An adaptive spectral method for oscillatory second-order linear ODEs with frequency-independent cost

Fruzsina Agocs[1] with Alex Barnett[1]

[1]Center for Computational Mathematics, Flatiron Institute, Simons Foundation

ICIAM Tokyo, August 2023

arxiv:2212.06924 (accepted to SINUM)

Acknowledgements

Thanks to:

			
Alex Barnett	Manas Rachh	Jim Bremer	Charlie Epstein

The problem

- Interested in solving the initial value problem

$$
\begin{aligned}
&u''(t) + 2\gamma(t)u'(t) + \omega^2(t)u(t) = 0, \quad t \in [t_0, t_1] \\
&\text{with} \quad u(t_0) = u_0, \quad u'(t_0) = u_0'.
\end{aligned}
$$

The problem

- Interested in solving the initial value problem

$$
\begin{aligned}
&u''(t) + 2\gamma(t)u'(t) + \omega^2(t)u(t) = 0, \quad t \in [t_0, t_1] \\
&\text{with} \quad u(t_0) = u_0, \quad u'(t_0) = u_0'.
\end{aligned}
$$

- $\omega(t), \gamma(t)$ real-valued and $\omega(t) \geq 0$

The problem

- Interested in solving the initial value problem

$$
\begin{aligned}
&u''(t) + 2\gamma(t)u'(t) + \omega^2(t)u(t) = 0, \quad t \in [t_0, t_1] \\
&\text{with} \quad u(t_0) = u_0, \quad u'(t_0) = u'_0.
\end{aligned}
$$

- $\omega(t), \gamma(t)$ real-valued and $\omega(t) \geq 0$
- When $\omega \gg 1$, $u(t)$ is oscillatory, conventional ODE solvers need discretization with $\mathcal{O}(\omega)$ steps $\rightarrow$ slow,

The problem

- Interested in solving the initial value problem

$$\begin{aligned} &u''(t) + 2\gamma(t)u'(t) + \omega^2(t)u(t) = 0, \quad t \in [t_0, t_1] \\ &\text{with} \quad u(t_0) = u_0, \quad u'(t_0) = u'_0. \end{aligned}$$

- $\omega(t), \gamma(t)$ real-valued and $\omega(t) \geq 0$
- When $\omega \gg 1$, $u(t)$ is oscillatory, conventional ODE solvers need discretization with $\mathcal{O}(\omega)$ steps $\rightarrow$ slow,
- Some efficient numerical solvers exist[1] (more about them later)

[1] **Agocs**, Handley, et al., *Phys Rev Research* (2020), Bremer, *ACHA* (2018), Bremer, *ACHA* (2023), Körner et al., *JCAM* (2022), Petzold, *SINUM* (1981) (`oscode`)

The problem

- Interested in solving the initial value problem

$$\begin{aligned} &u''(t) + 2\gamma(t)u'(t) + \omega^2(t)u(t) = 0, \quad t \in [t_0, t_1] \\ &\text{with} \quad u(t_0) = u_0, \quad u'(t_0) = u'_0. \end{aligned}$$

- $\omega(t), \gamma(t)$ real-valued and $\omega(t) \geq 0$
- When $\omega \gg 1$, $u(t)$ is oscillatory, conventional ODE solvers need discretization with $\mathcal{O}(\omega)$ steps $\rightarrow$ slow,
- Some efficient numerical solvers exist[1] (more about them later)
- But none have all of the following properties:

[1] **Agocs**, Handley, et al., *Phys Rev Research* (2020), Bremer, *ACHA* (2018), Bremer, *ACHA* (2023), Körner et al., *JCAM* (2022), Petzold, *SINUM* (1981) (oscode)

The problem

- Interested in solving the initial value problem

$$
\begin{aligned}
&u''(t) + 2\gamma(t)u'(t) + \omega^2(t)u(t) = 0, \quad t \in [t_0, t_1] \\
&\text{with} \quad u(t_0) = u_0, \quad u'(t_0) = u'_0.
\end{aligned}
$$

- $\omega(t), \gamma(t)$ real-valued and $\omega(t) \geq 0$
- When $\omega \gg 1$, $u(t)$ is oscillatory, conventional ODE solvers need discretization with $\mathcal{O}(\omega)$ steps $\rightarrow$ slow,
- Some efficient numerical solvers exist[1] (more about them later)
- But none have all of the following properties:
  - Efficient when $\omega \gg 1$ **or** $\omega \lessapprox 1$ (solution is oscillatory or non-oscillatory),

[1] **Agocs**, Handley, et al., *Phys Rev Research* (2020), Bremer, *ACHA* (2018), Bremer, *ACHA* (2023), Körner et al., *JCAM* (2022), Petzold, *SINUM* (1981) (`oscode`)

The problem

- Interested in solving the initial value problem

$$\begin{aligned} &u''(t) + 2\gamma(t)u'(t) + \omega^2(t)u(t) = 0, \quad t \in [t_0, t_1] \\ &\text{with} \quad u(t_0) = u_0, \quad u'(t_0) = u_0'. \end{aligned}$$

- $\omega(t), \gamma(t)$ real-valued and $\omega(t) \geq 0$
- When $\omega \gg 1$, $u(t)$ is oscillatory, conventional ODE solvers need discretization with $\mathcal{O}(\omega)$ steps $\rightarrow$ slow,
- Some efficient numerical solvers exist[1] (more about them later)
- But none have all of the following properties:
  - Efficient when $\omega \gg 1$ **or** $\omega \lessapprox 1$ (solution is oscillatory or non-oscillatory),
  - Works in the more general case of $\gamma(t) \neq 0$,

[1] **Agocs**, Handley, et al., *Phys Rev Research* (2020), Bremer, *ACHA* (2018), Bremer, *ACHA* (2023), Körner et al., *JCAM* (2022), Petzold, *SINUM* (1981) (oscode)

The problem

- Interested in solving the initial value problem

$$\begin{aligned} &u''(t) + 2\gamma(t)u'(t) + \omega^2(t)u(t) = 0, \quad t \in [t_0, t_1] \\ &\text{with} \quad u(t_0) = u_0, \quad u'(t_0) = u'_0. \end{aligned}$$

- $\omega(t), \gamma(t)$ real-valued and $\omega(t) \geq 0$
- When $\omega \gg 1$, $u(t)$ is oscillatory, conventional ODE solvers need discretization with $\mathcal{O}(\omega)$ steps $\rightarrow$ slow,
- Some efficient numerical solvers exist[1] (more about them later)
- But none have all of the following properties:
  - Efficient when $\omega \gg 1$ **or** $\omega \lessapprox 1$ (solution is oscillatory or non-oscillatory),
  - Works in the more general case of $\gamma(t) \neq 0$,
  - Is arbitrarily high-order.

[1] **Agocs**, Handley, et al., *Phys Rev Research* (2020), Bremer, *ACHA* (2018), Bremer, *ACHA* (2023), Körner et al., *JCAM* (2022), Petzold, *SINUM* (1981) (`oscode`)

Method overview

- Time-stepping with adaptive stepsize, keep local error below tolerance ε

Method overview

- Time-stepping with adaptive stepsize, keep local error below tolerance ε
- Right strategy is to exploit known properties/behavior of the solution

Method overview

- Time-stepping with adaptive stepsize, keep local error below tolerance ε
- Right strategy is to exploit known properties/behavior of the solution
- Two different methods for when $u(t)$ oscillatory and slowly-varying,
  - $\omega \lessapprox 1$: Spectral collocation method based on Chebyshev nodes, "Chebyshev/spectral method"

Method overview

- Time-stepping with adaptive stepsize, keep local error below tolerance ε
- Right strategy is to exploit known properties/behavior of the solution
- Two different methods for when $u(t)$ oscillatory and slowly-varying,
  - $\omega \lessapprox 1$: Spectral collocation method based on Chebyshev nodes, "Chebyshev/spectral method"
  - $\omega \gg 1$: Asymptotic expansion of nonoscillatory phase function, "Riccati/asymptotic method"

Method overview

- Time-stepping with adaptive stepsize, keep local error below tolerance ε
- Right strategy is to exploit known properties/behavior of the solution
- Two different methods for when $u(t)$ oscillatory and slowly-varying,
  - $\omega \lessapprox 1$: Spectral collocation method based on Chebyshev nodes, "Chebyshev/spectral method"
  - $\omega \gg 1$: Asymptotic expansion of nonoscillatory phase function, "Riccati/asymptotic method"
- Automatic **switching** between the methods

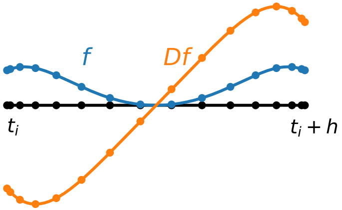
Spectral collocation on Chebyshev nodes

- Timestepping from t_i to $t_{i+1} = t_i + h$

Spectral collocation on Chebyshev nodes

- Timestepping from t_i to $t_{i+1} = t_i + h$
- Discretize ODE[2]over $[t_i, t_i + h]$ via an n-point Chebyshev grid:

$$\left[\mathrm{D}^2 + \mathrm{diag}(\omega^2(\mathbf{t}))\right]\mathbf{u} = 0$$

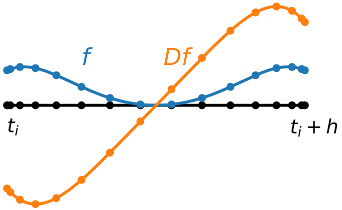


[2]We set $\gamma(t) = 0$ for simplicity.

Spectral collocation on Chebyshev nodes

- Timestepping from t_i to $t_{i+1} = t_i + h$
- Discretize ODE[2]over $[t_i, t_i + h]$ via an n-point Chebyshev grid:

$$\left[\mathrm{D}^2 + \mathrm{diag}(\omega^2(\mathbf{t}))\right]\mathbf{u} = 0$$



- To this $n \times n$ system, add two rows encoding initial conditions:

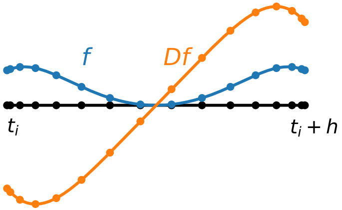
$$[1, 0, 0, 0, \ldots]\,\mathbf{u} = u_i$$
$$[\text{ first row of } \mathrm{D}\ ]\,\mathbf{u} = u_i'$$

[2]We set $\gamma(t) = 0$ for simplicity.

Spectral collocation on Chebyshev nodes

- Timestepping from t_i to $t_{i+1} = t_i + h$
- Discretize ODE[2]over $[t_i, t_i + h]$ via an n-point Chebyshev grid:

$$\left[\mathrm{D}^2 + \mathrm{diag}(\omega^2(\mathbf{t}))\right]\mathbf{u} = 0$$



- To this $n \times n$ system, add two rows encoding initial conditions:

$$[1, 0, 0, 0, \ldots]\,\mathbf{u} = u_i$$
$$[\text{ first row of } \mathrm{D}\ ]\,\mathbf{u} = u_i'$$

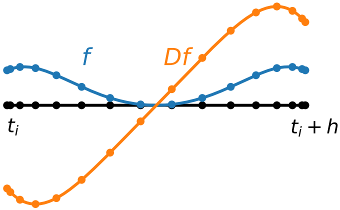
- Solve the system (least sq)

[2]We set $\gamma(t) = 0$ for simplicity.

Spectral collocation on Chebyshev nodes

- Timestepping from t_i to $t_{i+1} = t_i + h$
- Discretize ODE[2] over $[t_i, t_i + h]$ via an n-point Chebyshev grid:

$$\left[\mathrm{D}^2 + \mathrm{diag}(\omega^2(\mathbf{t}))\right] \mathbf{u} = 0$$



- To this $n \times n$ system, add two rows encoding initial conditions:

$$[1, 0, 0, 0, \ldots]\, \mathbf{u} = u_i$$
$$[\text{ first row of } \mathrm{D} \,]\, \mathbf{u} = u_i'$$

- Solve the system (least sq)
- Get error estimate from repeating the step with $2n$ Chebyshev points and comparing $u_n(t_{i+1})$ with $u_{2n}(t_{i+1})$. Typically, $n = 16$.

[2] We set $\gamma(t) = 0$ for simplicity.

The nonoscillatory phase function

- Rewrite $u'' + \omega^2 u = 0$ [3] using $u = e^z$, and $z'(t) = x(t)$:

$$x'(t) + x^2(t) + \omega^2(t) = 0, \quad \text{(Riccati)}$$

[3] Again setting $\gamma(t) = 0$.

The nonoscillatory phase function

- Rewrite $u'' + \omega^2 u = 0$ [3] using $u = e^z$, and $z'(t) = x(t)$:

$$x'(t) + x^2(t) + \omega^2(t) = 0, \quad \text{(Riccati)}$$

- Most solutions $x(t)$ (the **phase function**) are oscillatory $\rightarrow$ brute-force solution not feasible

[3] Again setting $\gamma(t) = 0$.

The nonoscillatory phase function

- Rewrite $u'' + \omega^2 u = 0$ [3] using $u = e^z$, and $z'(t) = x(t)$:

$$x'(t) + x^2(t) + \omega^2(t) = 0, \quad \text{(Riccati)}$$

- Most solutions $x(t)$ (the **phase function**) are oscillatory $\rightarrow$ brute-force solution not feasible
- But Heitman et al., *JCP* (2015): there exist nonoscillatory[4] $x(t)$ for analytic $\omega(t)$

[3] Again setting $\gamma(t) = 0$.
[4] In the sense that its logarithm's Fourier transform decays rapidly.

The nonoscillatory phase function

- Rewrite $u'' + \omega^2 u = 0$ [3] using $u = e^z$, and $z'(t) = x(t)$:

$$x'(t) + x^2(t) + \omega^2(t) = 0, \quad \text{(Riccati)}$$

- Most solutions $x(t)$ (the **phase function**) are oscillatory $\rightarrow$ brute-force solution not feasible
- But Heitman et al., *JCP* (2015): there exist nonoscillatory[4] $x(t)$ for analytic $\omega(t)$
- Bremer, *ACHA* (2018) (the Kummer's phase function method) build an oscillatory solver by finding the appropriate initial conditions that yield a nonoscillatory $x(t)$

[3] Again setting $\gamma(t) = 0$.
[4] In the sense that its logarithm's Fourier transform decays rapidly.

The nonoscillatory phase function

- Rewrite $u'' + \omega^2 u = 0$ [3] using $u = e^z$, and $z'(t) = x(t)$:

$$x'(t) + x^2(t) + \omega^2(t) = 0, \quad \text{(Riccati)}$$

- Most solutions $x(t)$ (the **phase function**) are oscillatory $\rightarrow$ brute-force solution not feasible
- But Heitman et al., *JCP* (2015): there exist nonoscillatory[4] $x(t)$ for analytic $\omega(t)$
- Bremer, *ACHA* (2018) (the Kummer's phase function method) build an oscillatory solver by finding the appropriate initial conditions that yield a nonoscillatory $x(t)$
- Algorithm is complex and only works if $\omega(t)$ is large

[3] Again setting $\gamma(t) = 0$.
[4] In the sense that its logarithm's Fourier transform decays rapidly.

Riccati defect correction

- **This work:** construct approximate, nonoscillatory $x(t)$ by functional iteration. $x_0(t), x_1(t), \ldots, x_j(t)$ forms an asymptotic series.

Riccati defect correction

- **This work:** construct approximate, nonoscillatory $x(t)$ by functional iteration. $x_0(t), x_1(t), \ldots, x_j(t)$ forms an asymptotic series.
- If $\omega \gg 1$, approximate nonoscillatory solutions $x(t) \approx \pm i\omega(t) \to$ let $x_0 := \pm i\omega$
- Define residual of **Riccati eq.** as

$$R[x](t) := R[x] = x' + x^2 + \omega^2, \quad \text{then}$$

Riccati defect correction

- **This work:** construct approximate, nonoscillatory $x(t)$ by functional iteration. $x_0(t), x_1(t), \ldots, x_j(t)$ forms an asymptotic series.
- If $\omega \gg 1$, approximate nonoscillatory solutions $x(t) \approx \pm i\omega(t) \rightarrow$ let $x_0 := \pm i\omega$
- Define residual of **Riccati eq.** as

$$R[x](t) := R[x] = x' + x^2 + \omega^2, \quad \text{then}$$
$$R[x_j + \delta] = R[x_j] + \delta' + 2x_j\delta + \mathcal{O}(\delta^2)$$

Riccati defect correction

- **This work:** construct approximate, nonoscillatory $x(t)$ by functional iteration. $x_0(t), x_1(t), \dots, x_j(t)$ forms an asymptotic series.
- If $\omega \gg 1$, approximate nonoscillatory solutions $x(t) \approx \pm i\omega(t) \to$ let $x_0 := \pm i\omega$
- Define residual of **Riccati eq.** as

$$R[x](t) := R[x] = x' + x^2 + \omega^2, \quad \text{then}$$
$$0 = R[x_j + \delta] = R[x_j] + \delta' + 2x_j\delta + \mathcal{O}(\delta^2)$$

- Seek a δ giving $R \equiv 0$.

Riccati defect correction

- **This work:** construct approximate, nonoscillatory $x(t)$ by functional iteration. $x_0(t), x_1(t), \ldots, x_j(t)$ forms an asymptotic series.
- If $\omega \gg 1$, approximate nonoscillatory solutions $x(t) \approx \pm i\omega(t) \rightarrow$ let $x_0 := \pm i\omega$
- Define residual of **Riccati eq.** as

$$R[x](t) := R[x] = x' + x^2 + \omega^2, \quad \text{then}$$
$$0 = R[x_j] + \delta' + 2x_j\delta + \mathcal{O}(\delta^2)$$

- Seek a δ giving $R \equiv 0$. After linearisation, δ solves an ODE which again is generally oscillatory

Riccati defect correction

- **This work:** construct approximate, nonoscillatory $x(t)$ by functional iteration. $x_0(t), x_1(t), \ldots, x_j(t)$ forms an asymptotic series.
- If $\omega \gg 1$, approximate nonoscillatory solutions $x(t) \approx \pm i\omega(t) \rightarrow$ let $x_0 := \pm i\omega$
- Define residual of **Riccati eq.** as

$$R[x](t) := R[x] = x' + x^2 + \omega^2, \quad \text{then}$$
$$0 = R[x_j] + \delta' + 2x_j\delta$$

- Seek a δ giving $R \equiv 0$. After linearisation, δ solves an ODE which again is generally oscillatory
- But if δ nonoscillatory, δ' is $\mathcal{O}(\omega)$ smaller than other terms $\rightarrow$ neglect,

Riccati defect correction

- **This work:** construct approximate, nonoscillatory $x(t)$ by functional iteration. $x_0(t), x_1(t), \ldots, x_j(t)$ forms an asymptotic series.
- If $\omega \gg 1$, approximate nonoscillatory solutions $x(t) \approx \pm i\omega(t) \rightarrow$ let $x_0 := \pm i\omega$
- Define residual of **Riccati eq.** as

$$R[x](t) := R[x] = x' + x^2 + \omega^2, \quad \text{then}$$
$$0 = \qquad R[x_j] \qquad + 2x_j\delta$$

- Seek a δ giving $R \equiv 0$. After linearisation, δ solves an ODE which again is generally oscillatory
- But if δ nonoscillatory, δ' is $\mathcal{O}(\omega)$ smaller than other terms $\rightarrow$ neglect,
- Get Newton-like, functional defect correction scheme:

$$x_{j+1}(t) = x_j(t) - \frac{R[x_j](t)}{2x_j(t)} \quad \text{for all } t \in [t_i, t_{i+1}]$$

Riccati defect correction

- **This work:** construct approximate, nonoscillatory $x(t)$ by functional iteration. $x_0(t), x_1(t), \dots, x_j(t)$ forms an asymptotic series.
- If $\omega \gg 1$, approximate nonoscillatory solutions $x(t) \approx \pm i\omega(t) \to$ let $x_0 := \pm i\omega$
- Define residual of **Riccati eq.** as

$$R[x](t) := R[x] = x' + x^2 + \omega^2, \quad \text{then}$$
$$0 = \qquad R[x_j] \qquad + 2x_j\delta$$

- Seek a δ giving $R \equiv 0$. After linearisation, δ solves an ODE which again is generally oscillatory
- But if δ nonoscillatory, δ' is $\mathcal{O}(\omega)$ smaller than other terms $\to$ neglect,
- Get Newton-like, functional defect correction scheme:

$$x_{j+1}(t) = x_j(t) - \frac{R[x_j](t)}{2x_j(t)} \quad \text{for all } t \in [t_i, t_{i+1}]$$

- Check: if $x = \mathcal{O}(\omega)$, $\omega' = \mathcal{O}(\omega)$, then

Riccati defect correction

- **This work:** construct approximate, nonoscillatory $x(t)$ by functional iteration. $x_0(t), x_1(t), \ldots, x_j(t)$ forms an asymptotic series.
- If $\omega \gg 1$, approximate nonoscillatory solutions $x(t) \approx \pm i\omega(t) \rightarrow$ let $x_0 := \pm i\omega$
- Define residual of **Riccati eq.** as

$$R[x](t) := R[x] = x' + x^2 + \omega^2, \quad \text{then}$$

$$0 = \qquad R[x_j] \qquad + 2x_j\delta$$

- Seek a δ giving $R \equiv 0$. After linearisation, δ solves an ODE which again is generally oscillatory
- But if δ nonoscillatory, δ' is $\mathcal{O}(\omega)$ smaller than other terms $\rightarrow$ neglect,
- Get Newton-like, functional defect correction scheme:

$$x_{j+1}(t) = x_j(t) - \frac{R[x_j](t)}{2x_j(t)} \quad \text{for all } t \in [t_i, t_{i+1}]$$

- Check: if $x = \mathcal{O}(\omega)$, $\omega' = \mathcal{O}(\omega)$, then

$$x_0 = i\omega, \qquad R[x_0] = i\omega' = \mathcal{O}(\omega),$$

Riccati defect correction

- **This work:** construct approximate, nonoscillatory $x(t)$ by functional iteration. $x_0(t), x_1(t), \dots, x_j(t)$ forms an asymptotic series.
- If $\omega \gg 1$, approximate nonoscillatory solutions $x(t) \approx \pm i\omega(t) \to$ let $x_0 := \pm i\omega$
- Define residual of **Riccati eq.** as

$$R[x](t) := R[x] = x' + x^2 + \omega^2, \quad \text{then}$$

$$0 = \qquad R[x_j] \qquad + 2x_j\delta$$

- Seek a δ giving $R \equiv 0$. After linearisation, δ solves an ODE which again is generally oscillatory
- But if δ nonoscillatory, δ' is $\mathcal{O}(\omega)$ smaller than other terms $\to$ neglect,
- Get Newton-like, functional defect correction scheme:

$$x_{j+1}(t) = x_j(t) - \frac{R[x_j](t)}{2x_j(t)} \quad \text{for all } t \in [t_i, t_{i+1}]$$

- Check: if $x = \mathcal{O}(\omega)$, $\omega' = \mathcal{O}(\omega)$, then

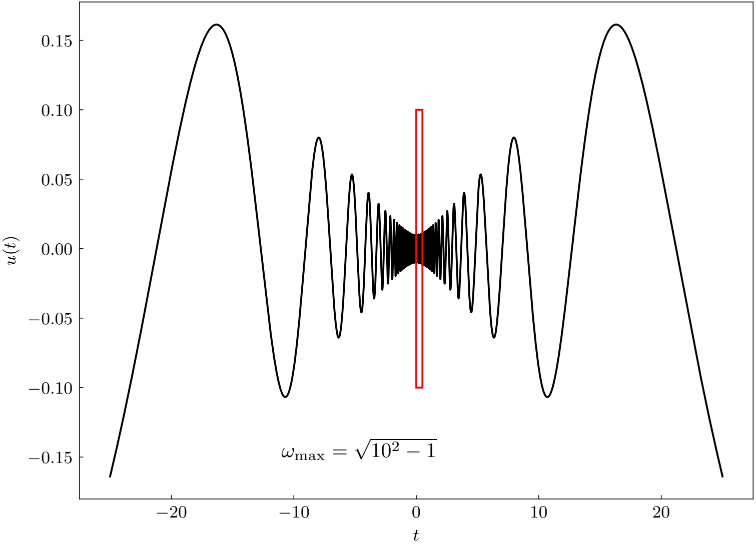
$$x_0 = i\omega, \qquad R[x_0] = i\omega' = \mathcal{O}(\omega),$$

$$x_1 = i\omega - \frac{\omega'}{2\omega}, \qquad R[x_1] = -\frac{\omega''}{2\omega} + \frac{3(\omega')^2}{4\omega^2} = \mathcal{O}(1).$$

Empirical residual drop, $u'' + \frac{m^2-1}{(1+t^2)^2} u = 0$

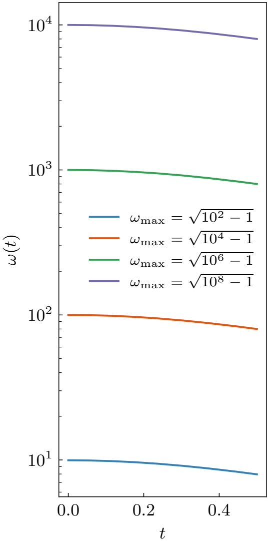
Empirical residual drop, $u'' + \frac{m^2-1}{(1+t^2)^2} u = 0$

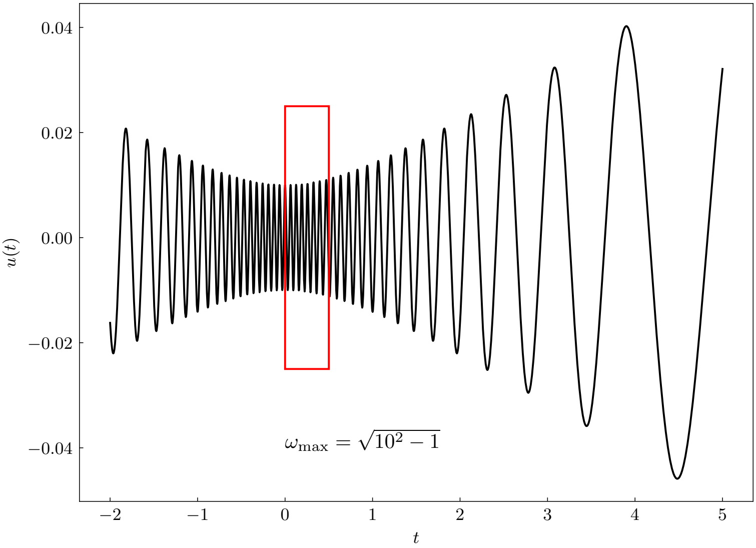
Here, $\omega_{\max} = \max_{t \in [t_i, t_{i+1}]} \omega(t), \quad [t_i, t_{i+1}] = [0, 0.5]$



Empirical residual drop, $u'' + \frac{m^2-1}{(1+t^2)^2} u = 0$

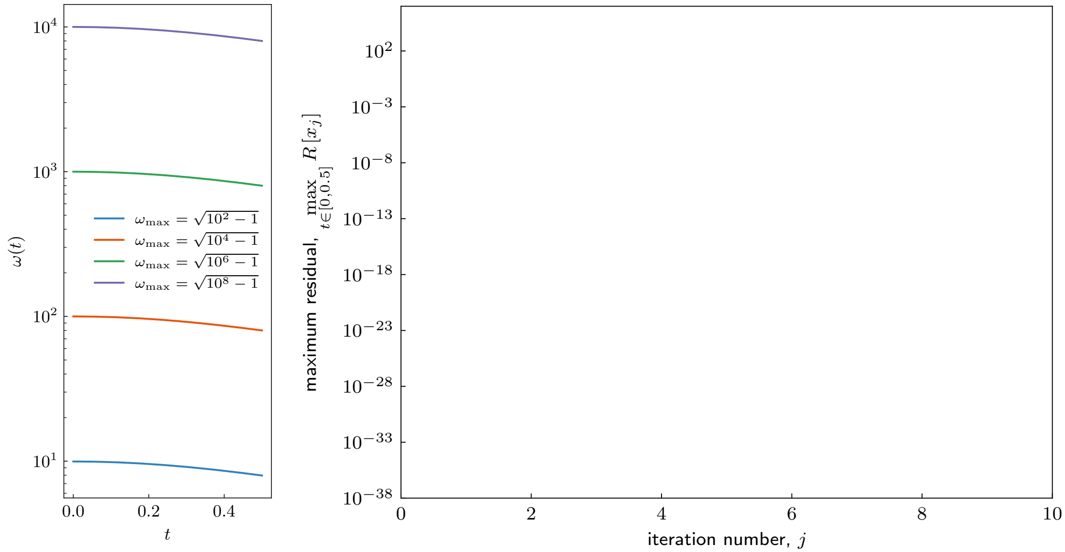
Here, $\omega_{\max} = \max_{t \in [t_i, t_{i+1}]} \omega(t), \quad [t_i, t_{i+1}] = [0, 0.5]$





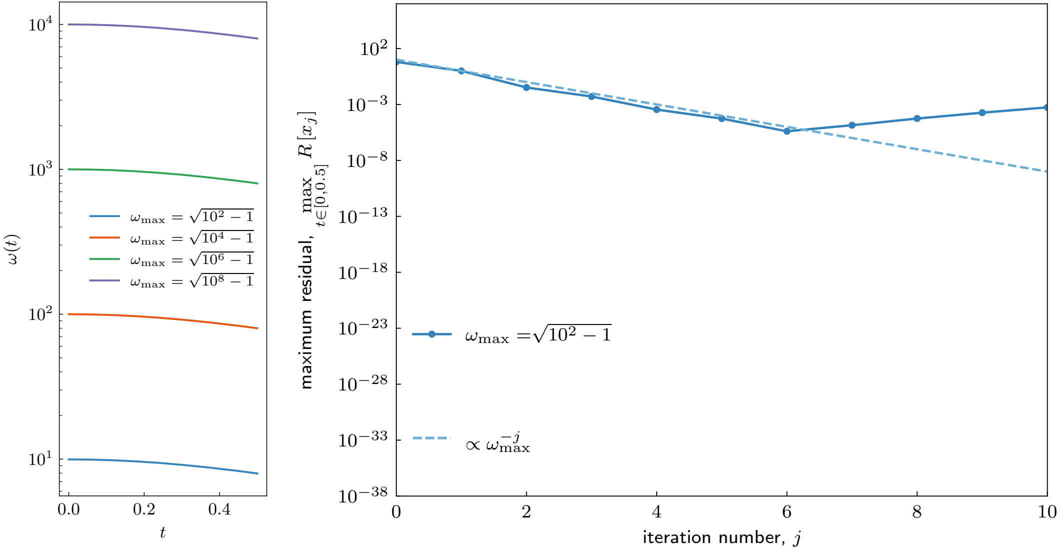
Empirical residual drop, $u'' + \frac{m^2-1}{(1+t^2)^2} u = 0$

Here, $\omega_{\max} = \max_{t\in[t_i,t_{i+1}]} \omega(t)$, $\quad [t_i, t_{i+1}] = [0, 0.5]$



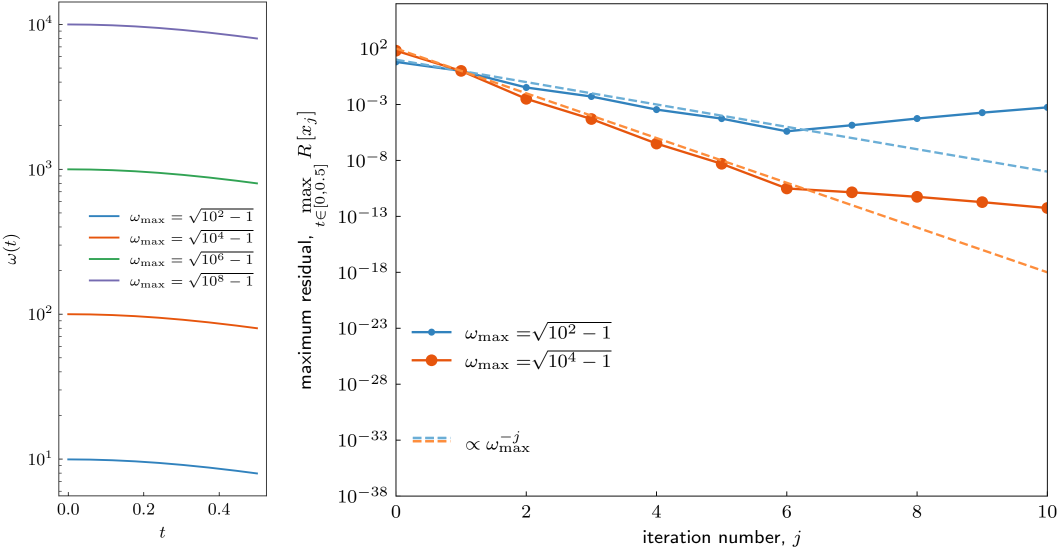
Empirical residual drop, $u'' + \frac{m^2-1}{(1+t^2)^2} u = 0$

Here, $\omega_{\max} = \max_{t \in [t_i, t_{i+1}]} \omega(t)$, $\quad [t_i, t_{i+1}] = [0, 0.5]$



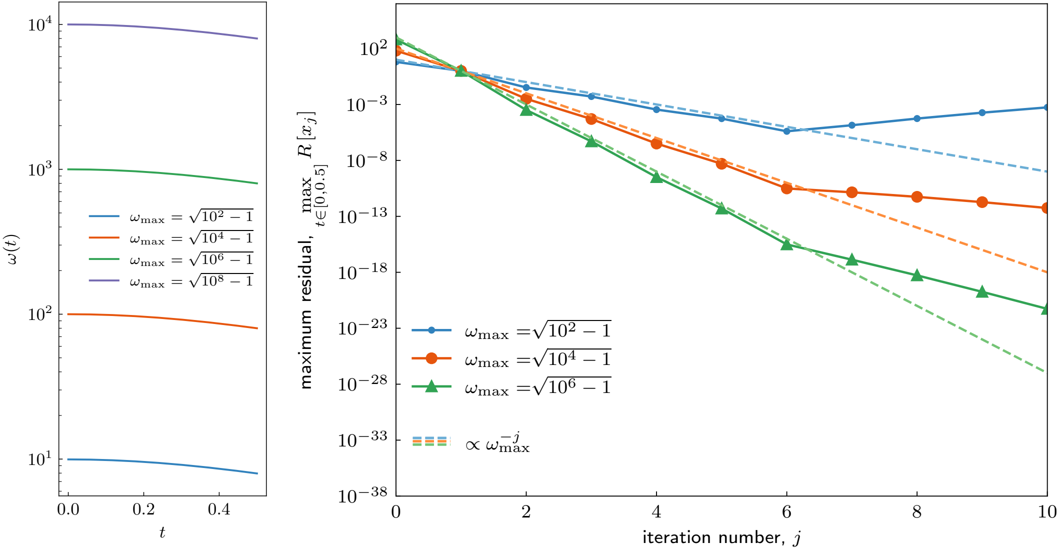
Empirical residual drop, $u'' + \frac{m^2-1}{(1+t^2)^2} u = 0$

Here, $\omega_{\max} = \max_{t\in[t_i, t_{i+1}]} \omega(t)$, $\quad [t_i, t_{i+1}] = [0, 0.5]$



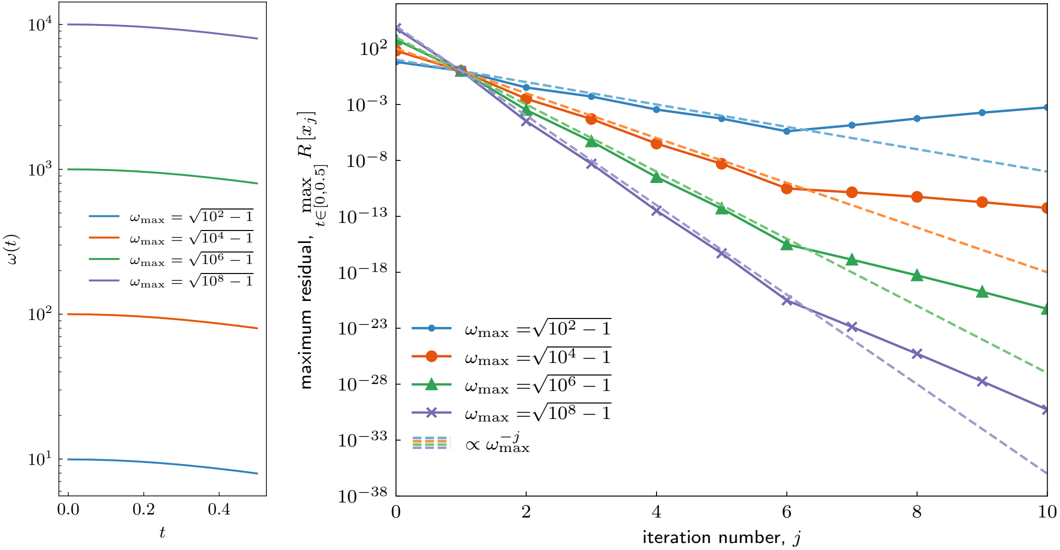
Empirical residual drop, $u'' + \frac{m^2-1}{(1+t^2)^2} u = 0$

Here, $\omega_{\max} = \max_{t\in[t_i,t_{i+1}]} \omega(t)$, $\quad [t_i, t_{i+1}] = [0, 0.5]$



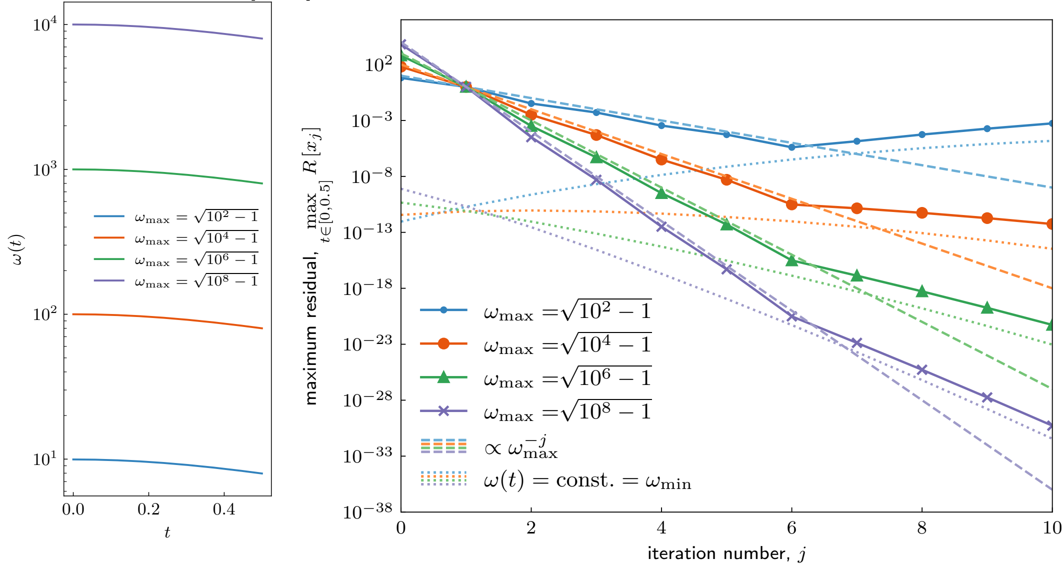
Empirical residual drop, $u'' + \frac{m^2-1}{(1+t^2)^2} u = 0$

Here, $\omega_{\max} = \max_{t\in[t_i,t_{i+1}]} \omega(t)$, $\quad [t_i, t_{i+1}] = [0, 0.5]$



Empirical residual drop, $u'' + \frac{m^2-1}{(1+t^2)^2} u = 0$

Here, $\omega_{\max} = \max_{t \in [t_i, t_{i+1}]} \omega(t), \quad [t_i, t_{i+1}] = [0, 0.5]$



Geometric convergence of the residual, $R[x_j]$, *for a while*: a theorem

Geometric convergence of the residual, $R[x_j]$, *for a while*: a theorem

Theorem

Let ω *be analytic in the closed ball* $B_\rho := \{z \in \mathbb{C} : |z - t| \leq \rho\}$ *centered on a given* t*.*
Then for $j = 1, 2, \ldots, k$,

$$R_j(t) \leq Ar^j$$

with

$$r(|\omega'|_{B_\rho}, |\omega|_{B_\rho}, k).$$

Geometric convergence of the residual, $R[x_j]$, *for a while*: a theorem

Theorem

Let ω *be analytic in the closed ball* $B_\rho := \{z \in \mathbb{C} : |z - t| \leq \rho\}$ *centered on a given* t.
Then for $j = 1, 2, \ldots, k$,

$$R_j(t) \leq Ar^j$$

with

$$r(|\omega'|_{B_\rho}, |\omega|_{B_\rho}, k).$$

Meaning:

Geometric convergence of the residual, $R[x_j]$, *for a while*: a theorem

Theorem

Let ω *be analytic in the closed ball* $B_\rho := \{z \in \mathbb{C} : |z - t| \leq \rho\}$ *centered on a given* t*.*
Then for $j = 1, 2, \ldots, k$,

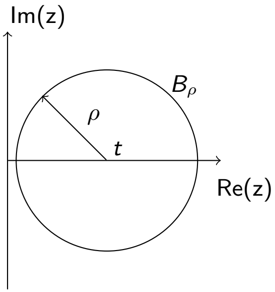
$$R_j(t) \leq Ar^j$$

with

$$r(|\omega'|_{B_\rho}, |\omega|_{B_\rho}, k).$$

Meaning:

- If $|\omega'|/|\omega|$ is small in B_ρ,



Geometric convergence of the residual, $R[x_j]$, *for a while*: a theorem

Theorem

Let ω be analytic in the closed ball $B_\rho := \{z \in \mathbb{C} : |z - t| \leq \rho\}$ centered on a given t. Then for $j = 1, 2, \ldots, k$,

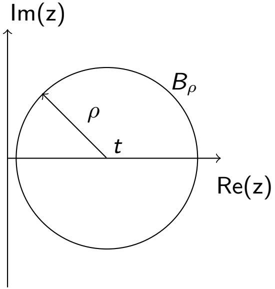
$$R_j(t) \leq Ar^j$$

with

$$r(|\omega'|_{B_\rho}, |\omega|_{B_\rho}, k).$$

Meaning:

- If $|\omega'|/|\omega|$ is small in B_ρ,
- and $|\omega|$ is large in B_ρ,



Geometric convergence of the residual, $R[x_j]$, *for a while*: a theorem

Theorem

Let ω be analytic in the closed ball $B_\rho := \{z \in \mathbb{C} : |z - t| \leq \rho\}$ centered on a given t.
Then for $j = 1, 2, \ldots, k$,

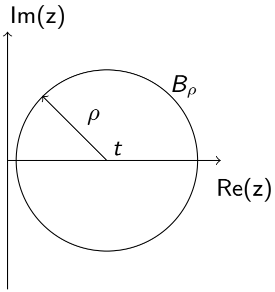
$$R_j(t) \leq Ar^j$$

with

$$r(|\omega'|_{B_\rho}, |\omega|_{B_\rho}, k).$$

Meaning:

- If $|\omega'|/|\omega|$ is small in B_ρ,
- and $|\omega|$ is large in B_ρ,
- then geometric convergence up to $j \leq k$ iterations.



Geometric convergence of the residual, $R[x_j]$, *for a while*: a theorem

Theorem

Let ω *be analytic in the closed ball* $B_\rho := \{z \in \mathbb{C} : |z - t| \le \rho\}$ *centered on a given* t*.*
Then for $j = 1, 2, \ldots, k$,

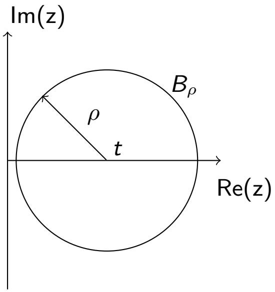
$$R_j(t) \le Ar^j$$

with

$$r(|\omega'|_{B_\rho}, |\omega|_{B_\rho}, k).$$

Meaning:

- If $|\omega'|/|\omega|$ is small in B_ρ,
- and $|\omega|$ is large in B_ρ,
- then geometric convergence up to $j \le k$ iterations.
- Note: The theorem generalises to the $\gamma(t) \ne 0$ case by introducing an upper bound on γ.



Geometric convergence of the residual, $R[x_j]$, *for a while*: proof

Proof:

Geometric convergence of the residual, $R[x_j]$, *for a while*: proof

Proof:

- Write down residual iteration ($R[x_{j+1}] := R_{j+1}$ in terms of R_j):

$$R_{j+1} = \frac{1}{2x_j}\left(\frac{x_j'}{x_j}R_j - R_j'\right) + \left(\frac{R_j}{2x_j}\right)^2.$$

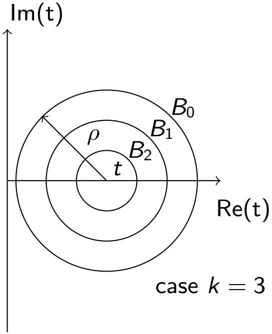
Geometric convergence of the residual, $R[x_j]$, *for a while*: proof

Proof:

- Write down residual iteration ($R[x_{j+1}] := R_{j+1}$ in terms of R_j):

$$R_{j+1} = \frac{1}{2x_j}\left(\frac{x_j'}{x_j}R_j - R_j'\right) + \left(\frac{R_j}{2x_j}\right)^2.$$

- Define the concentric nested set of closed balls $B_j = B_{\rho_j}(t)$, with radii $\rho_j = (1 - j/k)\rho$, $j = 0, 1, \ldots, k$



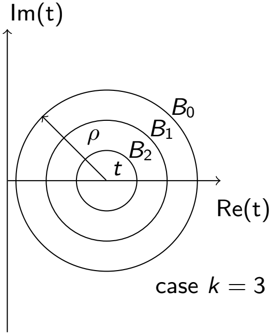
Geometric convergence of the residual, $R[x_j]$, *for a while*: proof

Proof:

- Write down residual iteration ($R[x_{j+1}] := R_{j+1}$ in terms of R_j):

$$R_{j+1} = \frac{1}{2x_j}\left(\frac{x_j'}{x_j}R_j - R_j'\right) + \left(\frac{R_j}{2x_j}\right)^2 .$$

- Define the concentric nested set of closed balls $B_j = B_{\rho_j}(t)$, with radii $\rho_j = (1 - j/k)\rho$, $j = 0, 1, \ldots, k$
- Bound f' in B_{j+1} in terms of $||f||_j = \max_{z \in B_j} |f(z)|$ by using Cauchy's theorem for derivatives,



Geometric convergence of the residual, $R[x_j]$, *for a while*: proof

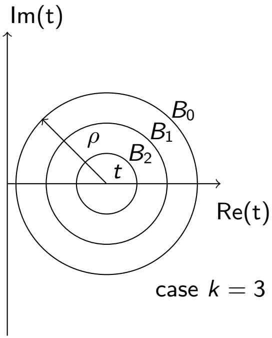
Proof:

- Write down residual iteration ($R[x_{j+1}] := R_{j+1}$ in terms of R_j):

$$R_{j+1} = \frac{1}{2x_j}\left(\frac{x_j'}{x_j}R_j - R_j'\right) + \left(\frac{R_j}{2x_j}\right)^2.$$

- Define the concentric nested set of closed balls $B_j = B_{\rho_j}(t)$, with radii $\rho_j = (1 - j/k)\rho$, $j = 0, 1, \ldots, k$
- Bound f' in B_{j+1} in terms of $||f||_j = \max_{z \in B_j} |f(z)|$ by using Cauchy's theorem for derivatives,
- Prove by induction that for iteration j,

$$\tilde{\eta}_1 \leq |x_l| \leq \tilde{\eta}_2 \quad \text{in } B_j, \quad \text{for all } l = 0, 1, \ldots, j,$$
$$|R_l| \leq \eta_3 r^l \quad \text{in } B_j, \quad \text{for all } l = 0, 1, \ldots, j.$$



Methods II: asymptotic expansion /3

Methods II: asymptotic expansion /3

- Once we have x_j, transform back:

$$u(t) = e^{\int^t x_j(\sigma)\mathrm{d}\sigma}$$

Methods II: asymptotic expansion /3

- Once we have x_j, transform back:

$$u(t) = e^{\int^t x_j(\sigma)\mathrm{d}\sigma}$$

- Two solutions for x_j: $x_{j\pm}$ (starting from $\pm i\omega$) give linearly independent solutions for u, $u_\pm$

Methods II: asymptotic expansion /3

- Once we have x_j, transform back:

$$u(t) = e^{\int^t x_j(\sigma)\mathrm{d}\sigma}$$

- Two solutions for x_j: $x_{j\pm}$ (starting from $\pm i\omega$) give linearly independent solutions for u, $u_\pm$
- Linearly combine to match initial conditions at the start of each timestep:

Methods II: asymptotic expansion /3

- Once we have x_j, transform back:

$$u(t) = e^{\int^t x_j(\sigma)\mathrm{d}\sigma}$$

- Two solutions for x_j: $x_{j\pm}$ (starting from $\pm i\omega$) give linearly independent solutions for u, $u_\pm$
- Linearly combine to match initial conditions at the start of each timestep:

$$u(t_{i+1}) = Au_+ + Bu_-, \quad u'(t_{i+1}) = Au'_+ + Bu'_-$$

Methods II: asymptotic expansion /3

- Once we have x_j, transform back:

$$u(t) = e^{\int^t x_j(\sigma)\mathrm{d}\sigma}$$

- Two solutions for x_j: $x_{j\pm}$ (starting from $\pm i\omega$) give linearly independent solutions for u, $u_\pm$
- Linearly combine to match initial conditions at the start of each timestep:

$$u(t_{i+1}) = Au_+ + Bu_-, \quad u'(t_{i+1}) = Au'_+ + Bu'_-$$

Methods II: asymptotic expansion /3

- Once we have x_j, transform back:

$$u(t) = e^{\int^t x_j(\sigma)\mathrm{d}\sigma}$$

- Two solutions for x_j: $x_{j\pm}$ (starting from $\pm i\omega$) give linearly independent solutions for u, $u_\pm$
- Linearly combine to match initial conditions at the start of each timestep:

$$u(t_{i+1}) = Au_+ + Bu_-, \quad u'(t_{i+1}) = Au'_+ + Bu'_-$$

- Error estimate is via residual $R[x_j]$. Fix stepsize, iterate over j.

Methods II: asymptotic expansion /3

- Once we have x_j, transform back:

$$u(t) = e^{\int^t x_j(\sigma) \mathrm{d}\sigma}$$

- Two solutions for x_j: $x_{j\pm}$ (starting from $\pm i\omega$) give linearly independent solutions for u, $u_\pm$
- Linearly combine to match initial conditions at the start of each timestep:

$$u(t_{i+1}) = Au_+ + Bu_-, \quad u'(t_{i+1}) = Au'_+ + Bu'_-$$

- Error estimate is via residual $R[x_j]$. Fix stepsize, iterate over j.
- Derivatives and integral via spectral differentiation / integration matrix ($n = 16, 32$) $\rightarrow$ stepsize determined only by how well ω, γ are represented on a Chebyshev grid

Algorithm overview

Algorithm overview

In stepping from t_i to $t_{i+1} = t_i + h$:

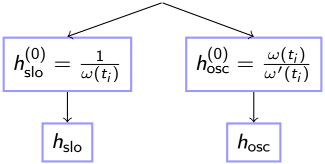
Algorithm overview

$$h_{\text{slo}}^{(0)} = \frac{1}{\omega(t_i)} \qquad h_{\text{osc}}^{(0)} = \frac{\omega(t_i)}{\omega'(t_i)}$$

In stepping from t_i to $t_{i+1} = t_i + h$:

1. Get initial stepsize estimate

Algorithm overview



In stepping from t_i to $t_{i+1} = t_i + h$:

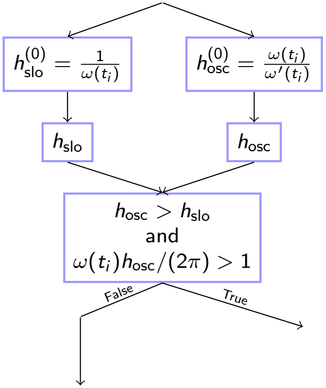
1. Get initial stepsize estimate

2. Refine stepsize estimate

Algorithm overview

In stepping from t_i to $t_{i+1} = t_i + h$:

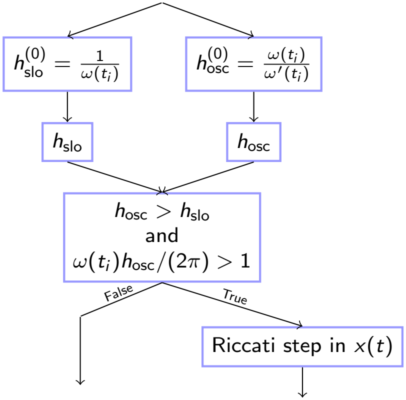
1. Get initial stepsize estimate
2. Refine stepsize estimate
3. Decide whether to attempt Riccati step



Algorithm overview

In stepping from t_i to $t_{i+1} = t_i + h$:

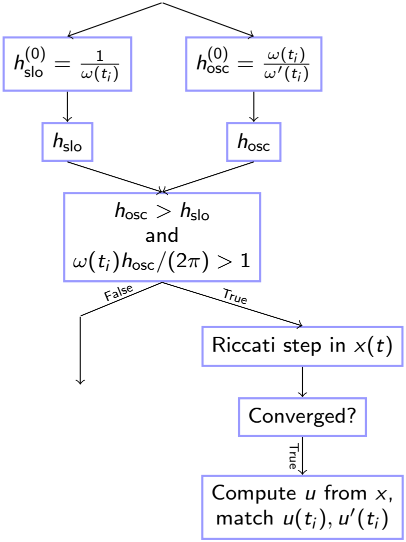
1. Get initial stepsize estimate
2. Refine stepsize estimate
3. Decide whether to attempt Riccati step
   3.1 Iterate over k to check if Riccati series converges



Algorithm overview

In stepping from t_i to $t_{i+1} = t_i + h$:

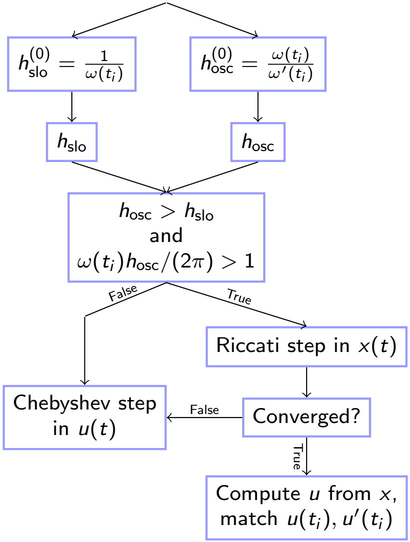
1. Get initial stepsize estimate
2. Refine stepsize estimate
3. Decide whether to attempt Riccati step
   - 3.1 Iterate over k to check if Riccati series converges
   - 3.2 If it does, accept it



Algorithm overview

In stepping from t_i to $t_{i+1} = t_i + h$:

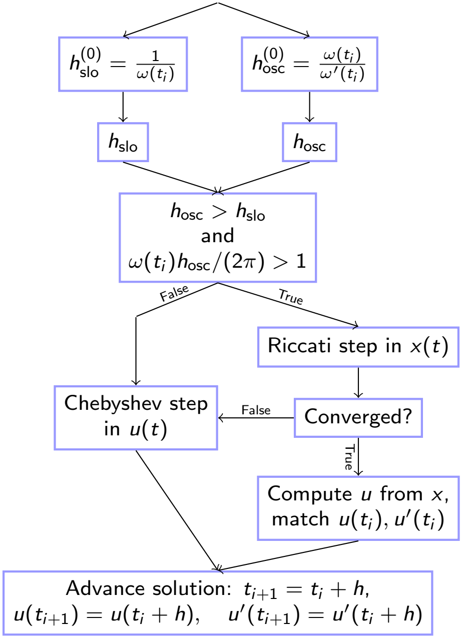
1. Get initial stepsize estimate
2. Refine stepsize estimate
3. Decide whether to attempt Riccati step
   3.1 Iterate over k to check if Riccati series converges
   3.2 If it does, accept it
   3.3 If it doesn't or solution not oscillatory enough, take Chebyshev step (iterate over n, h_{slo} if needed)



Algorithm overview

In stepping from t_i to $t_{i+1} = t_i + h$:

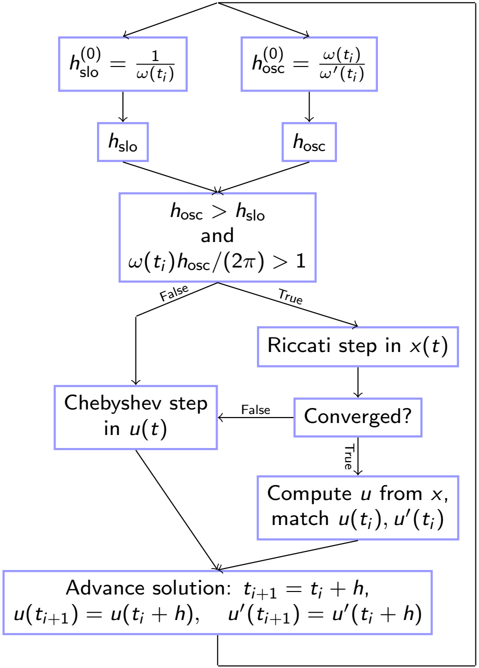
1. Get initial stepsize estimate
2. Refine stepsize estimate
3. Decide whether to attempt Riccati step
   3.1 Iterate over k to check if Riccati series converges
   3.2 If it does, accept it
   3.3 If it doesn't or solution not oscillatory enough, take Chebyshev step (iterate over n, h_{slo} if needed)
4. Advance time $t_i \to t_{i+1}$, and numerical solution



Algorithm overview

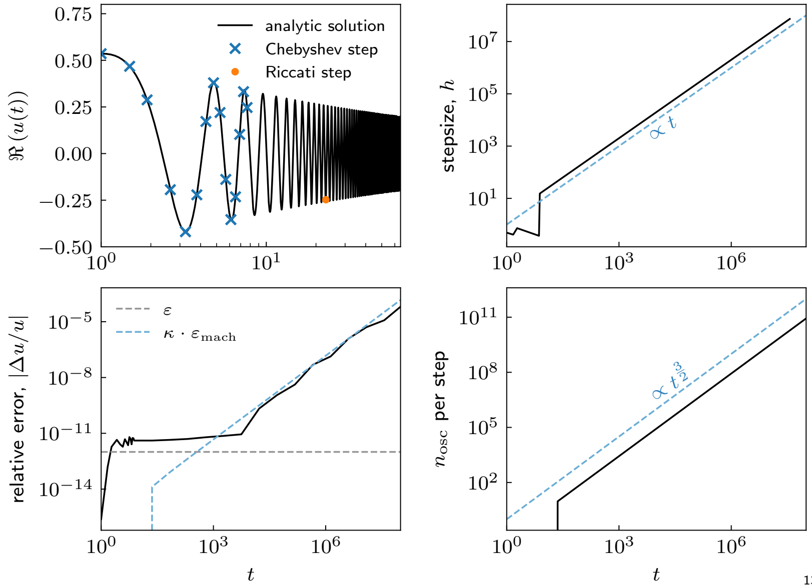
In stepping from t_i to $t_{i+1} = t_i + h$:

1. Get initial stepsize estimate
2. Refine stepsize estimate
3. Decide whether to attempt Riccati step
   3.1 Iterate over k to check if Riccati series converges
   3.2 If it does, accept it
   3.3 If it doesn't or solution not oscillatory enough, take Chebyshev step (iterate over n, h_{slo} if needed)
4. Advance time $t_i \to t_{i+1}$, and numerical solution



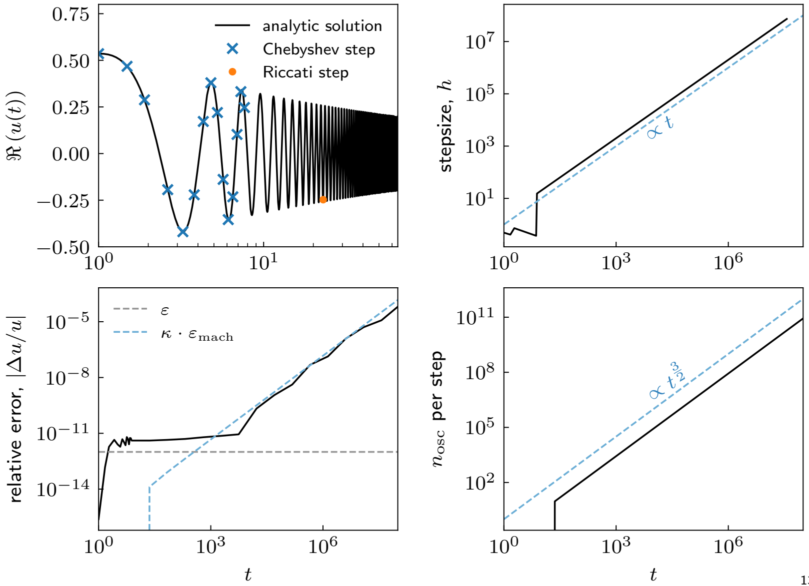
Examples I: Airy equation, $u'' + ut = 0$

Examples I: Airy equation, $u'' + ut = 0$



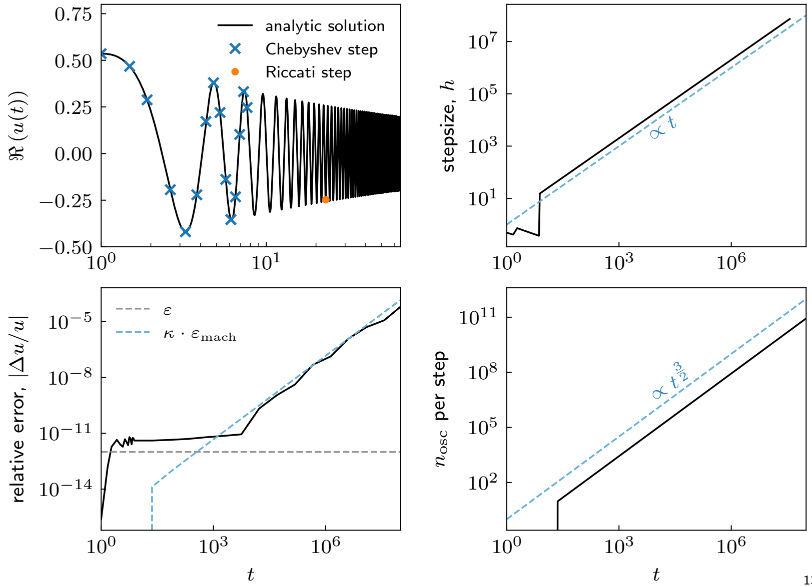
Examples I: Airy equation, $u'' + ut = 0$

- κ is condition number: sensitivity of the ODE to perturbations (Trefethen and Bau III (1997)). We approximate it as the total accumulated phase.



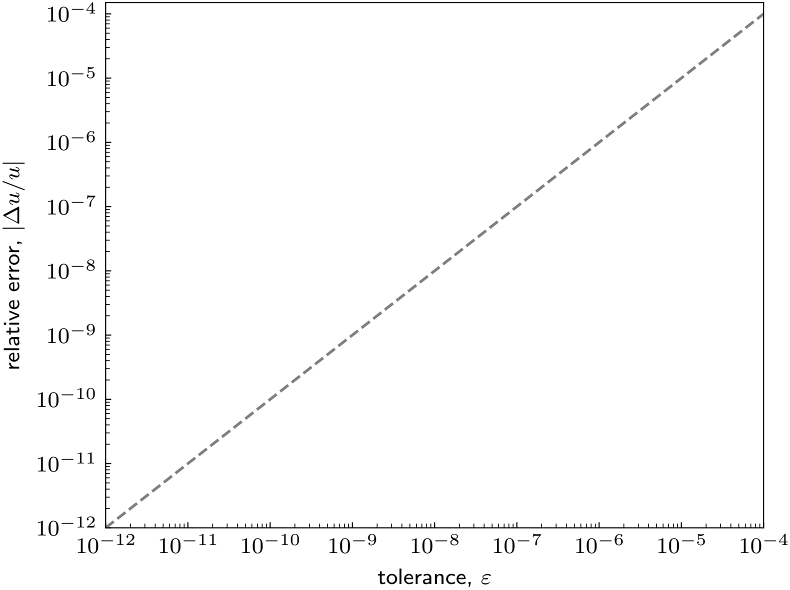
Examples I: Airy equation, $u'' + ut = 0$

- κ is condition number: sensitivity of the ODE to perturbations (Trefethen and Bau III (1997)). We approximate it as the total accumulated phase.
- The best attainable error is then $\kappa \cdot \varepsilon_{\mathrm{mach}}$, where $\varepsilon_{\mathrm{mach}}$ is machine precision

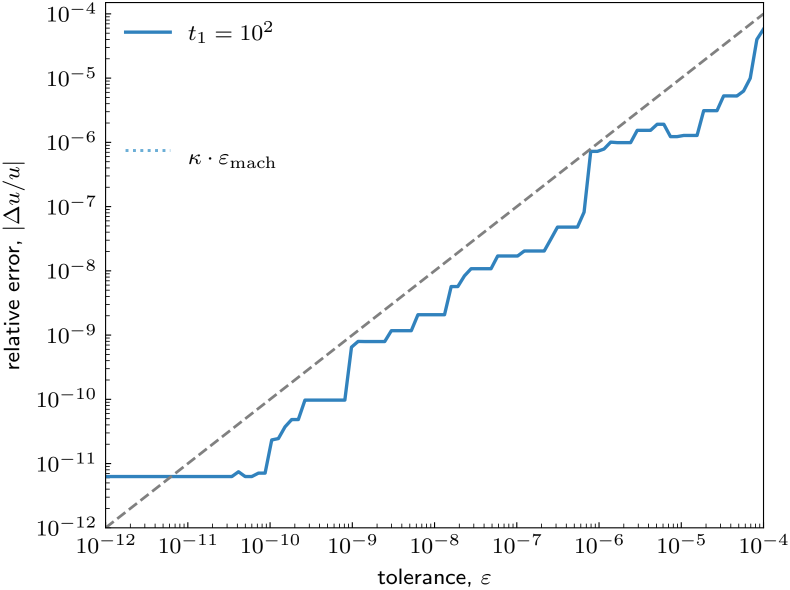


Adaptivity check (using the Airy equation)

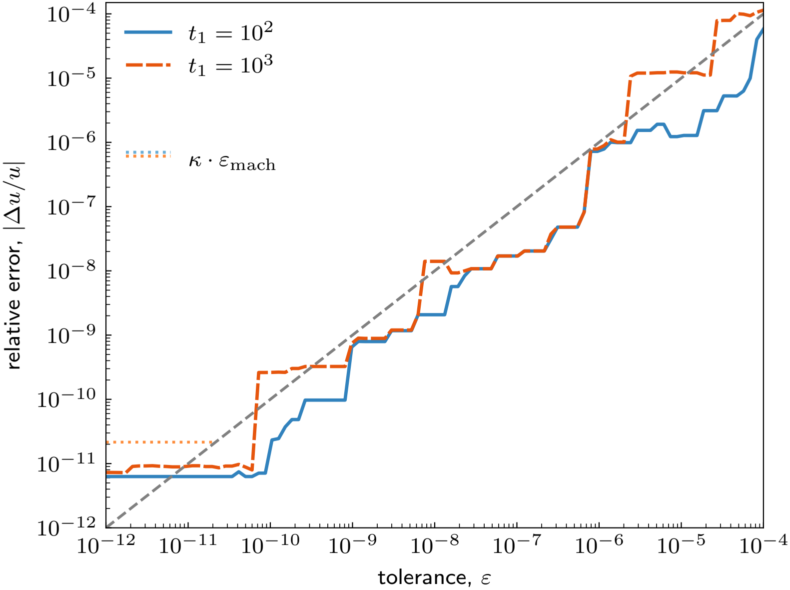
Adaptivity check (using the Airy equation)



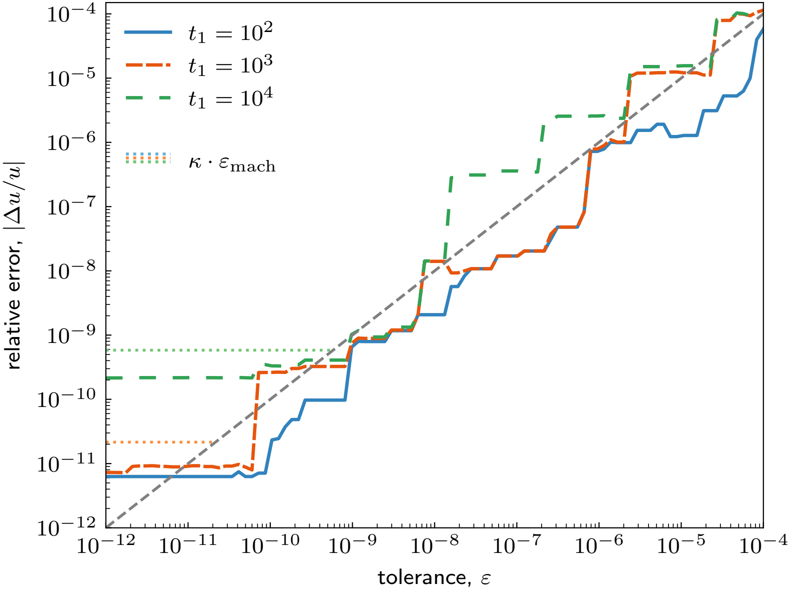
Adaptivity check (using the Airy equation)



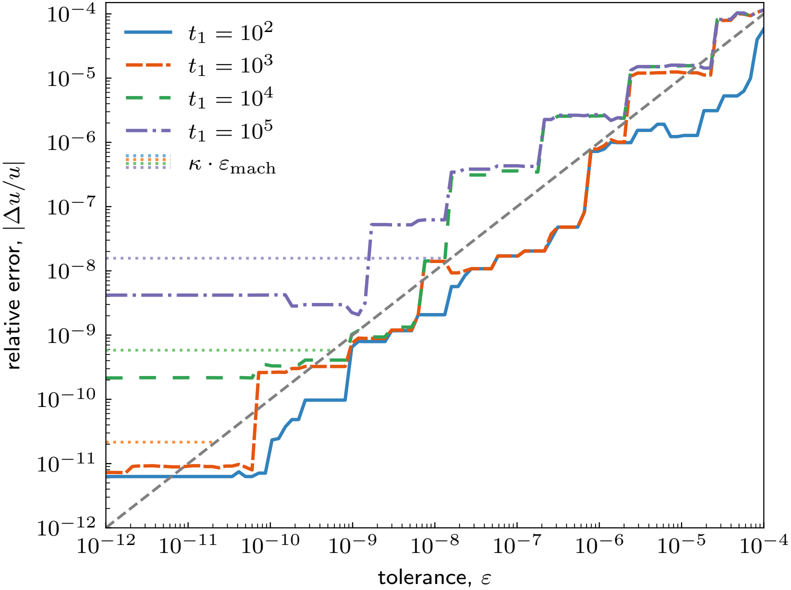
Adaptivity check (using the Airy equation)



Adaptivity check (using the Airy equation)



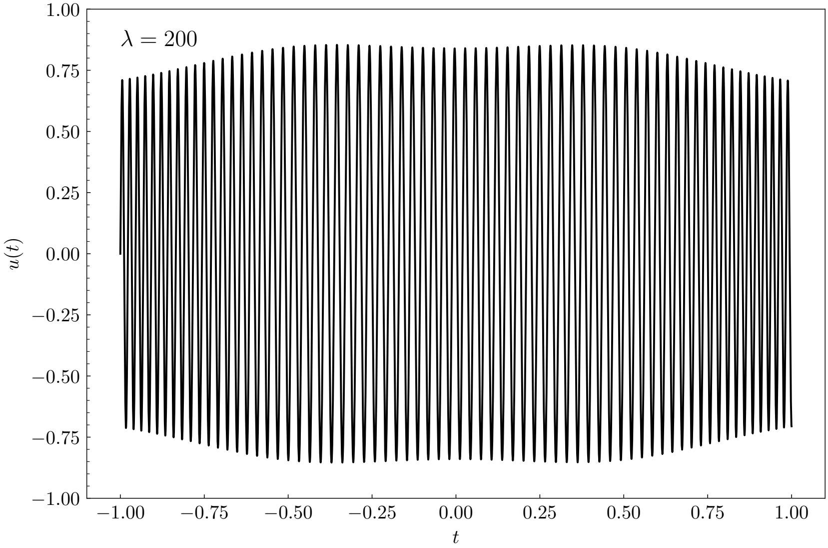
Adaptivity check (using the Airy equation)



Comparison with standard & state-of-the-art solvers, performance

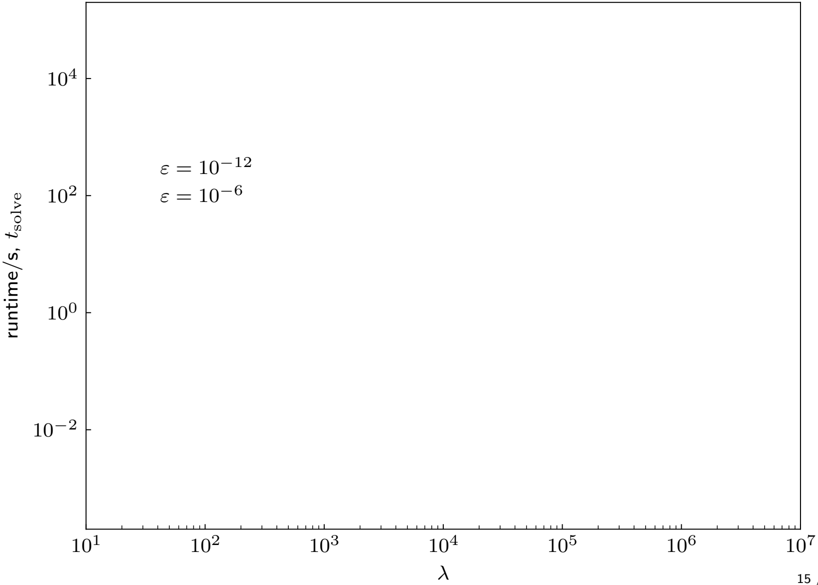
Comparison with standard & state-of-the-art solvers, performance

- $u'' + \lambda^2 q(t) u = 0$,
  $q(t) = 1 - t^2 \cos(3t)$,
  $t \in [-1, 1]$.



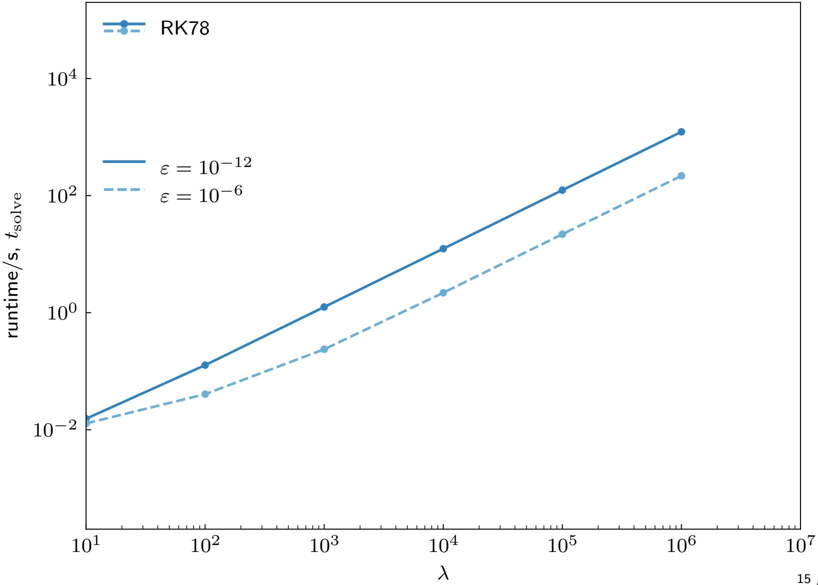
Comparison with standard & state-of-the-art solvers, performance

- $u'' + \lambda^2 q(t) u = 0$,
  $q(t) = 1 - t^2 \cos(3t)$,
  $t \in [-1, 1]$.
- RK78: Runge–Kutta,



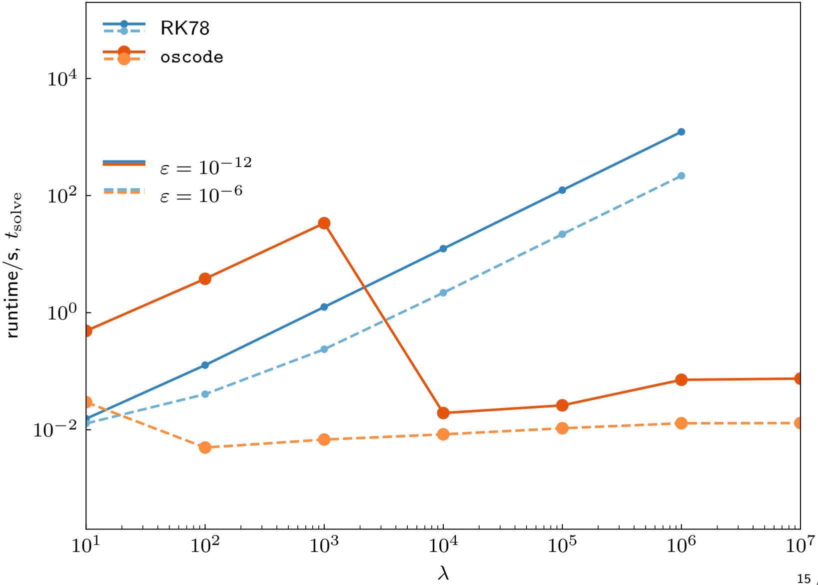
Comparison with standard & state-of-the-art solvers, performance

- $u'' + \lambda^2 q(t) u = 0$,
  $q(t) = 1 - t^2 \cos(3t)$,
  $t \in [-1, 1]$.
- RK78: Runge–Kutta,
  `oscode`: **Agocs**, Handley, et al., *Phys Rev Research* (2020),



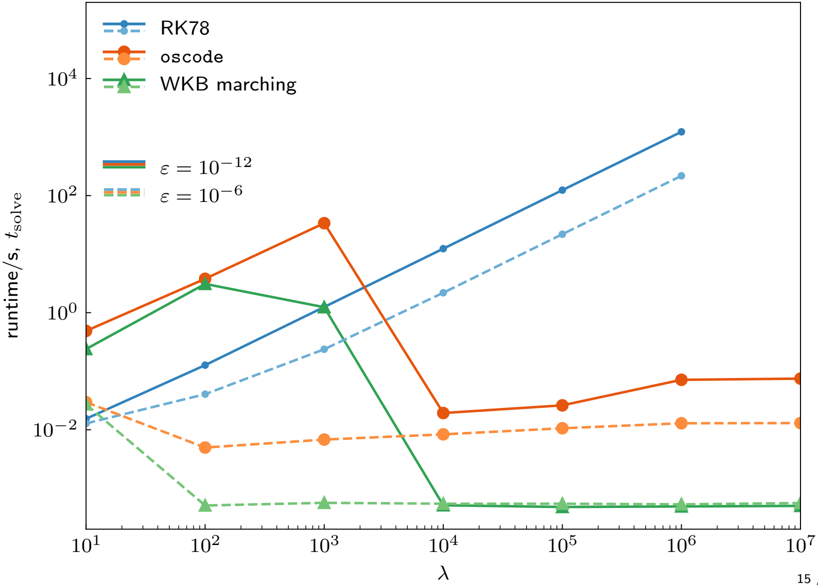
Comparison with standard & state-of-the-art solvers, performance

- $u'' + \lambda^2 q(t)u = 0$, $q(t) = 1 - t^2\cos(3t)$, $t \in [-1, 1]$.
- RK78: Runge–Kutta, `oscode`: **Agocs**, Handley, et al., *Phys Rev Research* (2020), WKB marching: Körner et al., *JCAM* (2022),



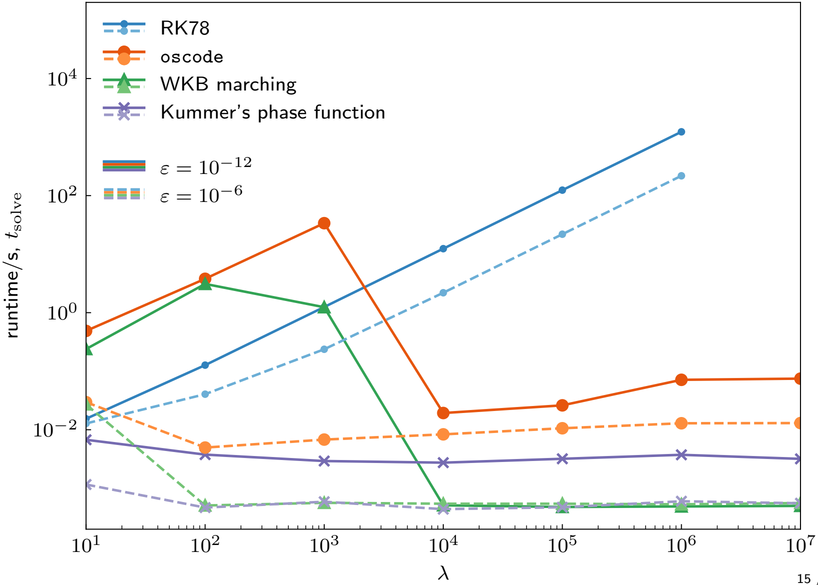
Comparison with standard & state-of-the-art solvers, performance

- $u'' + \lambda^2 q(t) u = 0$,
  $q(t) = 1 - t^2 \cos(3t)$,
  $t \in [-1, 1]$.
- RK78: Runge–Kutta,
  oscode: **Agocs**, Handley, et al., *Phys Rev Research* (2020),
  WKB marching: Körner et al., *JCAM* (2022),
  Kummer's phase function: Bremer, *ACHA* (2018).



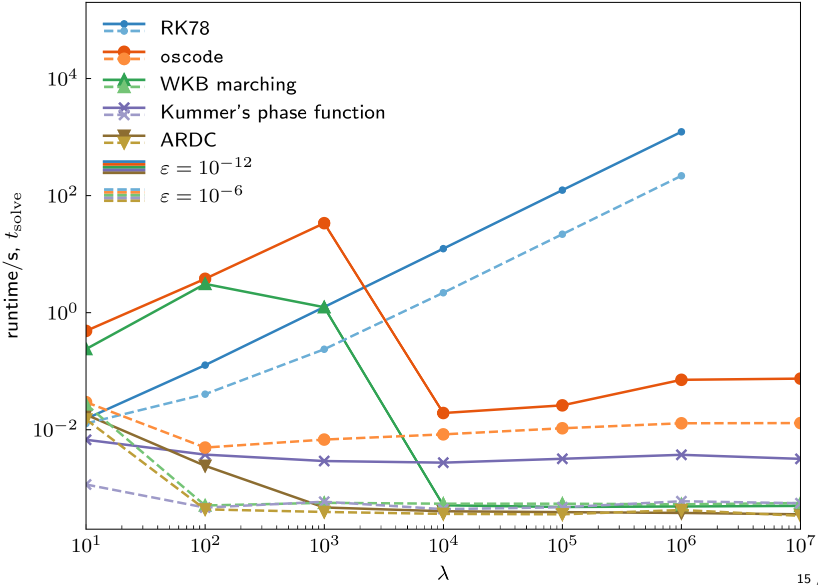
Comparison with standard & state-of-the-art solvers, performance

- $u'' + \lambda^2 q(t) u = 0$, $q(t) = 1 - t^2 \cos(3t)$, $t \in [-1, 1]$.
- RK78: Runge–Kutta, `oscode`: **Agocs**, Handley, et al., *Phys Rev Research* (2020), WKB marching: Körner et al., *JCAM* (2022), Kummer's phase function: Bremer, *ACHA* (2018).



Comparison with standard & state-of-the-art solvers, performance

- $u'' + \lambda^2 q(t) u = 0$,
  $q(t) = 1 - t^2 \cos(3t)$,
  $t \in [-1, 1]$.
- RK78: Runge–Kutta,
  oscode: **Agocs**, Handley, et al., *Phys Rev Research* (2020),
  WKB marching: Körner et al., *JCAM* (2022),
  Kummer's phase function: Bremer, *ACHA* (2018).



Current applications

Current applications

- Cosmology (previously unable to investigate these models because of costly oscillatory solve)

Current applications

- Cosmology (previously unable to investigate these models because of costly oscillatory solve)
    - closed universe models: Hergt, **Agocs**, et al., *Phys Rev D* (2022)

Current applications

- Cosmology (previously unable to investigate these models because of costly oscillatory solve)
  - closed universe models: Hergt, **Agocs**, et al., *Phys Rev D* (2022)
  - inference of primordial initial conditions: **Agocs**, Hergt, et al., *Phys Rev D* (2020), Letey, Shumaylov, **Agocs**, et al. (2022)

Current applications

- Cosmology (previously unable to investigate these models because of costly oscillatory solve)
  - closed universe models: Hergt, **Agocs**, et al., *Phys Rev D* (2022)
  - inference of primordial initial conditions: **Agocs**, Hergt, et al., *Phys Rev D* (2020), Letey, Shumaylov, **Agocs**, et al. (2022)
- Evaluation of special functions (e.g. Legendre polynomials of high order)

Current applications

- Cosmology (previously unable to investigate these models because of costly oscillatory solve)
  - closed universe models: Hergt, **Agocs**, et al., *Phys Rev D* (2022)
  - inference of primordial initial conditions: **Agocs**, Hergt, et al., *Phys Rev D* (2020), Letey, Shumaylov, **Agocs**, et al. (2022)
- Evaluation of special functions (e.g. Legendre polynomials of high order)
  - Possible because code is capable of *dense output*

Current applications

- Cosmology (previously unable to investigate these models because of costly oscillatory solve)
  - closed universe models: Hergt, **Agocs**, et al., *Phys Rev D* (2022)
  - inference of primordial initial conditions: **Agocs**, Hergt, et al., *Phys Rev D* (2020), Letey, Shumaylov, **Agocs**, et al. (2022)
- Evaluation of special functions (e.g. Legendre polynomials of high order)
  - Possible because code is capable of *dense output*
  - $\nu = 10^1$-10^9, solve: $\mathcal{O}(10^{-3})$ s, eval/dense output: $\mathcal{O}(10^{-6})$ s/point on a laptop, single core

Current applications

- Cosmology (previously unable to investigate these models because of costly oscillatory solve)
  - closed universe models: Hergt, **Agocs**, et al., *Phys Rev D* (2022)
  - inference of primordial initial conditions: **Agocs**, Hergt, et al., *Phys Rev D* (2020), Letey, Shumaylov, **Agocs**, et al. (2022)
- Evaluation of special functions (e.g. Legendre polynomials of high order)
  - Possible because code is capable of *dense output*
  - $\nu = 10^1$-10^9, solve: $\mathcal{O}(10^{-3})$ s, eval/dense output: $\mathcal{O}(10^{-6})$ s/point on a laptop, single core
- Quadrature of highly oscillatory functions (work in progress)

Software

Software

- Open-source, unit tested, documented, with executable tutorials

Software

- Open-source, unit tested, documented, with executable tutorials
- Easy install: `pip` or `conda(-forge)`

Software

- Open-source, unit tested, documented, with executable tutorials
- Easy install: `pip` or `conda(-forge)`
- Published in JOSS (Journal of open-source software)

Riccati

Future outlook & conclusions

Future outlook & conclusions

- An efficient method for solving linear, 2nd order ODEs, with a frequency term that may be large

Future outlook & conclusions

- An efficient method for solving linear, 2nd order ODEs, with a frequency term that may be large
- Unique: asymptotic methods applied numerically, spectral accuracy, can deal with oscillatory or slowly-varying regions, works in presence of friction term

Future outlook & conclusions

- An efficient method for solving linear, 2nd order ODEs, with a frequency term that may be large
- Unique: asymptotic methods applied numerically, spectral accuracy, can deal with oscillatory or slowly-varying regions, works in presence of friction term
- Asymptotic expansions reduce the residual **very quickly**, up until a certain iteration/term

Future outlook & conclusions

- An efficient method for solving linear, 2nd order ODEs, with a frequency term that may be large
- Unique: asymptotic methods applied numerically, spectral accuracy, can deal with oscillatory or slowly-varying regions, works in presence of friction term
- Asymptotic expansions reduce the residual **very quickly**, up until a certain iteration/term
- Could we generalise the method to ODE systems? PDEs?

Thank you!

References I

F. J. **Agocs** and A. H. Barnett (2022). *An adaptive spectral method for oscillatory second-order linear ODEs with frequency-independent cost*. DOI: 10.48550/ARXIV.2212.06924. URL: https://arxiv.org/abs/2212.06924.

F. J. **Agocs**, W. J. Handley, A. N. Lasenby, and M. P. Hobson (2020). "Efficient method for solving highly oscillatory ordinary differential equations with applications to physical systems". In: *Phys Rev Research* 2.1, p. 013030.

F. J. **Agocs**, L. T. Hergt, W. J. Handley, A. N. Lasenby, and M. P. Hobson (2020). "Quantum initial conditions for inflation and canonical invariance". In: *Phys Rev D* 102.2. ISSN: 2470-0029. DOI: 10.1103/physrevd.102.023507. URL: http://dx.doi.org/10.1103/physrevd.102.023507.

J. Bremer (2018). "On the numerical solution of second order ordinary differential equations in the high-frequency regime". In: *ACHA* 44.2, pp. 312–349.

— (2023). "Phase function methods for second order linear ordinary differential equations with turning points". In: *ACHA* 65, pp. 137–169. ISSN: 1063-5203. DOI: https://doi.org/10.1016/j.acha.2023.02.005. URL: https://www.sciencedirect.com/science/article/pii/S1063520323000210.

Z. Heitman, J. Bremer, and V. Rokhlin (2015). "On the existence of nonoscillatory phase functions for second order ordinary differential equations in the high-frequency regime". In: *JCP* 290, pp. 1–27.

References II

- L. T. Hergt, F. J. **Agocs**, W. J. Handley, M. P. Hobson, and A. N. Lasenby (2022). "Finite inflation in curved space". In: *Phys Rev D* 106.6. ISSN: 2470-0029. DOI: 10.1103/physrevd.106.063529. URL: http://dx.doi.org/10.1103/physrevd.106.063529.
- J. Körner, A. Arnold, and K. Döpfner (2022). "WKB-based scheme with adaptive step size control for the Schrödinger equation in the highly oscillatory regime". In: *JCAM* 404, p. 113905.
- M. I. Letey, Z. Shumaylov, F. J. **Agocs**, W. J. Handley, M. P. Hobson, and A. N. Lasenby (2022). *Quantum Initial Conditions for Curved Inflating Universes*. arXiv: 2211.17248 [gr-qc].
- L. R. Petzold (1981). "An efficient numerical method for highly oscillatory ordinary differential equations". In: *SINUM* 18.3, pp. 455–479.
- L. N. Trefethen and D. Bau III (1997). *Numerical linear algebra*. Vol. 50. SIAM.

WKB expansion /1

- Alternatively, build nonoscillatory (approx) solution: WKB/Riccati defect correction
  - Wentzel–Kramers–Brillouin (WKB) expansion:
    Extract a small parameter $1/\omega_0$: let $\omega(t) = \omega_0 \Omega(t)$, $\omega_0 \gg 1$, $\Omega(t)$ unit size,

$$u''(t) + \omega_0^2 \Omega(t)^2 u(t) = 0$$

for both real and imag ω, u has exp behavior, so transform as $z(t) = e^{\omega_0 z(t)}$, $z'(t) = x(t)$,

$$x' + \omega_0 x^2 + \omega_0 \Omega^2 = 0,$$

then expand as power series in small param,

$$x_j(t) = \sum_{l=0}^{j} \omega_0^{-l} s_l(t)$$

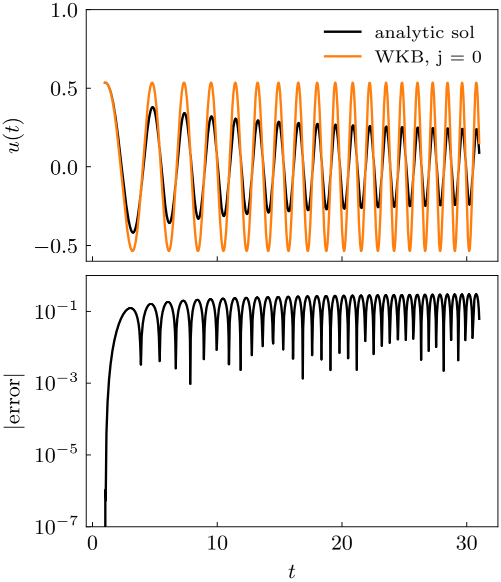
match powers of ω_0, then "reabsorb": set $\omega_0 = 1$. Get

$$s_0 = \pm i\omega, \quad s_{l+1} = -\frac{1}{2s_0}\left(s_l' + \sum_{k=1}^{l} s_k s_{l+1-k}\right),$$

WKB expansion /2

- Usually applied analytically, used in quantum mechanics
- Recursion relation involves all previous terms $\rightarrow$ hard to analyze
- Asymptotic
- First few iterations of series (start from $+i\omega$):

$$
\begin{aligned}
x_0 &= i\omega, \\
x_1 &= i\omega - \frac{\omega'}{2\omega}, \\
x_2 &= i\omega - \frac{\omega'}{2\omega} + i\frac{3\omega'^2}{\omega^3} - i\frac{\omega''}{4\omega^2}
\end{aligned}
$$



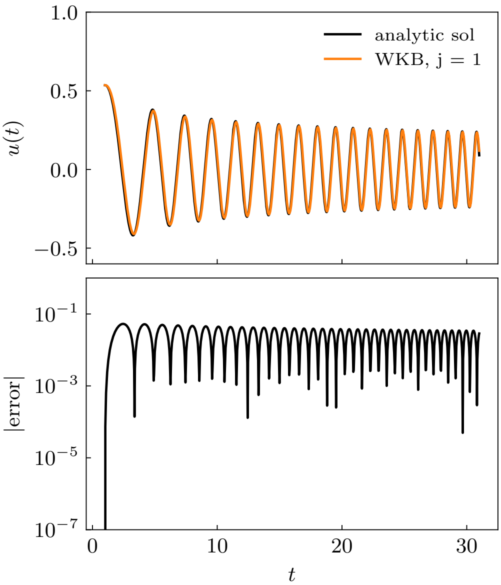
WKB expansion /2

- Usually applied analytically, used in quantum mechanics
- Recursion relation involves all previous terms $\rightarrow$ hard to analyze
- Asymptotic
- First few iterations of series (start from $+i\omega$):

$$x_0 = i\omega,$$

$$x_1 = i\omega - \frac{\omega'}{2\omega},$$

$$x_2 = i\omega - \frac{\omega'}{2\omega} + i\frac{3\omega'^2}{\omega^3} - i\frac{\omega''}{4\omega^2}$$



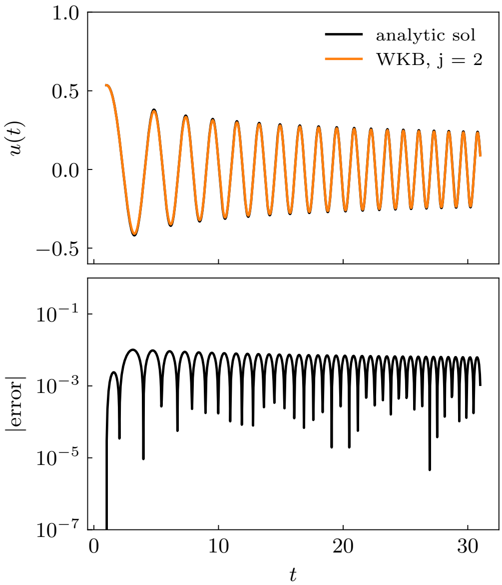
WKB expansion /2

- Usually applied analytically, used in quantum mechanics
- Recursion relation involves all previous terms $\rightarrow$ hard to analyze
- Asymptotic
- First few iterations of series (start from $+i\omega$):

$$x_0 = i\omega,$$

$$x_1 = i\omega - \frac{\omega'}{2\omega},$$

$$x_2 = i\omega - \frac{\omega'}{2\omega} + i\frac{3\omega'^2}{\omega^3} - i\frac{\omega''}{4\omega^2}$$



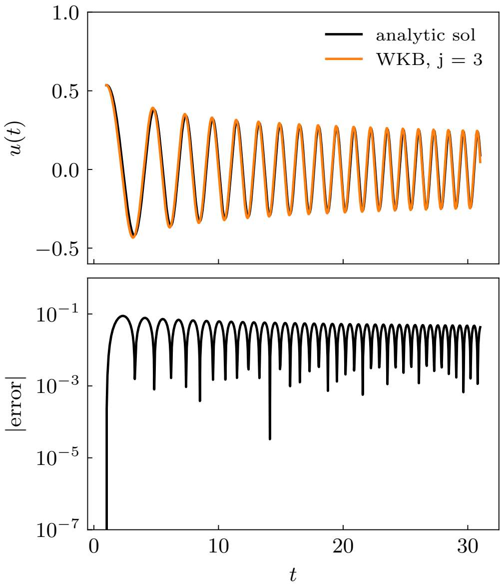
WKB expansion /2

- Usually applied analytically, used in quantum mechanics
- Recursion relation involves all previous terms $\rightarrow$ hard to analyze
- Asymptotic
- First few iterations of series (start from $+i\omega$):

$$x_0 = i\omega,$$

$$x_1 = i\omega - \frac{\omega'}{2\omega},$$

$$x_2 = i\omega - \frac{\omega'}{2\omega} + i\frac{3\omega'^2}{\omega^3} - i\frac{\omega''}{4\omega^2}$$



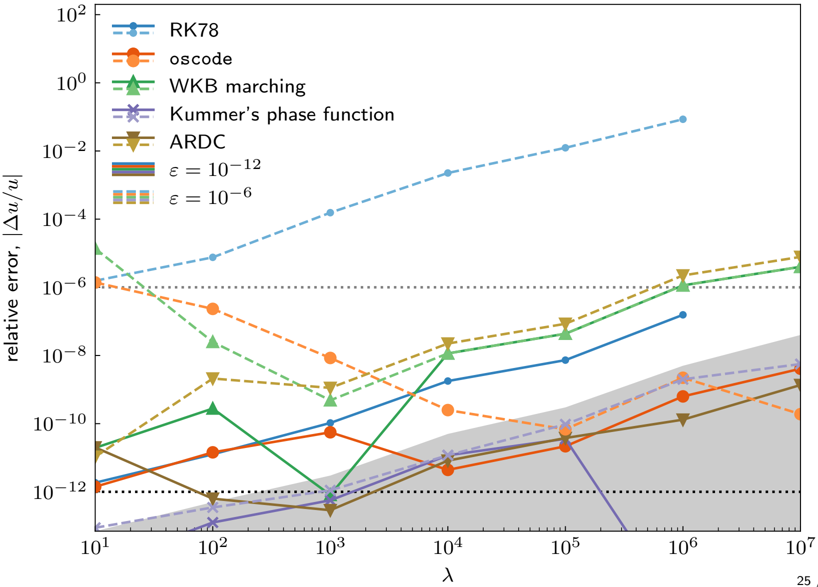
Some state-of-the-art oscillatory solvers

	ARDC/this work **Agocs** and Barnett (2022)	Kummer's phase function method Bremer, *ACHA* (2018)	`oscode` **Agocs**, Handley, et al., *Phys Rev Research* (2020)	WKB marching Körner et al., *JCAM* (2022)
high-order?	✓	✓	✗	✗
γ?	✓	✗[5]	✓	✗
code?	Python	Fortran 90	Python/C++	MATLAB
misc				need ω', ω'', $\ldots$, $\frac{\mathrm{d}^5\omega}{\mathrm{d}t^5}$

[5] Bremer, *ACHA* (2023) can be applied in this case, but no code $\rightarrow$ no comparison.

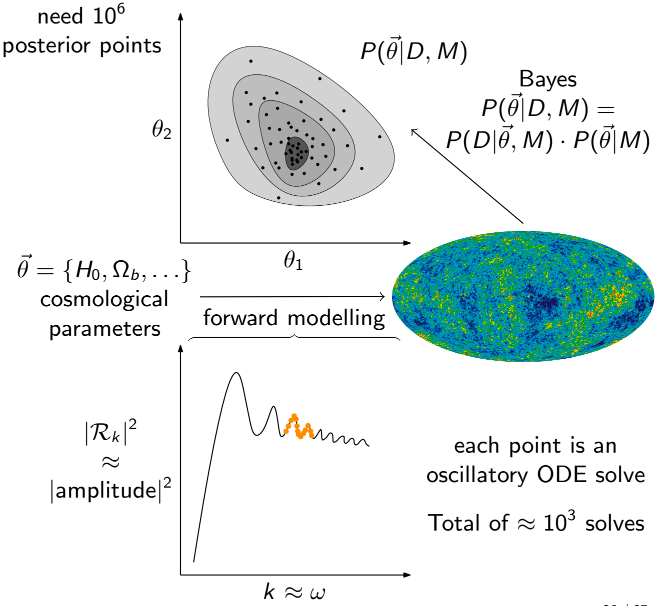
Comparison with standard & state-of-the-art solvers, convergence

- We used the Kummer's phase function method to compute a reference solution, therefore its reported accuracy (relative to spectral deferred correction), in **grey shading**, is an **upper limit** on the error



Motivation

- This ODE is extremely common in physics and math
  - inflationary cosmology
    - $\approx 10^9$ oscillatory ODE solves
  - 1D quantum mechanics
  - plasma physics, Hamiltonian dynamics, particle accelerators, electric circuits, acoustic and gravitational waves, ...
  - special function evaluation



The nonoscillatory phase function /2

- Most solutions for the Riccati equation ($x(t)$) oscillate with 2ω

The nonoscillatory phase function /2

- Most solutions for the Riccati equation ($x(t)$) oscillate with 2ω
- Example: $\omega = \omega_0$, analytic solution: $x(t) = \omega_0 \tan(\beta - \omega_0 t)$, only $\mathrm{Im}\beta \to \pm\infty$ gives nonosc. $x(t) = \pm i\omega_0$

The nonoscillatory phase function /2

- Most solutions for the Riccati equation ($x(t)$) oscillate with 2ω
- Example: $\omega = \omega_0$, analytic solution: $x(t) = \omega_0 \tan(\beta - \omega_0 t)$, only $\mathrm{Im}\beta \to \pm\infty$ gives nonosc. $x(t) = \pm i\omega_0$