x)\Big[\frac{dy_{ini}(x)}{dx}+\frac{dy(x)}{dx}\Big], \tag{7.1}$$

as well as the following three equations, comprising one constitutive and the two geometric relations

$$\kappa(x)=\frac{M(x)}{EI(x)}, \quad \frac{d\varphi(x)}{dx}=-\kappa(x), \quad \frac{dy(x)}{dx}=\varphi(x). \tag{7.2}$$

In Eqs. (7.1) and (7.2), x denotes the axial coordinate, M stands for the bending moment, R for the transversal force, N for the axial force, y_{ini} for the (initial) deflection in the stress-free configuration, y for the (additional) force-induced deflection, q the transversal loading per unit length; κ is the curvature of the tower's axis, EI the bending stiffness, and φ the force-induced angel of cross-sectional rotation. The bending stiffness is a cubic

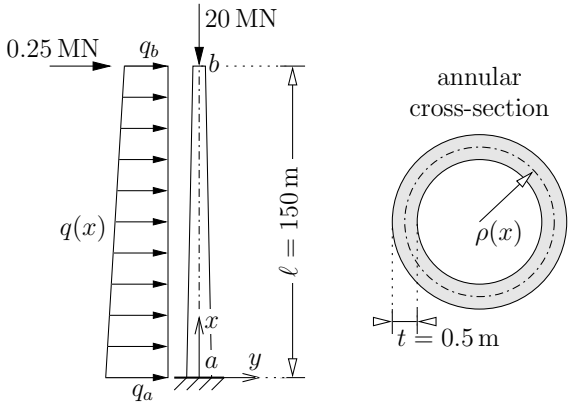


Figure 14: Benchmark problem for structural analysis within the framework of second-order beam theory: slender tower with annular cross-section analyzed by Rubin [22].

function of the tower's radius $\rho(x)$ reading as [22]:

$$EI(x)=E\pi\rho(x)t\Big[\rho^2(x)+\frac{t^2}{4}\Big], \tag{7.3}$$

where E and t denote Young's modulus of the tower material and the wall-thickness of the tower (see Fig. 14), respectively. At the clamped bottom of the tower, the two force-induced deformation variables (y and φ) vanish; and at the free top of the tower, the internal force variables (M and R) are *a priori* known, as expressed by the following four boundary conditions

$$y(x=0)=0,\quad \varphi(x=0)=0,\quad M(x=\ell)=0,\quad R(x=\ell)=0.25\text{MN}. \tag{7.4}$$

Combining Eqs. (7.1) and (7.2), with the aim to eliminate the state variables $\varphi(x)$, $\kappa(x)$, $M(x)$, and $R(x)$, delivers the following governing fourth-order differential equation for the force-induced deflection $y(x)$

$$\begin{aligned}EI(x)\frac{d^4y(x)}{dx^4}+2\frac{dEI(x)}{dx}\frac{d^3y(x)}{dx^3}+\Big[\frac{d^2EI(x)}{dx^2}-N(x)\Big]\frac{d^2y(x)}{dx^2}\\ -\frac{dN(x)}{dx}\frac{dy(x)}{dx}=q(x)+\frac{dN(x)}{dx}\frac{dy_{ini}(x)}{dx}+N(x)\frac{d^2y_{ini}(x)}{dx^2}.\end{aligned} \tag{7.5}$$

Eq. (7.5) provides the motivation to express the boundary conditions (7.4) through the function $y(x)$ and its derivatives with respect to x. While the first condition (7.4) is already formulated in the governing variable $y(x)$, the second condition (7.4) together with the third Eq. (7.2) imply that the first derivative of $y(x)$ vanishes at the beginning of the integration interval. Hence, we reformulate the first two boundary conditions (7.4) by replacing the label "$(x=0)$" by index a, and by adapting the notation defined in (1.1), resulting in

$$y_a^{(0)}=0,\quad y_a^{(1)}=0. \tag{7.6}$$

We are left with reformulating the boundary conditions at the top of the tower. To this end, we first combine Eqs. (7.2) in order to eliminate the state variables $\varphi(x)$ and $\kappa(x)$, and we solve the resulting expression for $M(x)$ delivering

$$M(x)=-EI(x)\frac{d^2y(x)}{dx^2}. \tag{7.7}$$

Specifying (7.7) for the top of the tower, i.e., for $x=\ell$, replacing the label "$(x=\ell)$" by index b, considering the notation defined in (1.1), and formulating the third boundary condition (7.4) yields

$$M_b=-EI_by_b^{(2)}=0. \tag{7.8}$$

Finally, we combine Eqs. (7.2) and the second Eq. (7.1) in order to eliminate the state variables $\varphi(x)$, $\kappa(x)$, and $M(x)$. Solving the resulting expression for $R(x)$ delivers:

$$R(x)=-\frac{dEI(x)}{dx}\frac{d^2y(x)}{dx^2}-EI(x)\frac{d^3y(x)}{dx^3}+N(x)\Big[\frac{dy_{ini}(x)}{dx}+\frac{dy(x)}{dx}\Big]. \tag{7.9}$$

Specifying (7.9) for the top of the tower, i.e., for $x=\ell$, replacing the label "$(x=\ell)$" by index b, considering the notation defined in (1.1), and formulating the last boundary condition (7.4) yields

$$R_b = -EI_b^{(1)} y_b^{(2)} - EI_b y_b^{(3)} + N_b \left[y_{ini,b}^{(1)} + y_b^{(1)} \right] = 0.25\text{MN}. \tag{7.10}$$

7.2 Solution by means of the presented method

The governing differential equation (7.5) together with the boundary conditions (7.6), (7.8), and (7.10) are ready to be solved by the presented solution method, provided that functions $\rho(x)$, $q(x)$, $y_{ini}(x)$, and $N(x)$ as well as constants E and t are specified numerically. The radius $\rho(x)$ of the tower and the transversal loading $q(x)$ are considered to decrease linearly with increasing tower height (see Fig. 14):

$$\rho(x) = \rho_a + (\rho_b - \rho_a)\frac{x}{\ell} \quad \text{with} \quad \begin{cases} \rho_a = 4.0\text{m}, \\ \rho_b = 2.5\text{m}, \end{cases} \tag{7.11a}$$

$$q(x) = q_a + (q_b - q_a)\frac{x}{\ell} \quad \text{with} \quad \begin{cases} q_a = 0.017\text{MN/m}, \\ q_b = 0.011\text{MN/m}. \end{cases} \tag{7.11b}$$

Young's modulus E of reinforced concrete and the width t of the annular cross-section are given as

$$E = 30000\text{MNm}^2 \quad \text{and} \quad t = 0.5\text{m}. \tag{7.12}$$

As for the initial (stress-free) deflection of the tower, we consider a linear inclination (angle $\psi_0 = 0.001$), resulting in the following definition of $y_{ini}(x)$

$$y_{ini}(x) = 0.001x. \tag{7.13}$$

Considering (i) an axial force amounting to 20 MN at the top of the tower (see Fig. 14) and (ii) a distributed normal force per unit length of the tower, reading as $n(x) = n_a + (n_b - n_a)x/\ell$ with $n_a = 0.48\text{MN/m}$ and $n_b = 0.30\text{MN/m}$, results in the following solution for the axial force distribution [22]:

$$N(x) = -78.5\text{MN} + n_a x + (n_b - n_a)\frac{x^2}{2\ell}. \tag{7.14}$$

When solving the problem at hand according to the outline of the presented solution method described in Subsection 3.2, four integration constants are introduced: $y_a^{(0)}$, $y_a^{(1)}$, $y_a^{(2)}$, and $y_a^{(3)}$. While boundary conditions (7.6) provide direct access to the first two integration constants, the remaining two integration constants $y_a^{(2)}$ and $y_a^{(3)}$ are to be identified from the boundary conditions at the top of the tower, see (7.8) and (7.10). We are left with establishing relations between the boundary values $y_b^{(1)}$, $y_b^{(2)}$, and $y_b^{(3)}$, appearing in (7.8) and (7.10), and the two integration constants $y_a^{(2)}$ and $y_a^{(3)}$. To this end, we focus on point 11 of the listing in Subsection 3.2, where relations between boundary values of

$y(x)$ and its derivatives with respect to x are established. Evaluating $y_b^{(1)}$, $y_b^{(2)}$, and $y_b^{(3)}$ according to (3.20) yields, under consideration of $y_a^{(0)}=0$ and $y_a^{(1)}=0$ (see (7.6)),

$$y_b^{(1)}=(\vec{u}^{[3]})^T\vec{w}+\left[x_b^{[1]}+(\vec{u}^{[3]})^T\vec{v}^{[2]}\right]y_a^{(2)}+\left[x_b^{[2]}+(\vec{u}^{[3]})^T\vec{v}^{[3]}\right]y_a^{(3)}, \tag{7.15a}$$

$$y_b^{(2)}=(\vec{u}^{[2]})^T\vec{w}+\left[x_b^{[0]}+(\vec{u}^{[2]})^T\vec{v}^{[2]}\right]y_a^{(2)}+\left[x_b^{[1]}+(\vec{u}^{[2]})^T\vec{v}^{[3]}\right]y_a^{(3)}, \tag{7.15b}$$

$$y_b^{(3)}=(\vec{u}^{[1]})^T\vec{w}+\left[(\vec{u}^{[1]})^T\vec{v}^{[2]}\right]y_a^{(2)}+\left[x_b^{[0]}+(\vec{u}^{[1]})^T\vec{v}^{[3]}\right]y_a^{(3)}. \tag{7.15c}$$

$y_a^{(2)}$ and $y_a^{(3)}$ are the only unknowns on the right-hand sides of Eqs. (7.15). The coefficients of (7.15), i.e.,

$$c_c^{[1]}=(\vec{u}^{[3]})^T\vec{w}, \qquad c_2^{[1]}=x_b^{[1]}+(\vec{u}^{[3]})^T\vec{v}^{[2]}, \qquad c_3^{[1]}=x_b^{[2]}+(\vec{u}^{[3]})^T\vec{v}^{[3]}, \tag{7.16a}$$

$$c_c^{[2]}=(\vec{u}^{[2]})^T\vec{w}, \qquad c_2^{[2]}=x_b^{[0]}+(\vec{u}^{[2]})^T\vec{v}^{[2]}, \qquad c_3^{[2]}=x_b^{[1]}+(\vec{u}^{[2]})^T\vec{v}^{[3]}, \tag{7.16b}$$

$$c_c^{[3]}=(\vec{u}^{[1]})^T\vec{w}, \qquad c_2^{[3]}=(\vec{u}^{[1]})^T\vec{v}^{[2]}, \qquad c_3^{[3]}=x_b^{[0]}+(\vec{u}^{[1]})^T\vec{v}^{[3]}, \tag{7.16c}$$

can be computed after having completed the first 10 points of the listing of Subsection 3.2. We now insert Eqs. (7.15) into the boundary conditions (7.8) and (7.10) which results in two algebraic equations for the two sought integration constants $y_a^{(2)}$ and $y_a^{(3)}$, with the solution:

$$y_a^{(2)}=\frac{\left(c_3^{[2]}c_c^{[3]}-c_c^{[2]}c_3^{[3]}\right)EI_b+\left(c_c^{[2]}c_3^{[1]}-c_3^{[2]}c_c^{[1]}-c_3^{[2]}y_{ini,b}^{(1)}\right)N_b+c_3^{[2]}R_b}{\left(c_3^{[3]}c_2^{[2]}-c_2^{[3]}c_3^{[2]}\right)EI_b+\left(c_2^{[1]}c_3^{[2]}-c_3^{[1]}c_2^{[2]}\right)N_b}, \tag{7.17a}$$

$$y_a^{(3)}=\frac{\left(c_2^{[3]}c_c^{[2]}-c_c^{[3]}c_2^{[2]}\right)EI_b+\left(c_c^{[1]}c_2^{[2]}-c_2^{[1]}c_c^{[2]}+c_2^{[2]}y_{ini,b}^{(1)}\right)N_b-c_2^{[2]}R_b}{\left(c_3^{[3]}c_2^{[2]}-c_2^{[3]}c_3^{[2]}\right)EI_b+\left(c_2^{[1]}c_3^{[2]}-c_3^{[1]}c_2^{[2]}\right)N_b}. \tag{7.17b}$$

Now, all integration constants are identified (see (7.6) and (7.17)), such that the further evaluation of the solution is the same as the one described for initial value problems, see point 12 of the listing of Subsection 3.2, providing access to $y(x)$ and to its derivatives with respect to x. The beam theory-related state variables $\varphi(x)$, $M(x)$, and $R(x)$ are accessible by means of post-processing based on the second of the Eqs. (7.2), as well as on (7.7) and (7.9).

The numerical solution, obtained with a grid point number $n=2001$ and with the integration method based on piecewise exact integration of quadratic approximation polynomials, satisfactorily reproduces the benchmark solution described in [22], see Fig. 15.

Next, we investigate the accuracy of our results as a function of the number of chosen grid points n, i.e., we repeatedly solve the ODE (7.5), based on many different choices for grid point number n. In the context of defining meaningful prediction errors, we set our focus on beam theory-related boundary values which were not given by boundary conditions, i.e., on y_b, φ_b, M_a, and R_a. In addition, we note that the solution provided in [22]

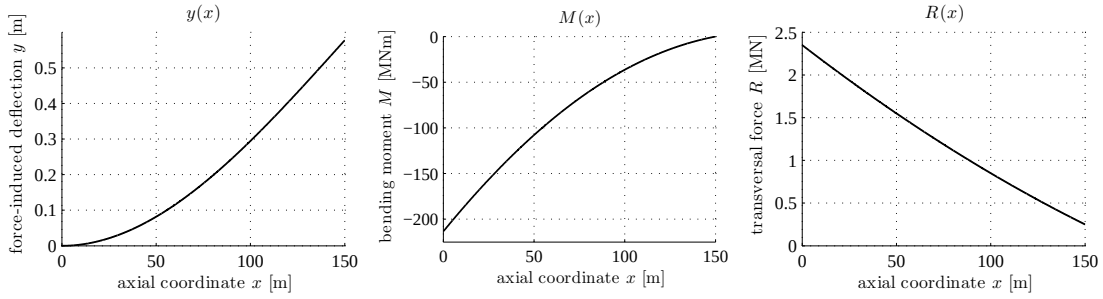
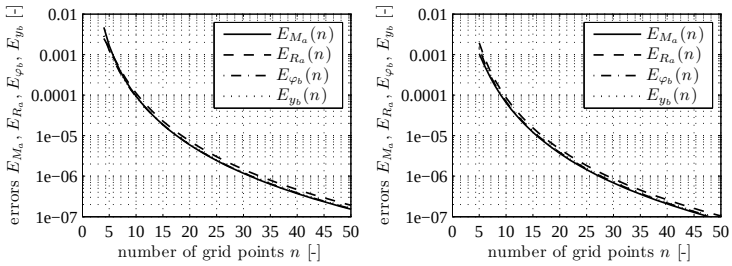


Figure 15: Solution for the slender tower problem (7.5) specified for input values (7.11)-(7.14), and $t=0.5$m, with boundary conditions (7.6), (7.8), and (7.10), computed based on the proposed solution method with grid point number $n=2001$ and piecewise exact integration of quadratic approximation polynomials, see Section 2: (a) solution function $y(x)$, (b) bending moment distribution $M(x)$ according to (7.7), (c) transversal force distribution $R(x)$ according to (7.9); for details on numerical results see also Table 3.



	(a)	(b)
errors $< 10^{-2}$	$n \geq 4$	$n \geq 4$
errors $< 10^{-3}$	$n \geq 6$	$n \geq 6$
errors $< 10^{-4}$	$n \geq 11$	$n \geq 10$
errors $< 10^{-5}$	$n \geq 19$	$n \geq 17$
errors $< 10^{-6}$	$n \geq 34$	$n \geq 29$

Figure 16: Convergence study for tower problem: errors (7.18) as a function of the number of used grid points n: (a) for piecewise exact integration of quadratic approximation polynomials, and (b) for piecewise exact integration of cubic approximation polynomials.

is numerically precise up to six significant digits. In order to investigate even higher precisions, we treat the results obtained with $n=2001$ (see Fig. 15) as the reference solution, and we compare the computed state variables y_b, φ_b, M_a and R_a with the corresponding values of the reference solution, based on the following definitions for the prediction errors:

$$E_\alpha = \left| \frac{\alpha(n) - \alpha(n=2001)}{\alpha(n=2001)} \right|, \quad \alpha = y_b, \varphi_b, M_a, R_a. \tag{7.18}$$

Errors (7.18) are evaluated for both integration methods: quadratic and cubic approximation polynomials (see Fig. 16).

Given a fixed grid point number, piecewise exact integration of cubic approximation polynomials results again in higher precisions than piecewise exact integration of quadratic approximation polynomials. With the former integration method, 5, 10, and 20 grid point deliver results with errors (7.18) smaller than 1.95×10^{-3}, 8.68×10^{-5}, and 4.65×10^{-6}, respectively.

Table 3: Numerical results for tower problem obtained with grid point number $n = 2001$ and piecewise exact integration of quadratic approximation polynomials, see also Fig. 15.

x [m]	$y(x)$ [m]	$\varphi(x)$ [mrad]	$M(x)$ [MNm]	$R(x)$ [MN]
0	0.00000000	0.0000000	−213.47556	2.3500
15	0.00777161	1.0251168	−178.41800	2.0995
30	0.03039607	1.9787338	−146.19708	1.8580
45	0.06672598	2.8508139	−116.93984	1.6255
60	0.11546156	3.6311097	−90.747720	1.4020
75	0.17514929	4.3094239	−67.694674	1.1875
90	0.24418547	4.8760118	−47.825588	0.9820
105	0.32082731	5.3221969	−31.154847	0.7855
120	0.40321547	5.6413116	−17.665266	0.5980
135	0.48941430	5.8301408	−7.3074773	0.4195
150	0.57747914	5.8911576	±0.0000000	0.2500

8 Discussions

We have presented a numerical solution method for linear, inhomogeneous, m-th order, ordinary differential equations with variable coefficients. The method is applicable for both initial value problems and boundary value problems, without raising the need to be adapted to specific properties of the underlying differential equation. In addition, the method can be easily implemented into modern software providing environments for numerical matrix-vector operation-based calculus, see Subsection 3.2, and the extension towards higher-order integration schemes is straightforward, see Section 2. Hence, the presented method complies with the requirement profile listed in the introductory section.

The major difference between the proposed solution scheme and other existing methods is that it is not only based on local approximation by means of polynomials and on their exact integration, but also on *averaging* these integrals. In more detail, local approximation by cubic polynomials and their exact integration is carried out within all existing subintervals containing four neighboring grid points, moving from the beginning of the overall integration interval to its end. Since these subintervals are overlapping each other, also the computed integrals refer to overlapping domains and, therefore, they are averaged arithmetically. This averaging is the reason why the proposed method based on *third*-order approximation polynomials could be shown to perform similarly to a *fourth*-order Runge-Kutta scheme (Subsection 4.1). When applying the presented method to a second-order boundary value problem (Subsection 4.2), it outperformed significantly the finite difference method.

Although the derivation of the method is non-trivial, the finally obtained solution concept can be simply implemented, see, e.g., Appendix C for the MATLAB solution of the stiff initial value problem (4.1) based on cubic approximation polynomials. In this context, the proposed method turns out to be user-friendly, because the formal mode of implementation is always the same (see, e.g., Appendix C), i.e., it is independent of

the chosen order of the local approximation polynomials. The only difference regarding different approximation orders concerns the integers in the integration matrices A^1, and they can be computed beforehand, such as described in Section 2.

The stability of the proposed concept, the influence of non-linear coefficient functions on the accuracy of a numerical solution, and numerical efficiency are discussed next.

- Up to the best of the authors' knowledge, the differential equation (4.1) represents a benchmark problem regarding stability of a numerical integration method. Since no stability problem was encountered when applying the proposed method to (4.1), it appears to be suitably robust. We conclude that the proposed method is well-suited for solving so-called stiff differential equations without stability problems.
- When applying the proposed method to a problem involving highly non-linear perturbation functions, see the pendulum problem (5.4) and Fig. 8, we obtain again satisfactory results. We conclude that the proposed method delivers – based on a suitable number of grid points – results which are accurate enough for engineering purposes. In order to find a suitable number of grid points, convergence studies are recommended. In this context, it is noteworthy that numerical results obtained with a monotonously increasing grid point number can be expected to oscillate around the exact solution, whereby the error amplitude decreases and the zero-error crossing-distance increases with increasing grid point number (see, e.g., Fig. 4).
- Regarding computational efficiency, we note that numerical efforts increase overlinearly with the grid point number (Fig. 17) because the inversion of the D matrix (see (3.19)) is a more and more demanding task given an increasing grid point number. Still, computation of a solution based on a reasonable number of grid points takes only a few milliseconds on a standard PC, and this is negligible with respect to the time required to choose and implement any solution scheme. We conclude that the proposed method is sufficiently efficient in order to represent an interesting alternative to traditional methods.

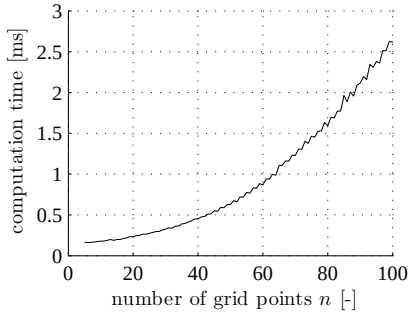


Figure 17: Computational efficiency of the proposed method regarding solution of the stiff initial value problem (4.1): average computation time as a function of grid point number (for each grid point number the problem was solved 10000 times on a standard Linux PC, and the required time was averaged).

Another main advantage of the presented solution concept over the here-discussed competing approaches is that the proposed method is not only applicable to initial value problems, but also to boundary value problems, no matter whether complicated boundary conditions refer to the beginning or to the end of the integration interval. This turned out to be beneficial for studying steady-state heat transfer through a cooling web (Section 6), and the structural analysis of a slender tower based on second-order beam theory (Section 7).

All problems considered herein were characterized by a smooth perturbation function and smooth coefficient functions within the interval of interest, but this is not a restriction of the presented method. Considering that the perturbation function and the coefficient functions are only piecewise continuous, the solution method should be applied within subintervals in which the aforementioned functions are indeed smooth, and the total solution could be composed from these elementary solutions, similar to the procedure described in Subsection 5.3. In such a case, generally speaking, transition conditions (jump conditions) for the state variables will have to be considered at the subinterval boundaries. While we note that the presented method provides a framework for uncomplicated formulation of such transition conditions, further details are beyond the scope of the present paper. The latter statement also holds for potential extensions towards consideration of sets of linear ordinary differential equations, linear partial differential equations, or nonlinear ordinary differential equations. These current limitations represent motivation for future work.

Appendix

A List of nomenclature

$a_i(x)$	coefficient function multiplied with i-th derivative of $y(x)$
$a^{[i]}(x)$	identical with $a_i(x)$
$a_k^{[i]}$	value of $a^{[i]}(x)$ at $x=x_k$
$\bar{c}$	torsional spring stiffness
$c_1^{[i]}$	coefficient, see (7.16)
$c_2^{[i]}$	coefficient, see (7.16)
$c_c^{[i]}$	coefficient, see (7.16)
$\vec{d}^{[j]}$	vector of coefficients, see (3.10)
$d_k^{[j]}$	k-th component of $\vec{d}^{[j]}$
$f(x)$	function to be integrated
f_s	value of $f(x)$ at position $x=x_s$
$\tilde{f}_k(\xi)$	approximation function for $f(x)$ referring to ξ with origin at x_k
$h(x)$	height of cooling web
h_a	boundary value of $h(x)$ at $x=x_a$

h_b	boundary value of $h(x)$ at $x=x_b$
i	index related to order of derivation
j	index related to the order of derivation
k	index related to positions of x-axis
ℓ	distance of point mass from center of rotation (in Section 5), length of cooling web (in Section 6), and height of tower (in Section 7)
ℓ_0	mean distance of point mass from center of rotation
m	order of underlying ordinary differential equation
m_d	modulation depth
n	number of equidistant points used to discretize the domain of interest
$n(x)$	distributed normal loading of tower
n_a	boundary value of $n(x)$ at $x=x_a$
n_b	boundary value of $n(x)$ at $x=x_b$
$q(x)$	heat flux per unit area (in Section 5) and transversal loading of tower (in Section 7)
q_a	boundary value of $q(x)$ at $x=x_a$
q_b	boundary value of $q(x)$ at $x=x_b$
$r(x)$	perturbation function
$\vec{r}$	vector with components r_k, $k=1,2,\cdots,n$
r_k	value of $r(x)$ at position $x=x_k$
s	index related to positions of x-axis
t	time (in Section 5), width of cooling web (in Section 6), and thickness of annular cross-section of tower (in Section 7)
$\vec{u}^{[i]}$	vector containing components of matrix A^i, see (3.21)
$\vec{u}_k^{[i]}$	k-th component of vector $\vec{u}^{[i]}$
$\vec{v}^{[i]}$	vector of coefficients, see (3.12)
$v_k^{[i]}$	k-th component of $\vec{v}^{[i]}$
$\vec{w}$	vector of coefficients, see (3.12)
w_k	k-th component of $\vec{d}$
x	variable
x_a	value of x at the beginning of the domain of interest
x_b	value of x at the end of the domain of interest
x_k	value of x at k-th point used to discretize the domain of interest
$x_k^{[j]}$	$=(x_k-x_a)^j/j!$, see (3.8)
$y(x)$	sought function (force-induced deflection of tower in Section 7)
$y^{(i)}(x)$	i-th derivative of $y(x)$ with respect to x
$\vec{y}^{(m)}(x)$	vector with components $y_k^{(m)}$, $k=1,2,\cdots,n$
$y_k^{(m)}$	value of $y^{(m)}(x)$ at position $x=x_k$
$y_a^{(i)}$	boundary value of $y^{(i)}$ at $x=x_a$

$y_b^{(i)}$	boundary value of $y^{(i)}$ at $x=x_b$
$y_k^{(i)}$	value of $y^{(i)}(x)$ at position $x=x_k$
$y_s^{(i)}$	value of $y^{(i)}(x)$ at position $x=x_s$
$y_{ini}(x)$	initial deflection of tower, in the stress-free configuration
$y_{ini,b}$	boundary values of $y_{ini}(x)$ at $x=x_b$
A	integration matrix
A^1	integration matrix for single integration
A^i	integration matrix for i-fold integration
A_{ks}	ks-th component of A
$A_{k,s}$	ks-th component of A
A^1_{ks}	ks-th component of A^1
A^i_{ks}	ks-th component of A^i
$A(x)$	cross-sectional area of cooling web
D	matrix of coefficients, see (3.10)
D_{ks}	ks-th component of D
E_b	prediction error at position $x=x_b$
E_k	prediction error at position $x=x_k$
E_n	prediction error at position $x=x_n$
$E_{\max}$	maximum prediction error out of E_k evaluated at all grid points
E_{mean}	prediction error averaged over E_k evaluated at all grid points
E	Young's modulus of the tower material
$EI(x)$	bending stiffness of tower
EI_b	boundary value of $EI(x)$ at $x=x_b$
J	moment of inertia
$N(x)$	axial force of tower
$M^{[i]}$	matrix of coefficients, see (3.16)
$M^{[i]}_{ks}$	ks-th component of matrix $M^{[i]}$
M_p	point mass
$M(x)$	bending moment of tower
M_b	boundary value of $M(x)$ at $x=x_b$
$Q(x)$	heat flux through the cooling web, in x-direction
Q_a	boundary value of $Q(x)$ at $x=x_a$
Q_b	boundary value of $Q(x)$ at $x=x_b$
$Q_{\ell at}$	heat loss through the four lateral surfaces of the cooling web
$R(x)$	transversal force of tower
T	modulation period (Section 5)
T_a	boundary value of $T(x)$ at $x=x_a$
T_{air}	temperature of air surrounding the cooling web
$T(x)$	temperature distribution in cooling web (Section 6)
$F_k\|_k^{k+1}$	value of integral of $\tilde{f}_k(x)$ from x_k to x_{k+1}

α	index
α_h	heat transfer coefficient
β	angle of inclination of cooling web surface with respect to midsurface
γ	damping constant
Δx	increment of x
δ_{ks}	Kronecker delta
η	thermal conductivity
κ	bending-induced curvature of tower axis
λ	width of intervals used to discretize the domain of interest
ξ	local coordinate which is parallel to x, and integration parameter
$\rho(x)$	radius of cross-section of tower
ρ_a	boundary value of $\rho(x)$ at $x=x_a$
ρ_b	boundary value of $\rho(x)$ at $x=x_b$
$\varphi(t)$	angular coordinate describing position of the pendulum at time t (Section 5)
$\varphi(x)$	cross-sectional rotation of tower (Section 7)
$\dot{\varphi}$	time-derivative of $\varphi(t)$
$\ddot{\varphi}$	time-derivative of $\dot{\varphi}$
φ_b	boundary value of $y(x)$ at $x=x_a$
ω	angular frequency of the modulation
ω_0	constant being equal to $\sqrt{\bar{c}/(2M_p\ell_0^2)}$

B Other numerical solution methods

B.1 Backward Euler method

Considering first-order initial value problems of the form

$$y'=f(x,y), \quad y(x_0)=y_0, \tag{B.1}$$

where y' stands for the first derivative of y with respect to x, and introducing the grid point distance λ, such that grid point are located at

$$x_{k+1}=x_k+\lambda \qquad \forall k=1,2,\cdots,(n-1), \tag{B.2}$$

the backward Euler method approximates, at position x_{k+1}, the left-hand side of (B.1) by means of the following finite difference quotient

$$y'(x_{k+1})\approx\frac{y_{k+1}-y_k}{\lambda}, \tag{B.3}$$

such that sought function values $y(x_{k+1})$ follow from knowledge on $y(x_k)$ as

$$y_{k+1}=y_k+\lambda f(x_{k+1},y_{k+1}). \tag{B.4}$$

B.2 Runge-Kutta method "RK4"

Considering the first-order initial value problem (B.1) and introducing the grid point distance λ, such that nodes are located at positions given in (B.2), the sought function values $y(x_{k+1})$ follow from knowledge on $y(x_k)$ as

$$y_{k+1}=y_k+\tfrac{\lambda}{6}(\varepsilon_1+2\varepsilon_2+2\varepsilon_3+\varepsilon_4), \tag{B.5}$$

where

$$\varepsilon_1=f(x_k,y_k), \qquad \varepsilon_2=f(x_k+\frac{1}{2}\lambda,y_k+\frac{1}{2}\lambda\varepsilon_1), \tag{B.6a}$$

$$\varepsilon_3=f(x_k+\frac{1}{2}\lambda,y_k+\frac{1}{2}\lambda\varepsilon_2), \qquad \varepsilon_4=f(x_k+\lambda,y_k+\lambda\varepsilon_3). \tag{B.6b}$$

B.3 Finite difference method

Approximating the derivatives in the differential equation

$$a_2(x)\frac{d^2y(x)}{dx^2}+a_1(x)\frac{dy(x)}{dx}+a_0(x)y(x)=r(x) \tag{B.7}$$

by finite difference expressions, the following discretized version of (B.7) is obtained:

$$a_2(x_k)\frac{y_{k+1}-2y_k+y_{k-1}}{\lambda^2}+a_1(x_k)\frac{y_{k+1}-y_{k-1}}{2\lambda}+a_0(x_k)y_k=r(x_k), \tag{B.8}$$

$\forall k=2,3,\cdots,(n-1)$.

C Implementation of proposed solution concept

C.1 MATLAB Code for stiff initial value problem (4.1)

```
xa = 0;     % lower bound of integration interval
xb = 1;     % upper bound of integration interval
ng = 100 ; % number of grid points
y0a = 1;    % initial value y(x=xa)
lambda = (xb-xa)/(ng-1); % grid point distance
x = (xa:lambda:xb)';      % coordinates of grid points
x0 = x.^0/factorial(0);  % vector x^0 / 0!
x1 = x.^1/factorial(1);  % vector x^1 / 1!
a1 = ones(size(x));       % evaluation of coefficient function a1
a0 = +15*ones(size(x));  % evaluation of coefficient function a0
r  = zeros(size(x));      % evaluation of perturbation function r
M0 = zeros(ng,ng);        % initialize M0 matrix
M1 = zeros(ng,ng);        % initialize M0 matrix
for k=1:1:ng
   M0(k,k) = a0(k);       % fill diagonal of M0 matrix with a0 values
```

```
   M1(k,k) = a1(k);       % fill diagonal of M1 matrix with a1 values
end
A0 = eye(ng);                    % A0 matrix
A1 = getA1_cubic(ng,lambda); % A1 matrix (cubic approximation)
Dinv = (M1*A0 + M0*A1)^(-1); % inverse of D matrix
d1 = M0*x1 + M1*x0;              % d1 vector
d0 = M0*x0;                      % d0 vector
v1 = - (Dinv*d1);                % v1 vector
v0 = - (Dinv*d0);                % v0 vector
w  =  (Dinv*r);                  % w vector
y1 = v0*y0a + w ;                % solution vector for y'(x)
y0 = x0*y0a + A1*y1;             % solution vector for y(x)
```

C.2 Subroutine: A1 matrix for cubic approximation

```
function A1 = getA1_cubic(ng,lambda)
Abasic = [[ 0   0   0   0   0   0   0   0   0   0   0]; ...
          [54 114 -30   6   0   0   0   0   0   0   0]; ...
          [48 192  48   0   0   0   0   0   0   0   0]; ...
          [54 162 162  54   0   0   0   0   0   0   0]; ...
          [54 168 132 168  54   0   0   0   0   0   0]; ...
          [53 171 136 136 171  53   0   0   0   0   0]; ...
          [53 170 139 140 139 170  53   0   0   0   0]; ...
          [53 170 138 143 143 138 170  53   0   0   0]; ...
          [53 170 138 142 146 142 138 170  53   0   0]; ...
          [53 170 138 142 145 145 142 138 170  53   0]; ...
          [53 170 138 142 145 144 145 142 138 170  53]] ;
if ng<12
   A = Abasic(1:ng,1:ng);
else
   A = Abasic;
   for k=12:1:ng
      A(k,1) = 53;
      A(k,k) = 53;
      A(k,2)   = 170;
      A(k,k-1) = 170;
      A(k,3)   = 138;
      A(k,k-2) = 138;
      A(k,4)   = 142;
      A(k,k-3) = 142;
      A(k,5)   = 145;
      A(k,k-4) = 145;
      for i = 6:1:k-5
         A(k,i) = 144;
      end
   end
end
A1 = lambda/144*A;
```

C.3 Subroutine: A1 matrix for quadratic approximation

```
function A1 = getA1_quadratic(ng,lambda)
Abasic = [[ 0  0  0  0  0 0]; ...
          [10 16 -2  0  0 0]; ...
          [ 8 32  8  0  0 0]; ...
          [ 9 27 27  9  0 0]; ...
          [ 9 28 22 28  9 0]; ...
          [ 9 28 23 23 28 9]] ;
if ng < 7
   A = Abasic(1:ng,1:ng);
else
   A = Abasic;
   for k=7:1:ng
      A(k,1) = 9;
      A(k,k) = 9;
      A(k,2)   = 28;
      A(k,k-1) = 28;
      A(k,3)   = 23;
      A(k,k-2) = 23;
      for i = 4:1:k-3
         A(k,i) = 24;
      end
   end
end
A1 = lambda/24*A;
```

Acknowledgments

Fruitful discussions with Gerhard Höfinger, from Feb 2007 until Dec 2010 research assistant at the Institute for Mechanics of Materials and Structures, Vienna University of Technology, are gratefully acknowledged.

References

[1] E. HAIRER, S. NØRSETT AND G. WANNER, Solving Ordinary Differential Equations I: Non-stiff Problems, 2nd edn., Springer Verlag: Berlin, 1993.

[2] E. HAIRER AND G. WANNER, Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems, 2nd edn., Springer Verlag: Berlin, New York, 1996.

[3] SD. COHEN AND AC. HINDMARSH, *CVODE, a stiff/nonstiff ODE solver in C*, Comput. Phys., 10(2) (1996), pp. 138–143.

[4] U. ASCHER AND L. PETZOLD, Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations, Society for Industrial and Applied Mathematics (SIAM): Philadelphia, 1998.

[5] H. ROSENBROCK, *Some general implicit processes for the numerical solution of differential equations*, The Computer J., 5(4) (1963), pp. 329–330.

[6] A. SANDU, JG. VERWER, JG. BLOM, EJ. SPEE, GR. CARMICHAEL AND FA. POTRA, *Benchmarking stiff ODE solvers for atmospheric chemistry problems II: Rosenbrock solvers*, Atmospheric Environment, 31(20) (1997), pp. 3459–3472.
[7] LF. SHAMPINE AND MW. REICHELT, *The MATLAB ODE suite*, SIAM J. Sci. Comput., 18(1) (1997), pp. 1–22.
[8] SN. ATLURI, JY. CHO AND HG. KIM, *Analysis of thin beams, using the meshless local Petrov-Galerkin method, with generalized moving least squares interpolations*, Comput. Mech., 24(5) (1999), pp. 334–347.
[9] YT. GU AND GR. LIU, *A local point interpolation method for static and dynamic analysis of thin beams*, Comput. Methods Appl. Mech. Eng., 190(42) (2001), pp. 5515–5528.
[10] PK. GUDLA AND R. GANGULI, *Discontinuous Galerkin finite element in time for solving periodic differential equations*, Comput. Methods Appl. Mech. Eng., 196(1-3) (2006), pp. 682–696.
[11] JI. RAMOS, *Piecewise-linearized methods for initial-value problems with oscillating solutions*, Appl. Math. Comput., 181(1) (2006), pp. 123–146.
[12] TC. FUNG, *Solving initial value problems by differential quadrature method-part 1: first-order equations*, Int. J. Numer. Methods Eng., 50(6) (2001), pp. 1411–1427.
[13] TC. FUNG, *Solving initial value problems by differential quadrature method-part 2: second- and higher-order equations*, Int. J. Numer. Methods Eng., 50(6) (2001), pp. 1429–1454.
[14] GJ. NIE AND Z. ZHONG Z, *Semi-analytical solution for three-dimensional vibration of functionally graded circular plates*, Comput. Methods Appl. Mech. Eng., 196(49-52) (2007), pp. :4901–4910.
[15] Y. HON AND Z. WU, *A quasi-interpolation method for solving stiff ordinary differential equations*, Int. J. Numer. Methods Eng., 48(8) (2000), pp. 1187–1197.
[16] N. MAI-DUY, *Solving high order ordinary differential equations with radial basis function networks*, Int. J. Numer. Methods Eng., 62(6) (2005), pp. 824–852.
[17] H. RUBIN AND M. AMINBAGHAI, *Wölbkrafttorsion bei veränderlichem, offenem Querschnitt – Hat die Biegezugstab-Analogie noch Gültigkeit? [Warping torsion for variable, open cross sections – Is the analogy for theory of beams with tensile force still valid?]*, Stahlbau, 76(10) (2007), pp. 747–760. In German.
[18] TC. FUNG, *Stability and accuracy of differential quadrature method in solving dynamic problems*, Comput. Methods Appl. Mech. Eng., 191(13-14) (2002), pp. 1311–1331.
[19] J. CHANG, Q. YANG AND C. LIU, *B-spline method for solving boundary value problems of linear ordinary differential equations*, Information Computing and Applications, Communications in Computer and Information Science, 106 (2010), R. Zhu, Y. Zhang, B. Liu and C. Liu (eds.), Springer Berlin Heidelberg, pp. 326–333.
[20] EI. BUTIKOV, *Parametric excitation of a linear oscillator*, Euro. J. Phys., 25(4) (2004), pp. 535–554.
[21] A. KRAUS AND A. BAR-COHEN, Thermal Analysis and Control of Electronic Equipment, Hemisphere Publishing Corporation: Washington, 1983.
[22] H. RUBIN, *Lösung linearer Differentialgleichungen beliebiger Ordnung mit Polynomkoeffizienten und Anwendung auf ein baustatisches Problem [Solution of linear differential equations of arbitrary order with polynomial coefficients and application to a structural analysis problem]*, ZAMM Zeitschrift für Angewandte Mathematik und Mechanik, 76(2) (1996), pp. 105–117. In German.