WKB-based scheme with adaptive step size control for the Schrödinger equation in the highly oscillatory regime

Jannis Körner, Anton Arnold*, Kirian Döpfner

Institute of Analysis and Scientific Computing, Technische Universität Wien, Wiedner Hauptstr. 8-10, 1040 Wien, Austria

---

Abstract

This paper is concerned with an efficient numerical method for solving the 1D stationary Schrödinger equation in the highly oscillatory regime. Being a hybrid, analytical-numerical approach it does not have to resolve each oscillation, in contrast to standard schemes for ODEs. We build upon the WKB-based (named after the physicists Wentzel, Kramers, Brillouin) marching method from [2] and extend it in two ways: By comparing the $\mathcal{O}(h)$ and $\mathcal{O}(h^2)$ methods from [2] we design an adaptive step size controller for the WKB method. While this WKB method is very efficient in the highly oscillatory regime, it cannot be used close to turning points. Hence, we introduce for such regions an automated methods switching, choosing between the WKB method for the oscillatory region and a standard Runge-Kutta-Fehlberg 4(5) method in smooth regions.

A similar approach was proposed recently in [9, 4], however, only for an $\mathcal{O}(h)$-method. Hence, we compare our new strategy to their method on two examples (Airy function on the spatial interval $[0, 10^8]$ with one turning point at $x = 0$ and on a parabolic cylinder function having two turning points), and illustrate the advantages of the new approach w.r.t. accuracy and efficiency.

Keywords: Schrödinger equation, highly oscillatory wave functions, higher order WKB-approximation, adaptive step size control, Airy function, parabolic cylinder function

---

*Corresponding author.

Email addresses: `jannis.koerner@tuwien.ac.at` (Jannis Körner), `anton.arnold@tuwien.ac.at` (Anton Arnold), `kirian.doepfner@gmail.com` (Kirian Döpfner)

1. Introduction

This paper deals with the numerical solution of the highly oscillatory 1D Schrödinger equation

$$\varepsilon^2 \varphi''(x) + a(x)\varphi(x) = 0 , \quad x \in \mathbb{R} . \tag{1.1}$$

Here, $0 < \varepsilon \ll 1$ is the rescaled Planck constant ($\varepsilon := \frac{\hbar}{\sqrt{2m}}$) and φ the (possibly complex valued) Schrödinger wave function. The real valued coefficient function $a(x)$ is related to the potential. We shall assume here that it is bounded away from zero, i.e. $a(x) \geq \tau$ for some $\tau > 0$. The (local) wave length of a solution φ to (1.1) is given by $\lambda(x) = \frac{2\pi\varepsilon}{\sqrt{a(x)}}$. Hence, for small values of ε the solution is highly oscillatory, especially in the semi-classical limit $\varepsilon \to 0$.

Oscillatory problems like (1.1) appear in a wide range of applications, e.g., quantum mechanics, electron transport in semiconductor devices, and acoustic scattering. For instance, the state of an electron that is injected with the prescribed energy E from the right boundary into an electronic device (e.g., diode), modeled on the interval $[0, 1]$, is described by the equation (see [2])

$$\begin{cases} -\varepsilon^2 \psi_E''(x) + V(x)\psi_E(x) = E\psi_E(x) , & x \in (0, 1) , \\ \psi_E'(0) + \mathrm{i}\, k(0)\psi_E(0) = 0 , \\ \psi_E'(1) - \mathrm{i}\, k(1)\psi_E(1) = -2\,\mathrm{i}\, k(1) , \end{cases} \tag{1.2}$$

where $k(x) := \varepsilon^{-1}\sqrt{E - V(x)}$ is the wave vector and V denotes the electrostatic potential. Note that our assumption $a(x) \geq \tau > 0$ implies $E > V(x)$, so the solution ψ_E becomes oscillatory. Then, one is often interested in macroscopic quantities like the electron density n and the current density j, which are given by

$$n(x) = \int_0^\infty f(k)|\psi_{E(k)}(x)|^2 \,\mathrm{d}k , \quad j(x) = \varepsilon \int_0^\infty f(k)\Im\left(\overline{\psi_{E(k)}(x)}\psi_{E(k)}'(x)\right) \mathrm{d}k , \tag{1.3}$$

where f represents the injection statistics of the electrons. Here, $\Im(\cdot)$ denotes the imaginary part and $E(k) = \varepsilon^2 k^2 + V$ means the energy for a given wave

vector k. In order to compute the quantities (1.3), the Schrödinger equation (1.2) has to be solved many times, as a fine grid in $E(k)$ is needed. Hence, efficient methods for the solution of (1.2) are of great interest in such applications. Instead of solving the boundary value problem (1.2) directly, one can also solve the equivalent initial value problem, which results if equation (1.1) on the interval $(0, 1)$ is augmented with the initial values $\varphi(0) = \varphi_0 = 1$ and $\varepsilon\varphi'(0) = \varphi_0' = -\mathrm{i}\sqrt{a(0)}$ with $a(x) = E - V(x)$. The solution φ of this initial value problem and that one of problem (1.2) are then related by

$$\psi_E(x) = -\frac{2\,\mathrm{i}\,k(1)}{\varphi'(1) - \mathrm{i}\,k(1)\varphi(1)}\varphi(x)\,,$$

according to [2]. Indeed, the method proposed in this paper will deal only with initial value problems for the Schrödinger equation (1.1), but through this equivalence it is equally suitable for solving problem (1.2).

In [2, 13, 14], efficient and accurate WKB-based (named after the physicists Wentzel, Kramers, Brillouin; cf. [15]) numerical schemes have been developed for (1.1) in the oscillatory regime. By transforming out the dominant oscillations, they allow to compute a solution using a coarse spatial grid with step size $h > \lambda$. In fact, the grid limitation can there be reduced to at least $h = \mathcal{O}(\sqrt{\varepsilon})$. Now, this work adds on top of the algorithm from [2] an adaptive step size control as well as a switching mechanism. This allows the algorithm to switch to a standard ODE method (e.g., Runge-Kutta) during the computation in order to avoid technical or efficiency problems in regions where the coefficient function $a(x)$ is very small or indeed equal to zero. We recall that the WKB-approximation is not valid close to turning points, i.e. where $a(x) = 0$. A switching mechanism was also used in [9], where the authors presented another WKB-based numerical scheme for the initial value problem corresponding to (1.1). Therefore, one goal of this work is to compare numerical results from our method with the results given by the method from [9], by considering two examples where the analytical solution is known.

Since numerical methods for oscillatory problems is an active field of research, let us mention some references that are intended for more general oscillatory problems, and hence also include adequate methods for (1.1): first the two monographs [18, 10]. The *adiabatic integrators* of [10, §XIV] are in fact closely related to a zeroth order WKB-approximation, see (2.2)-(2.3), below. Concerning (highly oscillatory) *Hamiltonian boundary value methods* we cite [7, 1].

This paper is organized as follows: In Section 2 we give a short review of the second order (w.r.t. ε) WKB-marching method from [2]. Section 3 then describes the adaptive step size control algorithm as well as the switching mechanism. In Section 4 we recap the Runge-Kutta-WKB method from [9] and point out the difference between their step size control and the one used in this paper. In Section 5 we present numerical investigations on the error estimators of our algorithm as well as a comparison of the numerical results of our method and the method from [9]. We then conclude in Section 6.

2. The WKB-marching method

We aim at solving the Schrödinger equation (1.1), augmented with the initial conditions

$$\varphi(x_0) = \varphi_0, \quad \varepsilon\varphi'(x_0) = \varphi_0'$$ (2.1)

with some $x_0 \in \mathbb{R}$. First we shall review the basics of the second order (w.r.t. ε) WKB-marching method from [2] with focus on the algorithm. The motivation for this method was the construction of a numerical scheme that is uniformly correct in ε and sometimes even asymptotically correct, i.e. the numerical error goes to zero with $\varepsilon \to 0$ while the grid size h remains constant. For further details see [2]. The method consists of two parts:

1. Analytic pre-processing of (1.1) by transforming the equation into a smoother (i.e. less oscillatory) problem that can be solved accurately and efficiently on a coarse grid.
2. Obtaining a numerical scheme by discretization of the smoother problem.

Analytic pre-processing. The well-known WKB-approximation (cf. [15]) for the oscillatory regime where $a(x) \geq \tau$ for some $\tau > 0$, is based on inserting the ansatz

$$\varphi(x) \sim \exp\left(\frac{1}{\varepsilon}\sum_{p=0}^{\infty}\varepsilon^p\phi_p(x)\right)$$ (2.2)

4

into equation (1.1). After a comparison of ε-powers one obtains the first three functions $\phi_p(x)$ as

$$\phi_0(x) = \pm\mathrm{i}\int_{x_0}^{x}\sqrt{a(y)}\,\mathrm{d}y\,,\tag{2.3}$$

$$\phi_1(x) = \ln(a(x)^{-\frac{1}{4}})\,,\tag{2.4}$$

$$\phi_2(x) = \mp\mathrm{i}\int_{x_0}^{x}b(y)\,\mathrm{d}y\,,\quad b(x) := -\frac{1}{2a(x)^{\frac{1}{4}}}\left(a(x)^{-\frac{1}{4}}\right)''\,.\tag{2.5}$$

Here the symbols $\pm$ and $\mp$ in (2.3) and (2.5) correspond to the fact that there is always a pair of approximate solutions to the Schrödinger equation (1.1), by analogy to the two fundamental solutions of (1.1). Hence the general solution is then a linear combination of the two. Therefore, a truncation of the sum (2.2) after $p = 2$ leads to the second order (w.r.t. ε) asymptotic WKB-approximation of $\varphi(x)$:

$$\varphi_2(x) = c_1\frac{\exp\left(\frac{\mathrm{i}}{\varepsilon}\phi^\varepsilon(x)\right)}{a(x)^{\frac{1}{4}}} + c_2\frac{\exp\left(-\frac{\mathrm{i}}{\varepsilon}\phi^\varepsilon(x)\right)}{a(x)^{\frac{1}{4}}}\,,\tag{2.6}$$

with constants $c_1, c_2 \in \mathbb{C}$ to be determined from initial or boundary conditions, and the phase is

$$\phi^\varepsilon(x) := \int_{x_0}^{x}\left(\sqrt{a(y)} - \varepsilon^2 b(y)\right)\,\mathrm{d}y\,.\tag{2.7}$$

In the WKB-marching method of [2] this second order WKB-approximation is used to transform (1.1) into a smoother problem. To clarify our terminology, we point out that this method (as well as the Runge-Kutta-WKB method of Section 4) has both a *WKB-order* (w.r.t. ε; referring to the used cut-off in the asymptotic expansion (2.2)) and a *numerical order* (w.r.t. the step size h; referring to the convergence order). Firstly, using the notation

$$U(x) = \begin{pmatrix} u_1(x) \\ u_2(x) \end{pmatrix} := \begin{pmatrix} a(x)^{\frac{1}{4}}\varphi(x) \\ \frac{\varepsilon\left(a(x)^{1/4}\varphi(x)\right)'}{\sqrt{a(x)}} \end{pmatrix}\,,\tag{2.8}$$

the second order differential equation (1.1) with the initial conditions (2.1) can be reformulated as a system of first order differential equations:

$$\begin{cases} U'(x) = \left[\frac{1}{\varepsilon}\mathbf{A}_0(x) + \varepsilon\mathbf{A}_1(x)\right]U(x)\,, & x > x_0\,, \\ U(x_0) = U_I\,. \end{cases}\tag{2.9}$$

5

Here, the two matrices $\mathbf{A}_0$ and $\mathbf{A}_1$ are given by

$$\mathbf{A}_0(x) := \sqrt{a(x)} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} ; \quad \mathbf{A}_1(x) := \begin{pmatrix} 0 & 0 \\ 2b(x) & 0 \end{pmatrix} .$$

Then, the first order system (2.9) for $U(x)$ is transformed by the change of variables

$$Z(x) = \begin{pmatrix} z_1(x) \\ z_2(x) \end{pmatrix} := \exp\left(-\frac{\mathrm{i}}{\varepsilon} \mathbf{\Phi}^\varepsilon(x) \right) \mathbf{P} U(x) , \tag{2.10}$$

with the two matrices

$$\mathbf{P} := \frac{1}{\sqrt{2}} \begin{pmatrix} \mathrm{i} & 1 \\ 1 & \mathrm{i} \end{pmatrix} ; \quad \mathbf{\Phi}^\varepsilon(x) := \begin{pmatrix} \phi^\varepsilon(x) & 0 \\ 0 & -\phi^\varepsilon(x) \end{pmatrix} ,$$

where ϕ^ε is the phase function defined in (2.7). This leads to the system

$$\begin{cases} Z'(x) = \varepsilon \mathbf{N}^\varepsilon(x) Z(x) , & x > x_0 , \\ Z(x_0) = Z_I = \mathbf{P} U_I , \end{cases} \tag{2.11}$$

where $\mathbf{N}^\varepsilon(x)$ is a (Hermitian) matrix with only off-diagonal non-zero entries:

$$N_{1,2}^\varepsilon(x) = b(x)\,\mathrm{e}^{-\frac{2\mathrm{i}}{\varepsilon}\phi^\varepsilon(x)} , \quad N_{2,1}^\varepsilon(x) = b(x)\,\mathrm{e}^{\frac{2\mathrm{i}}{\varepsilon}\phi^\varepsilon(x)} .$$

Since the transformation (2.10) eliminated the dominant oscillations, the system (2.11) can be solved numerically on a coarse grid $\{x_n, n \in \mathbb{N}_0\}$. Then the original solution can be recovered by the inverse transformation

$$U(x) = \mathbf{P}^{-1} \exp\left(\frac{\mathrm{i}}{\varepsilon} \mathbf{\Phi}^\varepsilon(x) \right) Z(x) .$$

It should be noted that, throughout the whole transformation from $U(x)$ to $Z(x)$, the phase integral $\phi^\varepsilon(x)$ is assumed to be known exactly. For a generalization using a spectral method to numerically compute the phase see [5].

Numerical scheme. The derivation of the second order (in h) scheme for (2.11) is obtained via the second order Picard approximation

$$\widetilde{Z}_{n+1} := \widetilde{Z}_n + \varepsilon \int_{x_n}^{x_{n+1}} \mathbf{N}^\varepsilon(x)\,\mathrm{d}x \widetilde{Z}_n + \varepsilon^2 \int_{x_n}^{x_{n+1}} \mathbf{N}^\varepsilon(x) \int_{x_n}^{x} \mathbf{N}^\varepsilon(y)\,\mathrm{d}y\mathrm{d}x \widetilde{Z}_n . \tag{2.12}$$

Since the entries of $\mathbf{N}^\varepsilon(x)$ are highly oscillatory, (2.12) involves (iterated) oscillatory integrals. With ϕ^ε assumed to be known exactly, they are then approximated using similar techniques as the *asymptotic method* in [12]. The first order (in h) scheme for (2.11) is derived by only taking into account the first two terms from (2.12). For both of these schemes we introduce the following notations:

$$b_0(y) := \frac{b(y)}{2\left(\sqrt{a(y)} - \varepsilon^2 b(y)\right)} \; ; \quad b_{k+1}(y) := \frac{1}{2\left(\phi^\varepsilon(y)\right)'} \frac{\mathrm{d}b_k}{\mathrm{d}y}(y) \, , \quad k = 0, 1, 2 \, ;$$

$$h_1(y) := \mathrm{e}^{\mathrm{i}\,y} - 1 \, ; \quad h_2(y) := \mathrm{e}^{\mathrm{i}\,y} - 1 - \mathrm{i}\,y \, .$$

Further, let $\{x_0, x_1, ..., x_N\}$ be a grid we want to compute the solution on, and $h := \max_{1 \le n \le N} |x_n - x_{n-1}|$ be the step size. Then both schemes read as follows:

<u>First order scheme:</u> Let $Z_0 := Z_I$ be the initial condition and let $n = 0, ..., N-1$. Then the algorithm updates as

$$Z_{n+1} = \left(I + \mathbf{A}_n^1\right) Z_n \, , \tag{2.13}$$

with the (Hermitian) matrix

$$\mathbf{A}_n^1 := \varepsilon^3 b_1(x_{n+1}) \begin{pmatrix} 0 & \mathrm{e}^{-\frac{2\mathrm{i}}{\varepsilon}\phi^\varepsilon(x_n)} h_1\left(-\frac{2}{\varepsilon} s_n\right) \\ \mathrm{e}^{\frac{2\mathrm{i}}{\varepsilon}\phi^\varepsilon(x_n)} h_1\left(\frac{2}{\varepsilon} s_n\right) & 0 \end{pmatrix}$$

$$- \mathrm{i}\,\varepsilon^2 \begin{pmatrix} 0 & b_0(x_n)\,\mathrm{e}^{-\frac{2\mathrm{i}}{\varepsilon}\phi^\varepsilon(x_n)} - b_0(x_{n+1})\,\mathrm{e}^{-\frac{2\mathrm{i}}{\varepsilon}\phi^\varepsilon(x_{n+1})} \\ b_0(x_{n+1})\,\mathrm{e}^{\frac{2\mathrm{i}}{\varepsilon}\phi^\varepsilon(x_{n+1})} - b_0(x_n)\,\mathrm{e}^{\frac{2\mathrm{i}}{\varepsilon}\phi^\varepsilon(x_n)} & 0 \end{pmatrix} \, ,$$

and the phase increments

$$s_n := \phi^\varepsilon(x_{n+1}) - \phi^\varepsilon(x_n) \, .$$

<u>Second order scheme:</u> Let $Z_0 := Z_I$ be the initial condition and let $n = 0, ..., N - 1$. Then the algorithm updates as

$$Z_{n+1} = \left(I + \mathbf{A}_{mod,n}^1 + \mathbf{A}_n^2\right) Z_n \, , \tag{2.14}$$

with the (Hermitian) matrix

$$\mathbf{A}^1_{mod,n} :=$$

$$-\,\mathrm{i}\,\varepsilon^2 \begin{pmatrix} 0 & b_0(x_n)\,\mathrm{e}^{-\frac{2\,\mathrm{i}}{\varepsilon}\phi^\varepsilon(x_n)} - b_0(x_{n+1})\,\mathrm{e}^{-\frac{2\,\mathrm{i}}{\varepsilon}\phi^\varepsilon(x_{n+1})} \\ b_0(x_{n+1})\,\mathrm{e}^{\frac{2\,\mathrm{i}}{\varepsilon}\phi^\varepsilon(x_{n+1})} - b_0(x_n)\,\mathrm{e}^{\frac{2\,\mathrm{i}}{\varepsilon}\phi^\varepsilon(x_n)} & 0 \end{pmatrix}$$

$$+\,\varepsilon^3 \begin{pmatrix} 0 & b_1(x_{n+1})\,\mathrm{e}^{-\frac{2\,\mathrm{i}}{\varepsilon}\phi^\varepsilon(x_{n+1})} - b_1(x_n)\,\mathrm{e}^{-\frac{2\,\mathrm{i}}{\varepsilon}\phi^\varepsilon(x_n)} \\ b_1(x_{n+1})\,\mathrm{e}^{\frac{2\,\mathrm{i}}{\varepsilon}\phi^\varepsilon(x_{n+1})} - b_1(x_n)\,\mathrm{e}^{\frac{2\,\mathrm{i}}{\varepsilon}\phi^\varepsilon(x_n)} & 0 \end{pmatrix}$$

$$+\,\mathrm{i}\,\varepsilon^4 b_2(x_{n+1}) \begin{pmatrix} 0 & -\,\mathrm{e}^{-\frac{2\,\mathrm{i}}{\varepsilon}\phi^\varepsilon(x_n)} h_1\left(-\frac{2}{\varepsilon}s_n\right) \\ \mathrm{e}^{\frac{2\,\mathrm{i}}{\varepsilon}\phi^\varepsilon(x_n)} h_1\left(\frac{2}{\varepsilon}s_n\right) & 0 \end{pmatrix}$$

$$-\,\varepsilon^5 b_3(x_{n+1}) \begin{pmatrix} 0 & \mathrm{e}^{-\frac{2\,\mathrm{i}}{\varepsilon}\phi^\varepsilon(x_n)} h_2\left(-\frac{2}{\varepsilon}s_n\right) \\ \mathrm{e}^{\frac{2\,\mathrm{i}}{\varepsilon}\phi^\varepsilon(x_n)} h_2\left(\frac{2}{\varepsilon}s_n\right) & 0 \end{pmatrix},$$

and the diagonal matrix

$$\mathbf{A}^2_n := -\,\mathrm{i}\,\varepsilon^3\,(x_{n+1} - x_n)\,\frac{b(x_{n+1})b_0(x_{n+1}) + b(x_n)b_0(x_n)}{2} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$$-\,\varepsilon^4 b_0(x_n)b_0(x_{n+1}) \begin{pmatrix} h_1\left(-\frac{2}{\varepsilon}s_n\right) & 0 \\ 0 & h_1\left(\frac{2}{\varepsilon}s_n\right) \end{pmatrix}$$

$$+\,\varepsilon^5 b_1(x_{n+1})\,[b_0(x_n) - b_0(x_{n+1})] \begin{pmatrix} h_2\left(-\frac{2}{\varepsilon}s_n\right) & 0 \\ 0 & -h_2\left(\frac{2}{\varepsilon}s_n\right) \end{pmatrix}.$$

3. Step size control and switching mechanism

The WKB scheme is efficient in the highly oscillatory regime, but not applicable close to turning points, i.e. zeros of $a(x)$, see [3]. This is evident already from the transformation (2.8), which does not make sense when $a(x) \leq 0$. For mixed problems, e.g., the Airy equation on $\mathbb{R}_0^+$ (see Section 5.1), which has a turning point at $x = 0$, it is therefore convenient to couple two different methods: a method for highly oscillatory ODEs (e.g., WKB-based) away from the turning point, and a standard ODE method (e.g., Runge-Kutta) close to the turning point, where the solution is smooth anyhow. Here, we choose the well-known Runge-Kutta-Fehlberg 4(5) (RKF45) scheme (cf. [11]) as the standard ODE method. The latter method will be applied directly on equation (1.1) and not on (2.11), since the WKB-transformation (2.8)-(2.11) is not permitted at turning points. The exact switching mechanism as well as the introduction of an adaptive step size control to the algorithm will be described in the two following subsections.

8

3.1. *The adaptive step size controller*

In order to compute the solution efficiently, an adaptive step size controller, based on an estimator for the local truncation error, will be added to the numerical methods. This control allows the step size to increase or decrease while aiming to keep the error estimator as close as possible within a given error tolerance. To illustrate the functionality of this step size controller, we shall consider a numerical scheme of order k. We are then able to apply this step size control individually to the different methods mentioned above.

Let $Y_n^{(k)} = (\varphi_n^{(k)}, \varphi_n'^{(k)})$ and $Y_n^{(k+1)} = (\varphi_n^{(k+1)}, \varphi_n'^{(k+1)})$ be two numerical solutions of order k and $k+1$ to approximate the exact solution $Y(x_n) = (\varphi(x_n), \varphi'(x_n))$ of the initial value problem (1.1), (2.1). E.g., one could choose the WKB schemes (2.13) and (2.14) of h-order 1 and 2, respectively. Next we want to decide whether to accept the numerical solution at x_n (typically the more accurate solution $Y_n^{(k+1)}$) or rather to retry the computation with a modified step size. To this end we define the estimator for the local truncation error as

$$\text{est}_n := \|Y_n^{(k)} - Y_n^{(k+1)}\|_\infty \,. \tag{3.1}$$

Let $h_{n,trial} := x_n - x_{n-1}$ be the (trial) step size which was used to compute the solutions at the current step n. We then use the common approach of varying the step size via the multiplicative control

$$h_{new} := \theta_n \cdot h_{n,trial} \,.$$

Here, we choose the factor θ_n to be based on the so-called *elementary controller* (e.g., see [17]). Additionally, we introduce limitations in such way that the step size controller responds "smoothly" to abrupt changes in the solution behaviour, that is, the ratio between two consecutive step sizes should not be exorbitantly large or small. That said, we choose the factor similar to [6, p. 310] as

$$\theta_n := \max\left(0.5, \min\left(2, 0.9\left(\frac{\text{ATol} + \text{RTol} \cdot \|Y_n^{(k+1)}\|_\infty}{\text{est}_n}\right)^{\frac{1}{k+1}}\right)\right), \tag{3.2}$$

where $\text{ATol} = \eta \cdot \text{Tol}$ and $\text{RTol} = \text{Tol}$ are absolute and relative error tolerances for a given master tolerance Tol, the values 0.5 and 2 are design parameters

9

that limit the ratio of two consecutive step sizes from below and above, and 0.9 is a common safety factor for increasing the probability of the next step to be accepted. Here, η is a scaling factor representing the gradual switch-over between absolute and relative errors, depending on the behaviour of the solution. This is because for $\|Y_n^{(k+1)}\|_\infty \to 0$ the ATol term in the numerator in (3.2) is dominating, whereas for large values of $\|Y_n^{(k+1)}\|_\infty$ the RTol term is dominating. For our numerical simulations in Section 5 we choose $\eta = 10^{-2}$. If $\mathrm{ATol} + \mathrm{RTol} \cdot \|Y_n^{(k+1)}\|_\infty \geq \mathrm{est}_n$, we accept the n-th step with the step size defined as $h_n := h_{n,trial}$ and define the trial step size for the next step as

$$h_{n+1,trial} := h_{new} \, .$$

However, if $\mathrm{ATol} + \mathrm{RTol} \cdot \|Y_n^{(k+1)}\|_\infty < \mathrm{est}_n$, the n-th step gets rejected and a reattempt is done with the smaller (trial) step size $h_{n,trial}$ by updating its value as

$$h_{n,trial} \to h_{new} \, .$$

Since such an acceptance criterion is based on aiming to maintain the local error in each step as close as possible to the given error tolerances it is often referred to as *error per step (EPS)* control. In practice, using EPS control one can hope to achieve a global truncation error proportional to $\mathrm{Tol}^{k/(k+1)}$ (e.g., see [6, p. 311]).

3.2. The switching mechanism

As already mentioned above, the algorithm shall automatically switch between two numerical methods, the WKB method in the oscillatory regime and another method, which is valid close to turning points. To realize this dynamical switching mechanism we follow a similar strategy as in [4]. To illustrate this procedure we now consider two numerical schemes of order $k^{(1)}$ and $k^{(2)}$, where the superscripts (1) and (2) correspond to the two schemes. In each step, the adaptive step size algorithm from the previous section is applied to both schemes individually up to the definition of the quantities $\theta_n^{(1)}$ and $\theta_n^{(2)}$, i.e., we just evaluate (3.1) and (3.2). Then, after checking the acceptance criterion $\mathrm{ATol} + \mathrm{RTol} \cdot \|Y_n^{(k^{(i)}+1)}\|_\infty \geq \mathrm{est}_n^{(i)}$ for each scheme $(i) \in \{(1),(2)\}$, the switching mechanism intervenes and it selects the acceptable numerical method that yields the larger value of $\theta_n^{(i)}$, for $i = 1,2$, hence yielding the larger proposed new step size. We thus favor the method with

10

the smaller error estimator, discounted by its respective order $k^{(i)}$. More precisely, we define

$$\Theta_n := \begin{cases} \theta_n^{(1)}, & \text{(1) accepted, (2) rejected} \\ \theta_n^{(2)}, & \text{(1) rejected, (2) accepted} \\ \max\left(\theta_n^{(1)}, \theta_n^{(2)}\right), & \text{otherwise} \end{cases}$$

and store the information on the method of choice in that n-th step. Through this procedure the algorithm does not only use the error estimator to find the next step size, but also to decide between the two methods.

If at least one method was accepted, the algorithm sets

$$h_n := h_{n,trial},$$
$$h_{n+1,trial} := \Theta_n \cdot h_{n,trial}.$$

Otherwise a reattempt is done with the smaller (trial) step size $h_{n,trial}$ by updating its value as

$$h_{n,trial} \rightarrow \Theta_n \cdot h_{n,trial}.$$

We remark that the coupling of two methods, as presented above, could incur additional computational costs, since in every step both methods have to be applied in order to compute Θ_n. However, in our case the WKB method (for highly oscillatory regions) and the standard ODE solver (for smoother regions) complement each other very well, yielding better results concerning accuracy as well as overall efficiency (see also Figures 7-8, 11 and 16).

4. The Runge-Kutta-WKB method

In this section we give a short review of the Runge-Kutta-WKB (RKWKB) method presented in [9] for the initial value problem (1.1), (2.1). The method is based on a dynamic switching mechanism between a standard Runge-Kutta scheme and a new stepping procedure that uses the WKB-ansatz (2.2) as an approximation of the true solution. This stepping procedure reads as follows:

$$\varphi_{n+1} := \gamma_+ f_+(x_n + h_n) + \gamma_- f_-(x_n + h_n), \tag{4.1}$$
$$\varphi'_{n+1} := \delta_+ f'_+(x_n + h_n) + \delta_- f'_-(x_n + h_n), \tag{4.2}$$
$$x_{n+1} := x_n + h_n, \tag{4.3}$$

where

$$\gamma_\pm := \frac{\varphi'_n f_\mp(x_n) - \varphi_n f'_\mp(x_n)}{f'_\pm(x_n)f_\mp(x_n) - f'_\mp(x_n)f_\pm(x_n)} \ , \tag{4.4}$$

$$\delta_\pm := \frac{\varphi''_n f'_\mp(x_n) - \varphi'_n f''_\mp(x_n)}{f''_\pm(x_n)f'_\mp(x_n) - f''_\mp(x_n)f'_\pm(x_n)} \ , \tag{4.5}$$

and φ''_n is computed from φ_n and equation (1.1). Here, $f_\pm$ are chosen as WKB-approximations (2.2) of some finite order. For instance, one gets the second (WKB-)order method by setting

$$f_\pm(x) := \frac{\exp\left(\pm\frac{i}{\varepsilon}\phi^\varepsilon(x)\right)}{a(x)^{\frac{1}{4}}} \ .$$

Note that this choice is equal to (2.6), but one can easily choose $f_\pm$ with higher (WKB-)orders. However, the stepping procedure (4.1)-(4.5) is always a first order (in h) numerical method. This holds because the coefficients $\gamma_\pm$ and $\delta_\pm$ are chosen in such way that one finds

$$\varphi_{n+1} = \varphi_n + \varphi'_n h + \mathcal{O}(h^2) \ ,$$
$$\varphi'_{n+1} = \varphi'_n + \varphi''_n h + \mathcal{O}(h^2) \ ,$$

from (4.1)-(4.3), as stated in [9].

It is also worth noting that in [9] the authors use a slightly different dynamic switching mechanism and a different step size control compared to the algorithm presented in Section 3. Firstly, their estimator of the relative error within the WKB-stepping procedure uses the difference between two numerical solutions $\varphi_n^{(k)}$ and $\varphi_n^{(k+1)}$ of different WKB-orders instead of different h-orders, simply since they do not have schemes of two different h-orders at their disposal. The algorithm then decides between a WKB step and a RK step by choosing the method with the smaller error estimator. In their more recent paper [4] the authors use a similar step size control and switching mechanism as presented in Section 3, but they do not limit the ratio of two consecutive step sizes and provide no option for controlling both absolute and relative errors as done in (3.2).

The goal of the following section is to compare numerical results from the WKB-marching method to results one gets using the RKWKB method. To both methods we will apply (exactly) the step size control and switching

mechanism from Section 3, for the sake of comparability. Since the WKB-stepping procedure of the RKWKB method is always of first order w.r.t. h, the definition of the error estimator (3.1) does not make sense. Therefore we shall use two different WKB orders (instead of h-orders) to be able to compute the error estimator in this case, as also done in [9]. Further, since our algorithm consists of a different step size update formula and switching criterion, we will call this modified RKWKB method simply RKWKBmod. Since this modification may produce slightly different numerical results compared to [9, 4] we will also include the original RKWKB method from [9] into our comparison.

5. Numerical results: WKB-marching method vs. the Runge-Kutta-WKB method

In this section we will compare numerical results of the WKB-marching method and of the RKWKB method by applying both algorithms to two examples, where exact analytical solutions are available. The first example corresponds to a linear coefficient function $a(x)$ and is taken from [9, 4], whereas the second example involves a quadratic function $a(x)$ and appears in [3]. In both examples the phase integral (2.7) in the WKB basis functions (2.3)-(2.5) can easily be computed exactly, hence we do not need any numerical integration routine here. By contrast, in [4] they evaluate the WKB basis functions with a numerical integration routine and therefore get another source of error in their method. Moreover, we will always use the second order WKB-marching method, since no scheme of higher WKB-order has been developed yet, see [2].

For clarity of the presentation we shall use in the sequel the following terminology for the methods to be compared:

WKB+RKF45	WKB-marching method (see Section 2) + step size and switching algorithms to RKF45 (see Section 3)
RKWKB	original method from [9]: Runge-Kutta-WKB (see Section 4) + original step size and switching algorithms (see Section 4)
RKWKBmod	modified method from [9]: Runge-Kutta-WKB (see Section 4) + modified step size and switching algorithms (see Section 3)
RKF45	Runge-Kutta-Fehlberg 4(5) scheme + step size algorithm (see Section 3)

Table 1: Terminology for the methods to be compared.

All simulations in this paper are done with MATLAB version 9.10.0.1669831 (R2021a).

5.1. First example: Airy function

The first example investigated in [9] is the Airy equation

$$
\begin{cases}
\varepsilon^2 \varphi''(x) + x\varphi(x) = 0, & x > 0, \\
\varphi(0) = \frac{3^{-2/3} + \mathrm{i}\, 3^{-1/6}}{\Gamma(\frac{2}{3})}, \\
\varepsilon \varphi'(0) = \frac{3^{-1/3} - \mathrm{i}\, 3^{1/6}}{\varepsilon^{-1/3}\Gamma(\frac{1}{3})},
\end{cases}
\tag{5.1}
$$

which results if one chooses the coefficient function $a(x) = x$. Here, Γ denotes the gamma function. The exact solution to the problem (5.1) is given by

$$
\varphi_{exact}(x) = \mathrm{Ai}(-\frac{x}{\varepsilon^{2/3}}) + \mathrm{i}\,\mathrm{Bi}(-\frac{x}{\varepsilon^{2/3}}),
$$

where Ai and Bi denote the Airy functions of first and second kind, respectively. This example demonstrates very well the advantages of a WKB method, since the solution becomes more and more oscillatory for large values of x. While standard adaptive ODE methods, e.g., Runge-Kutta methods, would need to decrease the step size more and more, a WKB-based method does not have to resolve the individual oscillations. Actually, it even allows to increase the step size for large x and is therefore highly efficient. This can be seen, e.g., in Figures 4-6.

Before starting to discuss the numerical solution of (5.1) we first remark that the *evaluation* of an oscillatory function (even of trigonometric functions

14

such as sin or cos) is numerically ill posed for very large values of x. This is a generic problem and not related to the numerical solution of (5.1), or to the choice of the numerical method:

Remark 5.1. *We consider to evaluate $\varphi_{exact}(x)$ for an argument x that is only known with finite accuracy, specified by machine eps. Then, using the lowest order expansions for φ_{exact} and φ'_{exact} from Appendix A, we have*

$$\varphi_{exact}(x) \sim \frac{\varepsilon^{1/6}}{\sqrt{\pi} x^{1/4}} \exp\left(i\left(\frac{\pi}{4} - \frac{2}{3}\frac{x^{3/2}}{\varepsilon} \right) \right) \qquad as \quad x \to \infty \,,$$

and the relative error of $\varphi_{exact}(x(1 + eps))$ is asymptotically $eps\, x^{3/2}/\varepsilon$.

Considering double precision, e.g., with MATLAB's $eps \approx 2.2 \cdot 10^{-16}$ and $x = 50$, this generic evaluation error is $7.8 \cdot 10^{-10}$ for $\varepsilon = 10^{-4}$ and $7.8 \cdot 10^{-11}$ for $\varepsilon = 10^{-3}$. These unavoidable errors limit the achievable accuracy, and it will play a role in Figures 10-11 below.

As a first step we shall now test the reliability of our choice of error estimator (3.1) – but only for the WKB steps of WKB+RKF45 and RKWKBmod. Since we know the exact solution to this problem, we can compute the local truncation error in each step and are able to compare it to the error estimator. Moreover, we apply the adaptive step size control from Section 3 with the error tolerance Tol $= 10^{-5}$ and set $\varepsilon = 1$.
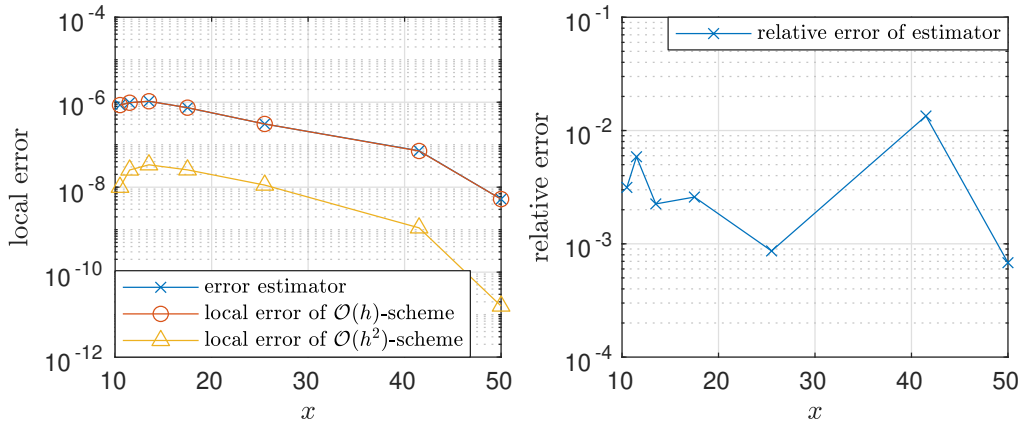
Figure 1: Left: The error estimator (3.1) in comparison to the actual local truncation error computed using only WKB steps for WKB+RKF45 of first order in h, (2.13), and second order in h, (2.14). Right: The relative error between the estimator and the local truncation error for the $\mathcal{O}(h)$-scheme. In both pictures we set Tol $= 10^{-5}$ and $\varepsilon = 1$.
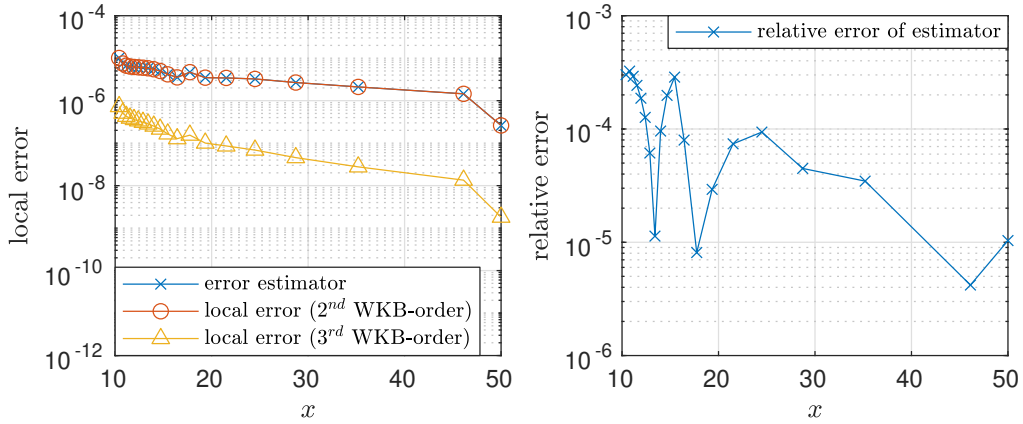
15

Figure 2: Left: The error estimator (3.1) in comparison to the actual local truncation error computed using only WKB steps in the RKWKBmod method. Since the method is of first order (in h), the estimator as well as the local errors were computed by using different WKB-orders instead (here order 2 and 3). Right: The relative error between the estimator and the local truncation error (2nd WKB-order). In both pictures we set $\text{Tol} = 10^{-5}$ and $\varepsilon = 1$.

According to the results of Figures 1 and 2, the error estimator is in excellent agreement with the local truncation error (for this example). Hence, it seems to be an adequate choice. We also find a very good agreement by plotting the respective relative errors for one single step as functions of the step size h for a fixed starting point x_0, see Figure 3. Again, the relative error is for both methods much smaller than one. Note that in the case of WKB+RKF45 the relative error goes to zero, if h tends to zero. Therefore, the estimator seems to be asymptotically correct.

16
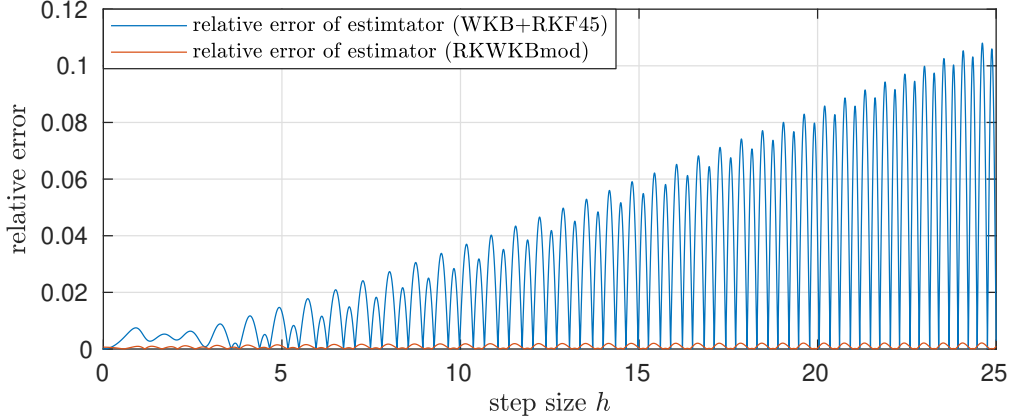
Figure 3: The relative error between the estimator (3.1) and the local truncation errors for WKB+RKF45 and RKWKBmod in one single WKB step as functions of the step size h for a fixed starting point $x_0 = 10$. Here, we set $\text{Tol} = 10^{-5}$ and $\varepsilon = 1$. For the computation of the estimator with the RKWKBmod method a 2nd and 3rd WKB-order was used.

Remark 5.2. *The computation of the local truncation error involves the evaluation of the Airy function, which seems to have an accuracy problem for large values of x when using the standard routine* `airy()` *in* MATLAB. *Hence we used a modified implementation of the Airy functions, which consists of the function* `airy()` *from* MATLAB *for small values of x and asymptotic expressions for the Airy function for large values of x. More precisely, the evaluations were performed as given in Table 2. The asymptotic expressions are based on well-known expansions, which can be found in Appendix A. The order for truncating the expansions was set to $K = 3$.*

	`airy()`	asymptotics
Ai	$x \in [0, 500]$	$x \in (500, \infty)$
Ai$'$	$x \in [0, 400]$	$x \in (400, \infty)$
Bi	$x \in [0, 500]$	$x \in (500, \infty)$
Bi$'$	$x \in [0, 400]$	$x \in (400, \infty)$

Table 2: Intervals for evaluating the Airy function of first and second kind as well as their derivatives. For small values of x the original function from MATLAB was used, and for large x the evaluation was performed using the asymptotic expressions (A.1)-(A.4) with truncation after $K = 3$.

Now, we will give numerical results for solving the initial value problem (5.1) for the Airy equation with WKB+RKF45 and RKWKBmod. We recall that the algorithm automatically chooses between RKF45 and the respective WKB steps. To illustrate this difference in the Figures 4-9 as well

17

as Figures 13-14, we will mark RKF45 steps with red dots, WKB steps using WKB+RKF45 with blue squares, and WKB steps using the RKWKBmod method with green triangles. To exclude the turning point $x = 0$, we solve the Airy equation (5.1) in Figures 4 and 5 on $[0.1, 50]$.
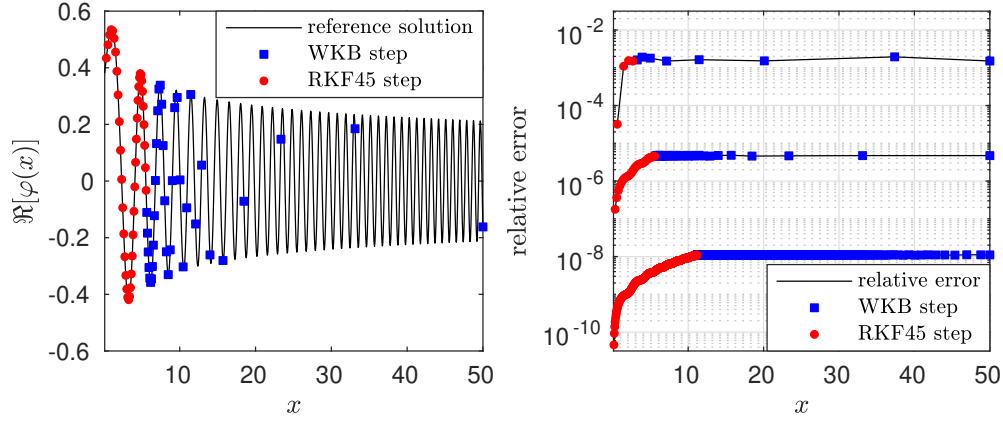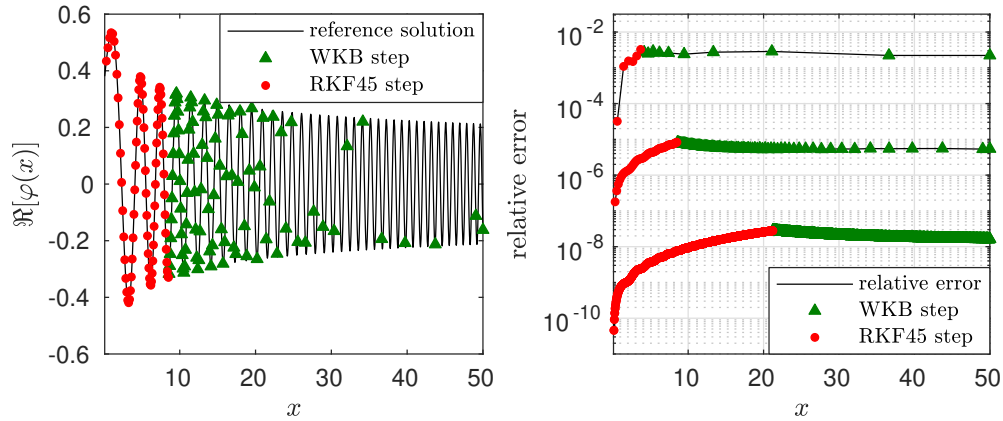

Figure 4: Left: Real part of the numerical solution obtained by using WKB+RKF45 compared to the (exact) reference solution (solid line) for Tol $= 10^{-6}$. Right: The global error for the choices Tol $= 10^{-3}, 10^{-6}, 10^{-9}$ (read from top to bottom). The respective overall number of steps made are 12, 77, and 856. For both pictures the initial step size was set to $h_{1,trial} = 0.5$ and the parameter ε was set to 1.

Figure 5: Left: Real part of the numerical solution obtained by using the RKWKBmod method compared to the (exact) reference solution (solid line) for Tol $= 10^{-6}$. Right: The global error for the choices Tol $= 10^{-3}, 10^{-6}, 10^{-9}$ (read from top to bottom). The respective overall number of steps made are 16, 171, and 2352. For both pictures the initial step size was set to $h_{1,trial} = 0.5$, the parameter ε was set to 1 and a third order WKB-ansatz was used.

According to Figures 4 and 5, WKB+RKF45 seems to perform slightly

better than the RKWKBmod method in this example, in matters of global error. But at the same time WKB+RKF45 needs significantly fewer steps than RKWKBmod. Also, within the algorithm using WKB+RKF45, the switch from RKF45 steps to WKB steps happens earlier as can be seen in Figure 4. In Figure 6, the relative global errors of both algorithms are compared on $[0.1, 10^8]$. We find that the algorithm using WKB+RKF45 made fewer steps (58 vs. 91) while producing a slightly lower global error at the same time. The almost identical grid spacing of both methods from $x \approx 300$ onwards is due to limiting the quotient of two consecutive step sizes, imposed in both methods.
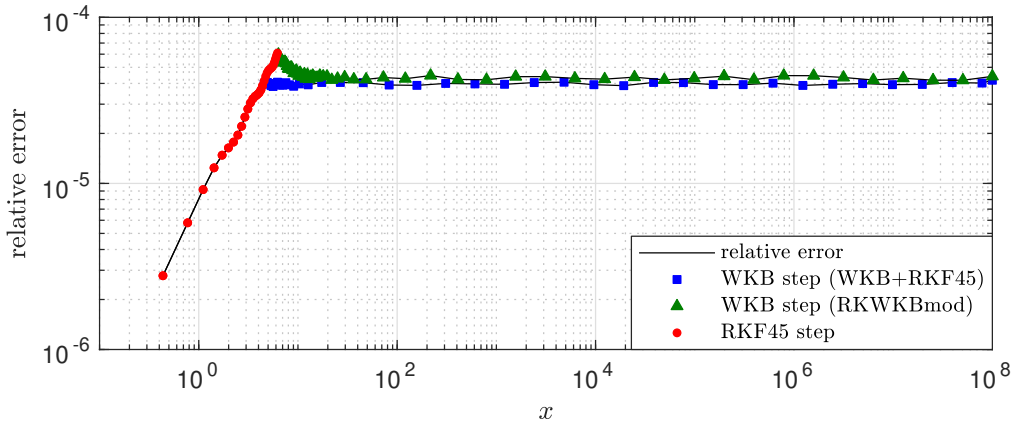
Figure 6: Global error comparison of WKB+RKF45 and RKWKBmod on $[0.1, 10^8]$ for $\varepsilon = 1$ using an initial step size $h_{1,trial} = 0.5$. The error tolerance was set to Tol $= 10^{-5}$. A third order WKB-ansatz was used for the RKWKBmod method. Overall the algorithm using WKB+RKF45 made 58 steps, whereas RKWKBmod made 91 steps.

Furthermore, Figures 7 and 8 show the ratio between the WKB- and RK-error estimators as well as the ratio of the proposed step sizes (by the WKB and RKF schemes) for each step of the above simulations.
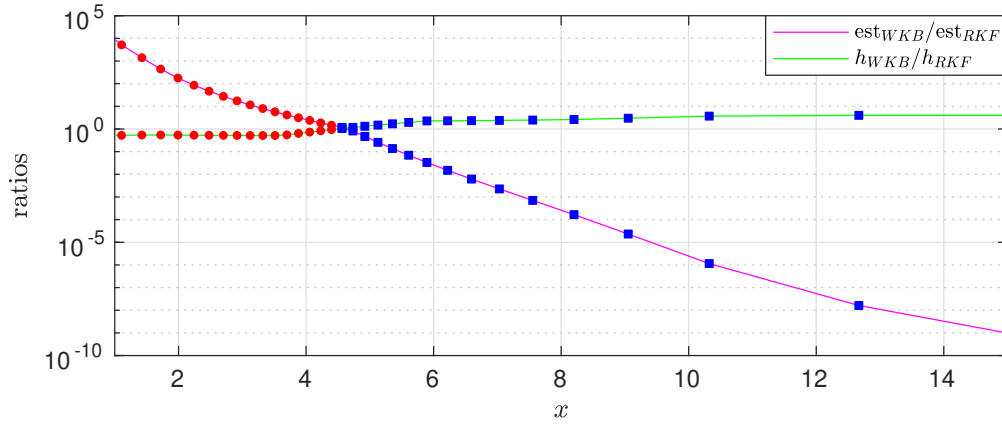
Figure 7: Numerical computation using WKB+RKF45: The magenta line indicates the ratio of the two error estimators (due to the WKB and RKF scheme). The green line gives the analogous ratio of the step sizes (locally) proposed by these two methods. For this computation we set Tol $= 10^{-5}$ and $\varepsilon = 1$.
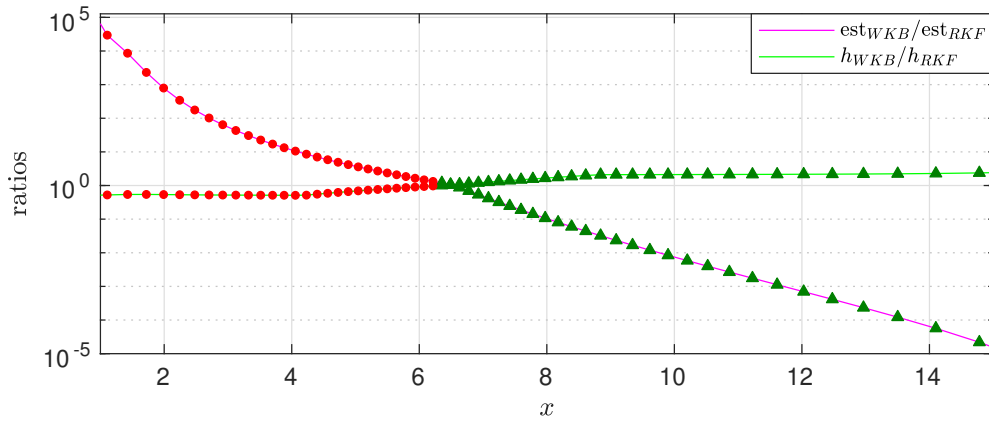

Figure 8: Numerical computation using the RKWKBmod method: The magenta line indicates the ratio of the two error estimators (due to the WKB and RKF scheme). The green line gives the analogous ratio of the step sizes (locally) proposed by these two methods. For this computation we set Tol $= 10^{-5}$, $\varepsilon = 1$ and a third order WKB-ansatz was used.

For both WKB+RKF45 and RKWKBmod, these plots demonstrate well the superiority of the RK scheme in the less oscillatory regime, i.e. for x small, as the ratio of the error estimators is very large there. The switching points to the WKB schemes are well defined (again for both WKB methods) – due to the monotonous behaviour of both the ratio of error estimators and the ratio of proposed step sizes. Note also that the step size ratios are bounded from above and below because of the limitations in (3.2).

20

We also want to give a perception of how WKB+RKF45 can perform, if the phase integral (2.7) can not be evaluated exactly. For this purpose we use the *Clenshaw-Curtis* quadrature (cf. [8]) to approximate the phase (2.7). That spectral method was already used in combination with the WKB-marching method, see [5, §5]. In [5] they approximate the phase at first on Chebyshev collocation points throughout the *whole interval* and use barycentric interpolation for the ODE-grid points x_n. This was possible since they worked only on the "small" interval $[0, 1]$. In contrast, we will instead approximate the phase (2.7) in *each interval* $[x_n, x_{n+1}]$ individually, since we are dealing here with the "long" interval $[0.1, 10^8]$. Figure 9 gives a comparison of the global error for the Airy equation (5.1), when using the exact phase vs. a numerically computed phase. According to these results the relative φ-errors from computing the phase (2.7) numerically become visible only from $x \approx 10^7$ onwards.
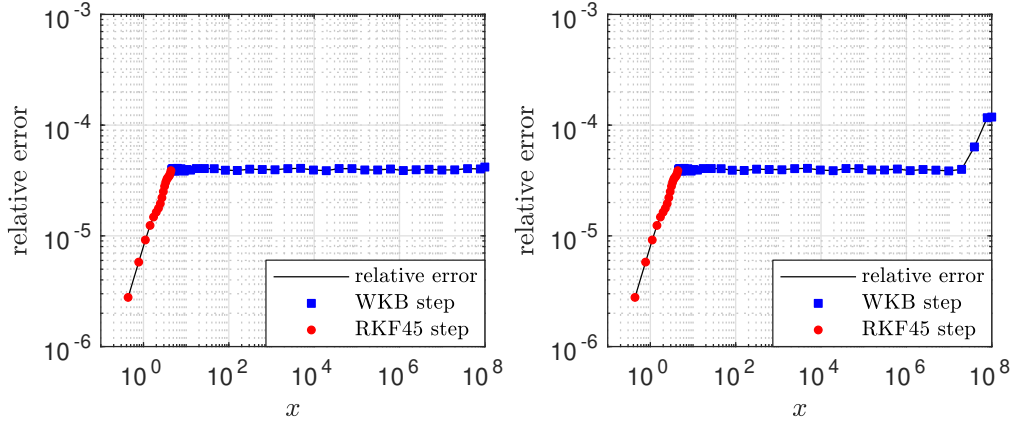
Figure 9: Left: Global (relative) error using WKB+RKF45 with exact phase. Right: Global (relative) error using WKB+RKF45 with numerically computed phase using the Clenshaw-Curtis quadrature with 15 integration nodes per step. For both computations we set $h_{1,trial} = 0.5$, Tol $= 10^{-5}$ and $\varepsilon = 1$.

We will next compare the numerical results of four methods, namely, WKB+RKF45, RKWKBmod, RKWKB, and RKF45 on the Airy equation for several values of the parameter ε. For RKF45, we do not use any smaller value of ε than 10^{-1}, as the CPU time gets exorbitantly large. In Figure 10 we compare the accuracy of each of the methods depending on Tol. We find that WKB+RKF45 outperforms the other methods, in matters of global errors, particularly for small values of ε and Tol. By Remark 5.1 the accuracy limit at $x = 50$ is $7.8 \cdot 10^{-10}$ for $\varepsilon = 10^{-4}$, which seems to explain the lower bound

21

of the errors in Figure 10. Note also that for RKF45 the error increases like $\mathcal{O}(\varepsilon^{-1})$ for smaller ε-values, while it decreases for the WKB methods.

In Figure 11 we give a work-precision diagram; for a fair comparison between the methods, points showing the same error should be compared. There is a big difference between WKB+RKF45 and RKWKB(mod), regarding the CPU time: For $\varepsilon = 10^0, 10^{-1}, 10^{-2}$ this difference is particularly significant for small errors (stemming from small prescribed tolerances). For $\varepsilon = 10^{-3}$ WKB+RKF45 already beats RKWKB(mod) for all data points. For $\varepsilon = 10^{-4}$ the error intervals of WKB+RKF45 and RKWKB(mod) do not overlap. But for the same CPU time WKB+RKF45 yields much more accurate results than RKWKB or RKWKBmod. Overall we conclude from Figure 11 that WKB+RKF45 outperforms RKWKBmod and RKWKB significantly. Using RKF45, the CPU time increases drastically for smaller ε-values.

Figure 12 displays the respective number of steps needed in each method, as a function of the prescribed tolerance. Note that for WKB+RKF45 and RKWKBmod the number of steps is bounded from below. This is because of limiting the quotient of two consecutive step sizes and it is clearly visible for $\varepsilon = 10^{-3}, 10^{-4}$.
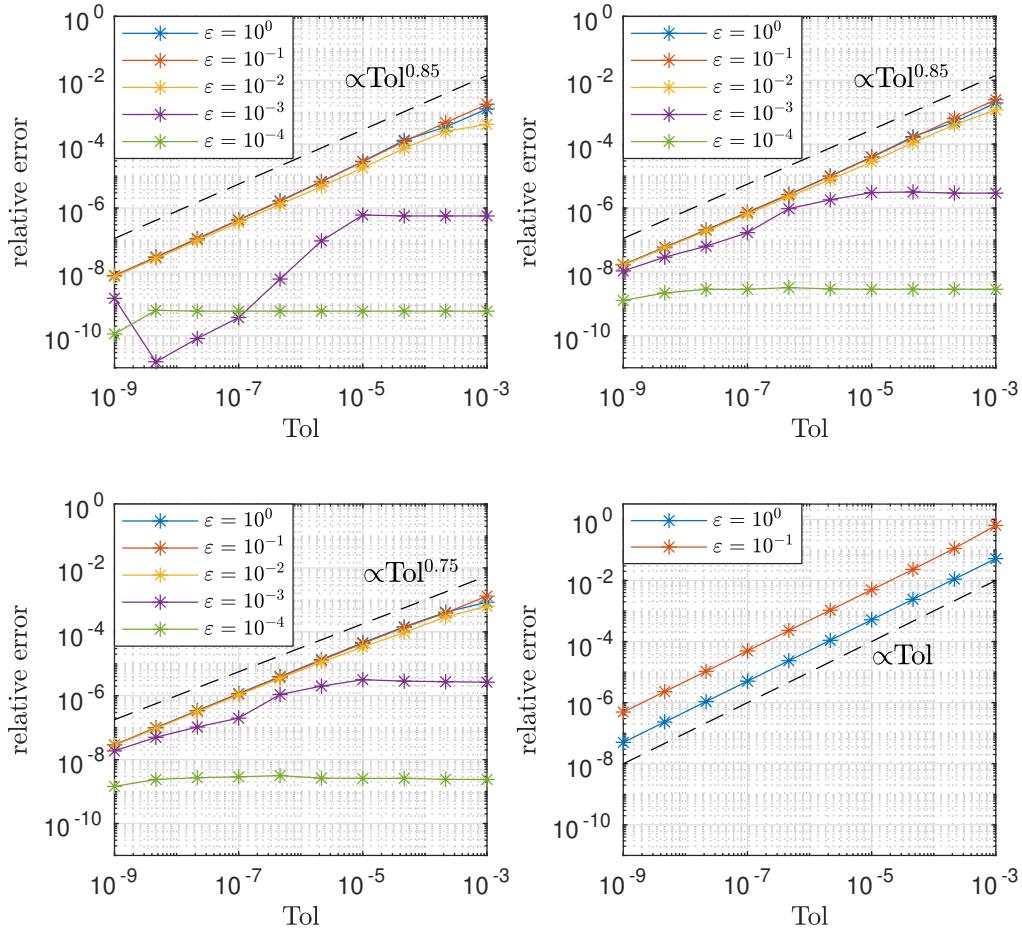
Figure 10: Global (relative) errors (in the l^2-norm) for the Airy equation (5.1) on $[0.1, 50]$ as a function of Tol, for several ε-values. Top-left: WKB+RKF45. Top-right: RKWKBmod. Bottom-left: RKWKB. Bottom-right: RKF45. For all methods we set $h_{1,trial} = 0.5$. A third order WKB-ansatz for RKWKBmod and RKWKB was used.
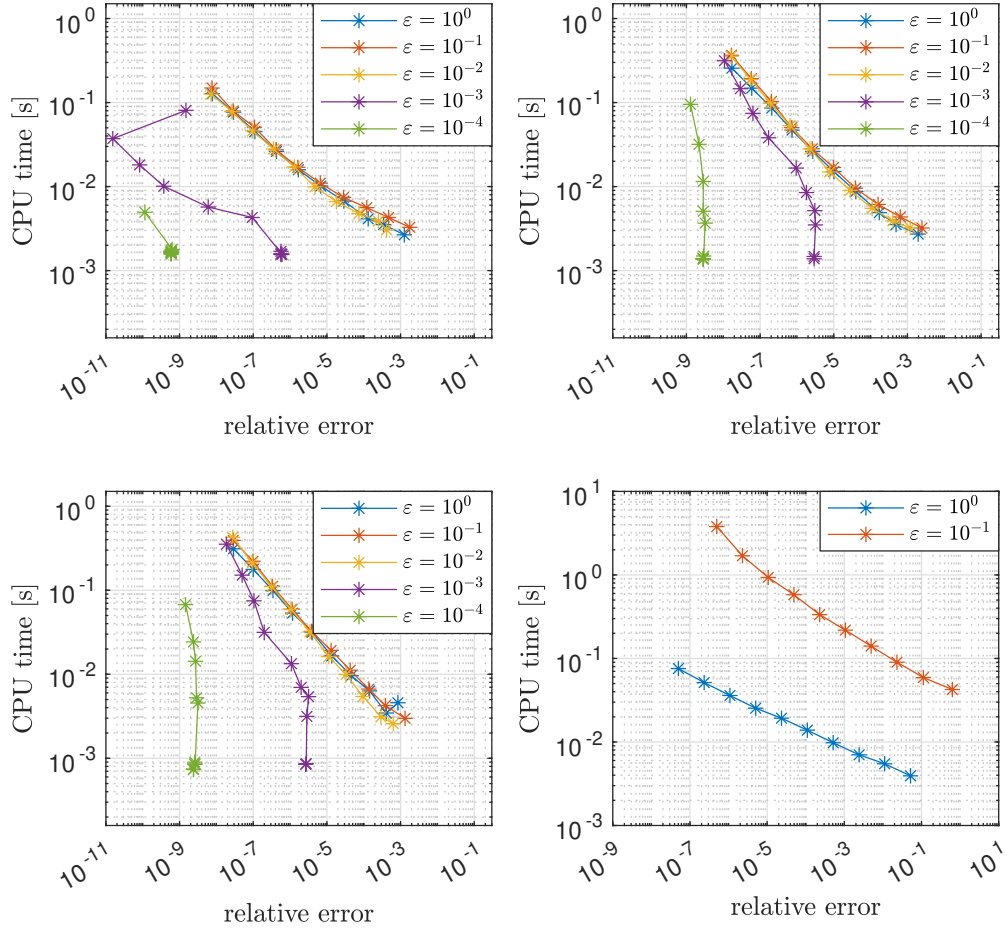
Figure 11: CPU times vs. global (relative) errors (in the l^2-norm) for the Airy equation (5.1) on $[0.1, 50]$, computed for 10 logarithmically evenly spaced values of Tol in the range $[10^{-9}, 10^{-3}]$, for several ε-values. Top-left: WKB+RKF45. Top-right: RKWKBmod. Bottom-left: RKWKB. Bottom-right: RKF45 (note the different scales). For all methods we set $h_{1,trial} = 0.5$. A third order WKB-ansatz for RKWKBmod and RKWKB was used.
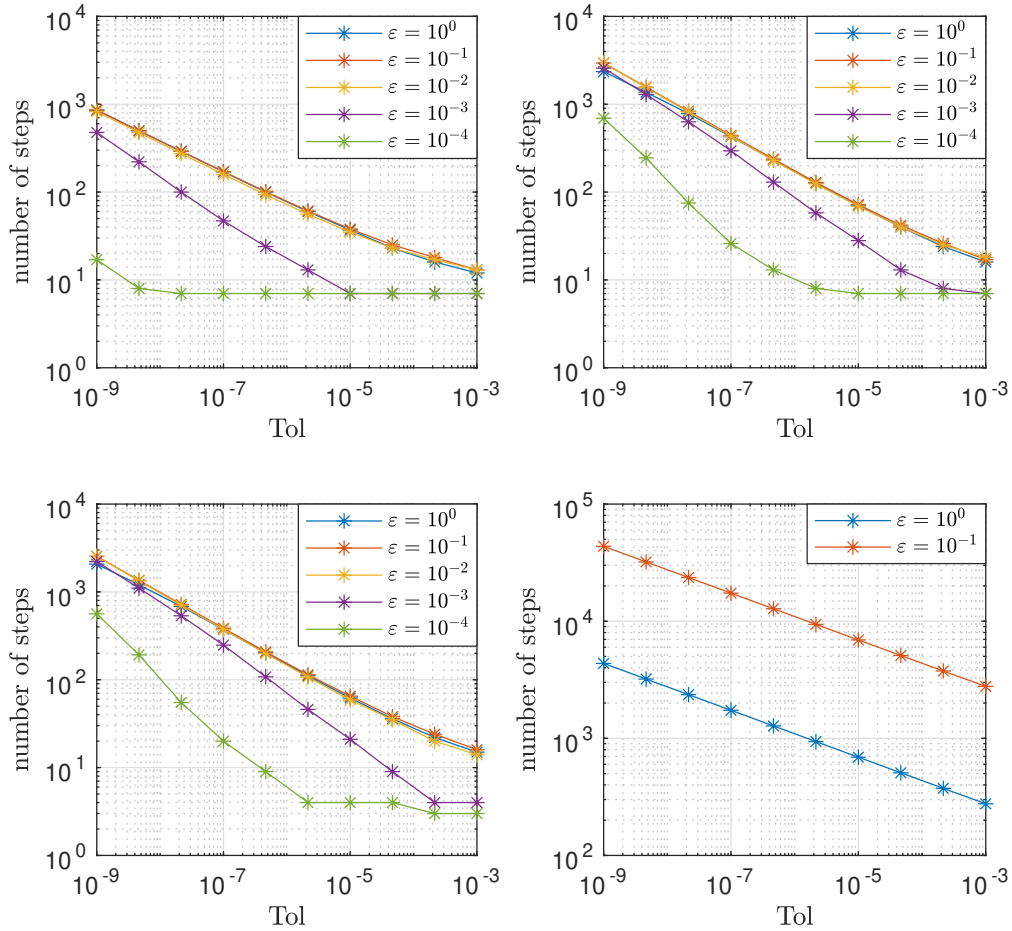
Figure 12: Number of steps needed for the Airy equation (5.1) on $[0.1, 50]$ as a function of Tol, for several ε-values. Top-left: WKB+RKF45. Top-right: RKWKBmod. Bottom-left: RKWKB. Bottom-right: RKF45. For all methods we set $h_{1,trial} = 0.5$. A third order WKB-ansatz for RKWKBmod and RKWKB was used.

5.2. Second Example: Parabolic cylinder function

The second example is taken from [3] and includes a quadratic coefficient function $a(x)$:

$$\begin{cases} \varepsilon^2 \varphi''(x) + \left(-\frac{1}{2}x^2 + x\right) \varphi(x) = 0, & x \in (0, 2), \\ \varphi(0) = \kappa U(\nu, z(0)), \\ \varphi'(0) = -\kappa 2^{-\frac{1}{4}} \varepsilon^{-\frac{1}{2}} U'(\nu, z(0)), \end{cases} \quad (5.2)$$

25

with

$$\nu := -\frac{1}{\sqrt{8}\varepsilon} \,, \quad z(x) := \frac{2^{\frac{1}{4}}}{\sqrt{\varepsilon}}(1-x) \,,$$

and

$$\kappa := \frac{2}{U(\nu,0) - \mathrm{i}\,\sqrt{\varepsilon}2^{\frac{3}{4}}U'(\nu,0)} \,.$$

The exact solution reads

$$\varphi_{exact}(x) = \kappa U(\nu, z(x)) \,,$$

where $U(\nu, z)$ denotes the parabolic cylinder function (PCF) (cf. [16, §12]). As before, let us compare numerical results only for WKB+RKF45 and RKWKBmod at first. There are two turning points, namely at $x = 0$ and $x = 2$. Therefore, we expect the two methods to make RKF45 steps near the turning points and WKB steps between them. Numerical results for the specific choice $\varepsilon = 2^{-6}$ are presented in Figures 13-14.
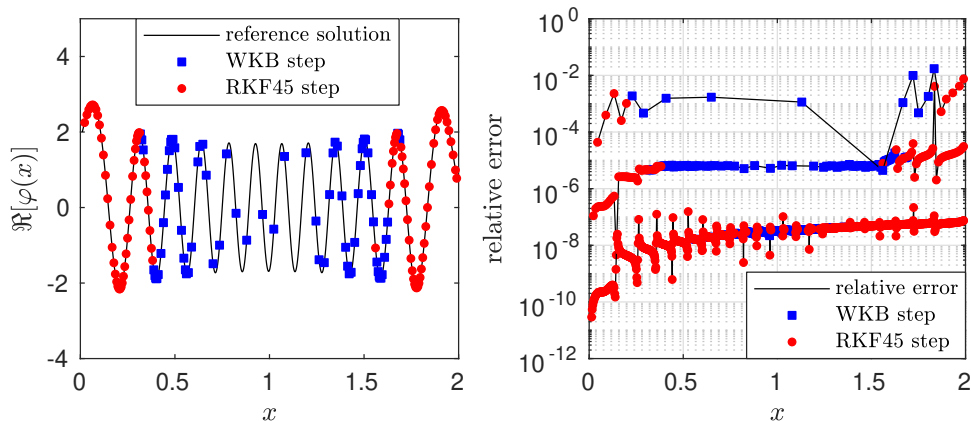

Figure 13: Left: Real part of the numerical solution obtained by using WKB+RKF45 compared to the (exact) reference solution (solid line) for Tol $= 10^{-6}$. Right: The global error for the choices Tol $= 10^{-3}, 10^{-6}, 10^{-9}$ (read from top to bottom). The respective overall number of steps made are 21, 166, and 1287. For both pictures the initial step size was set to $h_{1,trial} = 0.05$ and the parameter ε was set to 2^{-6}.
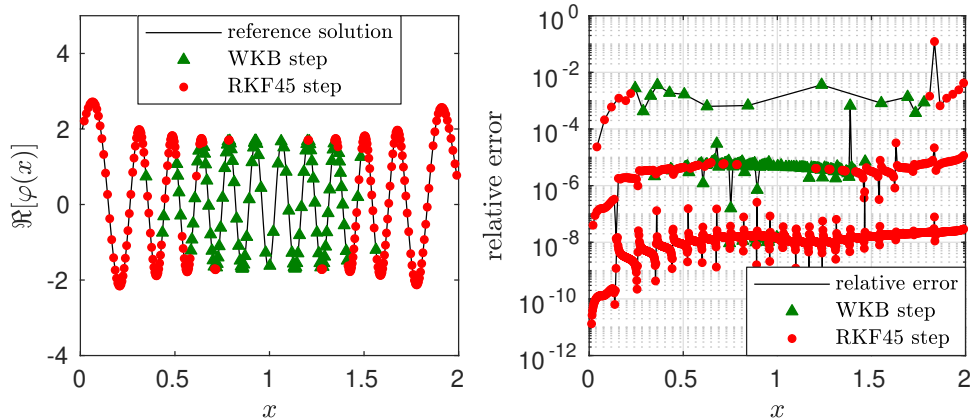
Figure 14: Left: Real part of the numerical solution obtained by using RKWKBmod compared to the (exact) reference solution (solid line) for Tol $= 10^{-6}$. Right: The global error for the choices Tol $= 10^{-3}, 10^{-6}, 10^{-9}$ (read from top to bottom). The respective overall number of steps made are 26, 326, and 1543. For both pictures the initial step size was set to $h_{1,trial} = 0.05$, the parameter ε was set to 2^{-6} and a third order WKB-ansatz was used.

According to Figures 13 and 14 no significant difference between WKB+RKF45 and RKWKBmod can be observed for $\varepsilon = 2^{-6}$, in matters of global error. But again, WKB+RKF45 needs significantly fewer steps than RKWKBmod. Within the algorithm using WKB+RKF45, the switch-over between RKF45 steps and WKB steps happens closer to the turning points.

We shall now present numerical results for the four methods WKB+RKF45, RKWKBmod, RKWKB, and RKF45 for several values of ε. In Figure 15, we compare the global errors of each method depending on Tol. Here, we observe only a small difference between WKB+RKF45 and RKWKBmod. Using RKWKB, less smooth error curves can bee seen, with large peaks for lower errors (i.e. lower tolerances). In contrast to every other method, WKB+RKF45 seems to produce quite ε-independent global error curves. For RKF45 one sees that the error again increases like $\mathcal{O}(\varepsilon^{-1})$ for smaller values of ε.

In the work-precision diagrams in Figure 16, we observe for WKB+RKF45 that the CPU times are quite independent of ε, whereas for RKWKB(mod) they grow with decreasing ε, particularly for small errors (stemming from small prescribed tolerances). Overall we conclude from Figure 16 that WKB+RKF45 outperforms RKWKBmod and RKWKB (particularly for small ε and small tolerances) while showing an ε-uniform behaviour at the same time. Note also that, for RKF45, the CPU time increases drastically for smaller values of ε.

27

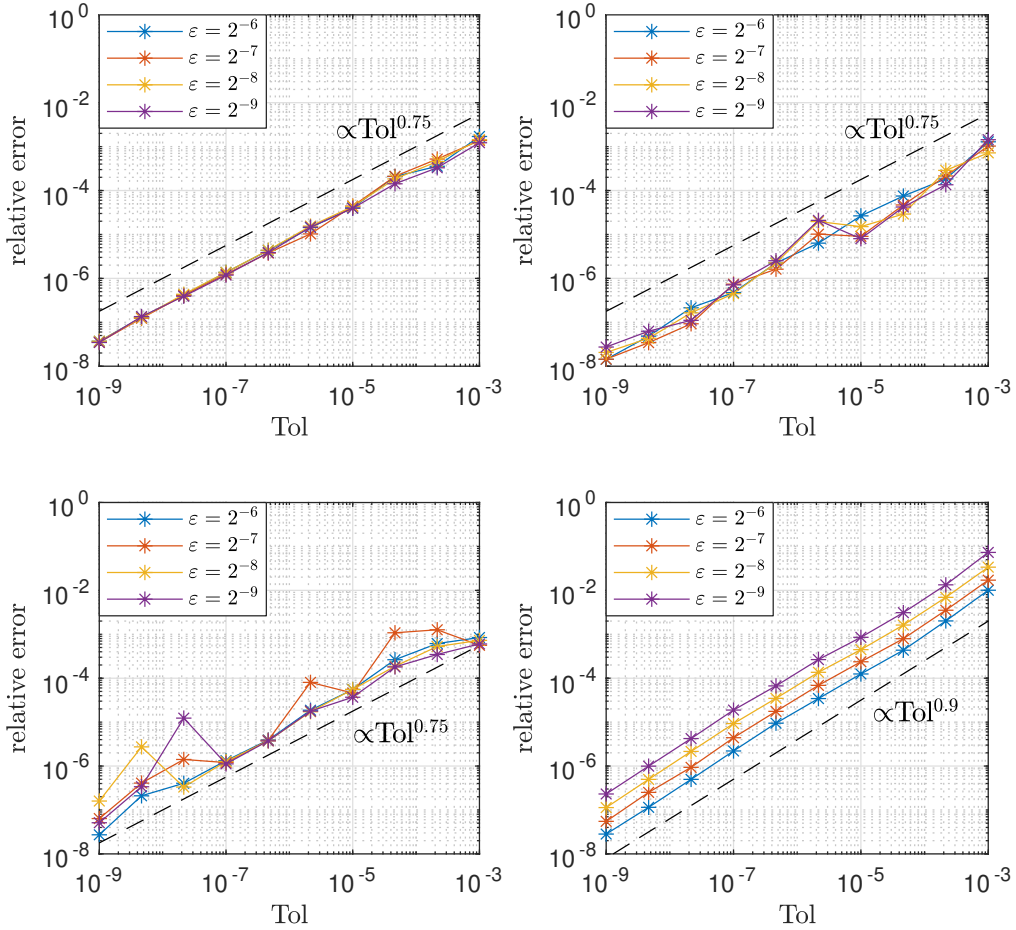Figure 17 shows the respective number of steps needed in each method, as a function of the prescribed tolerance.

Figure 15: Global (relative) errors (in the l^2-norm) for equation (5.2) on $[0.01, 1.99]$ as a function of Tol, for several ε-values. Top-left: WKB+RKF45. Top-right: RKWKBmod. Bottom-left: RKWKB. Bottom-right: RKF45. For all methods we set $h_{1,trial} = 0.05$. A third order WKB-ansatz for RKWKBmod and RKWKB was used.
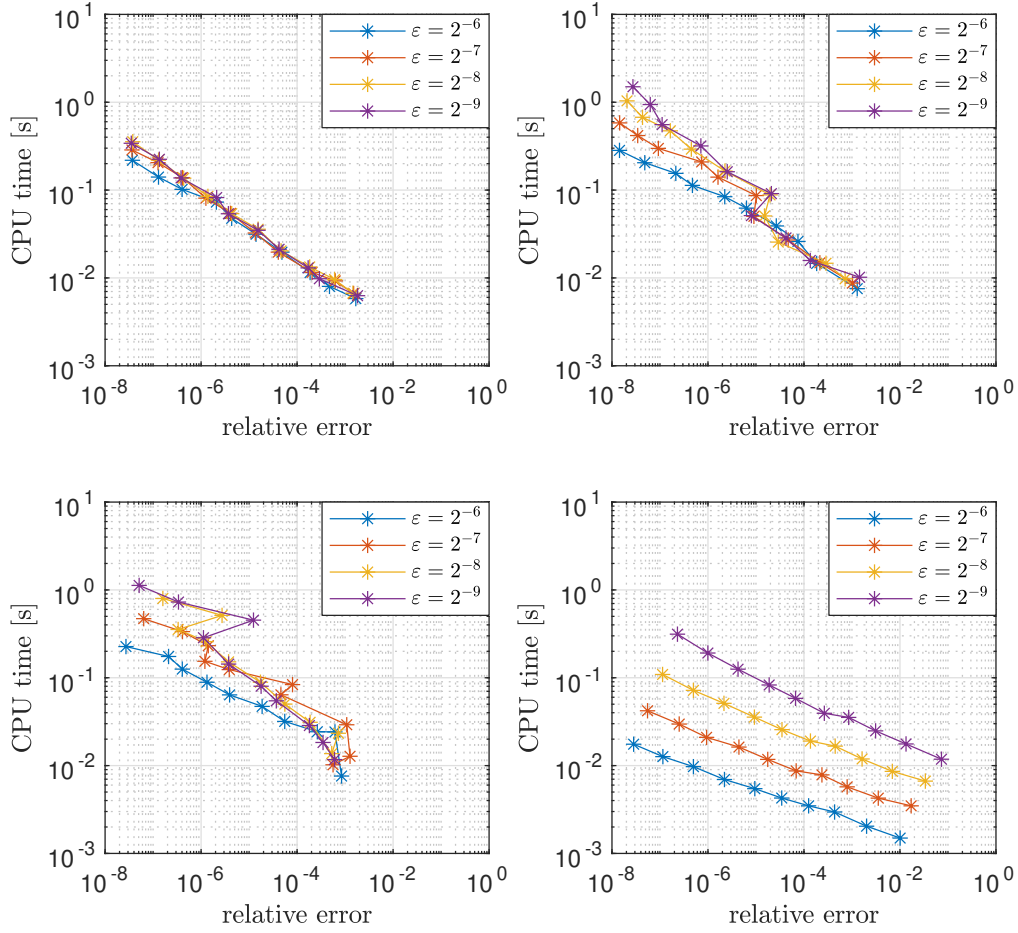
Figure 16: CPU times vs. global (relative) errors (in the l^2-norm) for equation (5.2) on $[0.01, 1.99]$, computed for 10 logarithmically evenly spaced values of Tol in the range $[10^{-9}, 10^{-3}]$, for several ε-values. Top-left: WKB+RKF45. Top-right: RKWKBmod. Bottom-left: RKWKB. Bottom-right: RKF45. For all methods we set $h_{1,trial} = 0.05$. A third order WKB-ansatz for RKWKBmod and RKWKB was used.
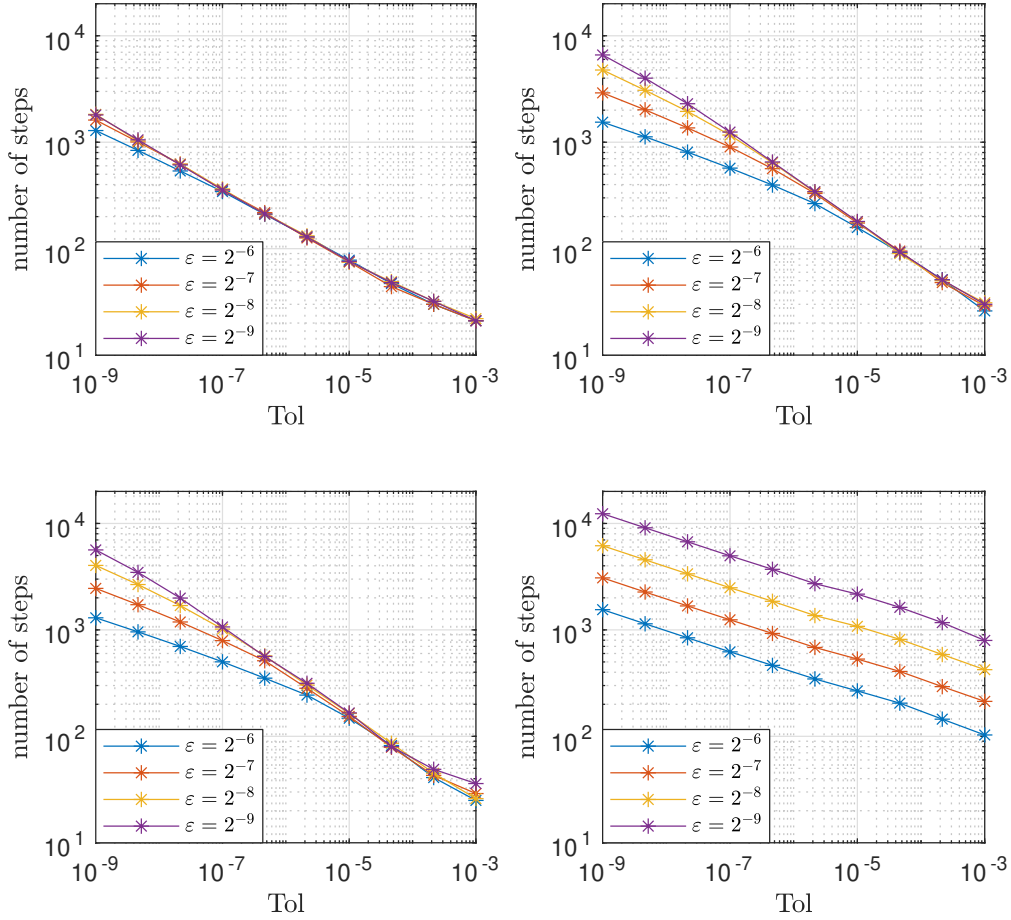
Figure 17: Number of steps needed for equation (5.2) on [0.01, 1.99] as a function of Tol, for several ε-values. Top-left: WKB+RKF45. Top-right: RKWKBmod. Bottom-left: RKWKB. Bottom-right: RKF45. For all methods we set $h_{1,trial} = 0.05$. A third order WKB-ansatz for RKWKBmod and RKWKB was used.

Remark 5.3. *The computation of the reference solution involves the evaluation of the PCF, which is not readily available in MATLAB. But the PCF can be related to the Kummer confluent hypergeometric function $_1F_1$ (see [16, §13]), which is available in MATLAB as* `hypergeom()`. *However, for small parameters ε this evaluation is very time consuming. For Figures 15-17 we thus computed the reference solutions by solving the initial value problem (5.2) with MATLAB's routine* `ode45()` *(for all ε) using a very small error tolerance.*

6. Conclusion

We have introduced in this paper an extension to the WKB-marching method from [2] by including into the algorithm an adaptive step size controller as well as a switching mechanism. In numerical simulations based on two examples this method yielded smaller global errors (particularly for small tolerances and small ε-values) in comparison to the Runge-Kutta-WKB method from [9, 4], an alternative WKB-based scheme. Our tests revealed that the efficiency gain is mostly due to the different WKB method used here, while the different step size controls (here vs. [9, 4]) do not play a big role. Our switching mechanism ensures well defined switching points between WKB steps and Runge-Kutta-Fehlberg 4(5) steps for oscillatory and, respectively, smoother regions of the ODE-solution. Especially for the Airy equation on the large spatial interval $[0.1, 10^8]$ the efficiency of the method is demonstrated very well, as the scheme skips millions of oscillations within one step, while staying accurate at the same time. There is also a MATLAB program available in a GitHub repository[1], which also offers the possibility to compute the phase (2.7) numerically, as done for Figure 9.

---

[1]https://github.com/JannisKoerner/adaptive-WKB-marching-method

Appendix A. Asymptotic formulas for Airy functions

For real-valued x and $x \to \infty$, asymptotic expansions for the Airy functions and their first derivatives are given in [16, §9.7 (ii)]:

$$\mathrm{Ai}(-x) \sim \frac{1}{\sqrt{\pi}x^{\frac{1}{4}}} \left(\cos\left(\zeta - \frac{\pi}{4}\right) \sum_{k=0}^{\infty} (-1)^k \frac{u_{2k}}{\zeta^{2k}} + \sin\left(\zeta - \frac{\pi}{4}\right) \sum_{k=0}^{\infty} (-1)^k \frac{u_{2k+1}}{\zeta^{2k+1}} \right) ,$$

$$\tag{A.1}$$

$$\mathrm{Ai}'(-x) \sim \frac{x^{\frac{1}{4}}}{\sqrt{\pi}} \left(\sin\left(\zeta - \frac{\pi}{4}\right) \sum_{k=0}^{\infty} (-1)^k \frac{v_{2k}}{\zeta^{2k}} - \cos\left(\zeta - \frac{\pi}{4}\right) \sum_{k=0}^{\infty} (-1)^k \frac{v_{2k+1}}{\zeta^{2k+1}} \right) ,$$

$$\tag{A.2}$$

$$\mathrm{Bi}(-x) \sim \frac{1}{\sqrt{\pi}x^{\frac{1}{4}}} \left(-\sin\left(\zeta - \frac{\pi}{4}\right) \sum_{k=0}^{\infty} (-1)^k \frac{u_{2k}}{\zeta^{2k}} + \cos\left(\zeta - \frac{\pi}{4}\right) \sum_{k=0}^{\infty} (-1)^k \frac{u_{2k+1}}{\zeta^{2k+1}} \right) ,$$

$$\tag{A.3}$$

$$\mathrm{Bi}'(-x) \sim \frac{x^{\frac{1}{4}}}{\sqrt{\pi}} \left(\cos\left(\zeta - \frac{\pi}{4}\right) \sum_{k=0}^{\infty} (-1)^k \frac{v_{2k}}{\zeta^{2k}} + \sin\left(\zeta - \frac{\pi}{4}\right) \sum_{k=0}^{\infty} (-1)^k \frac{v_{2k+1}}{\zeta^{2k+1}} \right) .$$

$$\tag{A.4}$$

Here, $\zeta := \frac{2}{3}x^{\frac{3}{2}}$ and the coefficients u_k and v_k are given by (see [16, §9.7 (i)]):

$$u_0 = v_0 = 1 ,$$
$$u_k = \frac{(2k+1) \cdot (2k+3) \cdot (2k+5) \cdot \ldots \cdot (6k-1)}{216^k k!}$$
$$= \frac{(6k-5) \cdot (6k-3) \cdot (6k-1)}{(2k-1)216k} u_{k-1} , \quad k = 1, 2, \ldots ,$$
$$v_k = \frac{6k+1}{1-6k} u_k , \quad k = 1, 2, \ldots .$$

References

[1] P. Amodio, L. Brugnano, F. Iavernaro, *Analysis of spectral Hamiltonian boundary value methods (SHBVMs) for the numerical solution of ODE problems*, Numerical Algorithms 83 (2020), no. 4, 1489–1508.

[2] A. Arnold, N. Ben Abdallah, C. Negulescu, *WKB-based schemes for the oscillatory 1D Schrödinger equation in the semi-classical limit*, SIAM J. Numer. Anal. 49 (2011), no. 4, pp. 1436-1460.

[3] A. Arnold, K. Döpfner, *Stationary Schrödinger equation in the semi-classical limit: WKB-based scheme coupled to a turning point*, Calcolo 57, No. 1 (2020) 44 pp.

[4] F. J. Agocs, W. J. Handley, A. N. Lasenby, and M. P. Hobson, *An efficient method for solving highly oscillatory ordinary differential equations with applications to physical systems*, Phys. Rev. Res. 2 , 013030 (2020), arXiv:1906.01421.

[5] A. Arnold, C. Klein, B. Ujvari, *WKB-method for the 1D Schrödinger equation in the semi-classical limit: enhanced phase treatment*, submitted, (2019). arxiv.org/abs/1808.01887

[6] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*, Wiley-Interscience, Second Edition, New York, 2008

[7] L. Brugnano, F, Iavernaro, J. I. Montijano, L. Rández, *Spectrally accurate space-time solution of Hamiltonian PDEs*, Numerical Algorithms (2019), 81:1183–1202

[8] C. W. Clenshaw, A. R. Curtis *A method for numerical integration on an automatic computer*, Numerische Mathathik 2, 197-205 (1960)

[9] W. J. Handley, A. N. Lasenby, and M. P. Hobson, *The Runge-Kutta-Wentzel-Kramers-Brillouin method*, arXiv:1612.02288, preprint 2016.

[10] E. Hairer, C. Lubich, G. Wanner, *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, 2nd Ed., Springer-Verlag, Berlin Heidelberg, 2006.

[11] E. Hairer, S. Nørsett, G. Wanner, *Solving Ordinary Differential Equations I*, Springer, Berlin, 2000

[12] A. Iserles, S.P. Norsett, S. Olver, *Highly oscillatory quadrature: The story so far*, in A. Bermudez de Castro, ed., *Proceeding of ENuMath, Santiago de Compostella (2006)*, Springer Verlag, (2006), pp. 97-118.

[13] T. Jahnke, C. Lubich, *Numerical integrators for quantum dynamics close to the adiabatic limit*, Numerische Mathematik, 94 (2003), pp. 289-314.

[14] K. Lorenz, T. Jahnke, C. Lubich, *Adiabatic integrators for highly oscillatory second-order linear differential equations with time-varying eigendecomposition*, BIT, 45 (2005), no. 1, pp. 91-115.

[15] L. D. Landau, E. M. Lifschitz, *Quantenmechanik*, Akademie-Verlag, Berlin, 1985

[16] F. W. Olver, D. W. Lozier, R. F. Boisvert, C. W. Clark, *NIST handbook of mathematical functions*, Cambridge University Press, 2010

[17] G. Söderlind, *Automatic Control and Adaptive Time-Stepping*, Numerical Algorithms. 31., pp. 281-310, 2002

[18] X. Wu, B. Wang, *Recent Developments in Structure-Preserving Algorithms for Oscillatory Differential Equations*, Springer, Singapore (2018)