Numerical Simulation of Dynamic Systems V

Prof. Dr. François E. Cellier
Department of Computer Science
ETH Zurich

March 12, 2013

The Accuracy Domain

We already noticed that the numerical stability of an algorithm can be expressed in the complex $\boxed{\lambda \cdot h}$ plane. We furthermore saw that a *numerically stable* solution isn't necessarily also an *accurate* solution.

We would now like to investigate if it is possible to obtain something like an *accuracy domain* similar to the *numerical stability domain*.

We shall start with the linear system:

$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} \quad ; \quad \mathbf{x}(t_0) = \mathbf{x_0}$$

using the same **A**-matrix that we had been using before in the stability analysis:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 \\ -1 & 2\cos(\alpha) \end{pmatrix}$$

This matrix exhibits two eigenvalues on the unit circle forming an angle α with the negative real axis.

We use the normalized initial conditions:

$$\mathbf{x_0} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

The Accuracy Domain II

The analytical solution can easily be found:

$$\mathbf{x_{anal}} = \exp(\mathbf{A} \cdot (t - t_0)) \cdot \mathbf{x_o}$$

A numerical solution can be obtained using any one of the previously introduced numerical ODE solvers, such as the RK4 algorithm:

```
function [x] = rk4(A, h, x0)
    h2 = h/2;  h6 = h/6;
    x(:,1) = x0;
    for i = 1 : 10/h,
        xx = x(:,i);
        k1 = A * xx;
        k2 = A * (xx + h2 * k1);
        k3 = A * (xx + h2 * k2);
        k4 = A * (xx + h * k3);
        x(:,i+1) = xx + h6 * (k1 + 2 * k2 + 2 * k3 + k4);
    end
return
```

The Accuracy Domain III

We simulate across 10 seconds and compute the *global error*:

$$\varepsilon_{\text{global}} = \|\mathbf{x}_{\mathbf{anal}} - \mathbf{x}_{\mathbf{simul}}\|_\infty$$

We iterate over the integration step size, h, until the global error stays below a specified threshold value, *tol*:

$$\varepsilon_{\text{global}} \leq tol$$

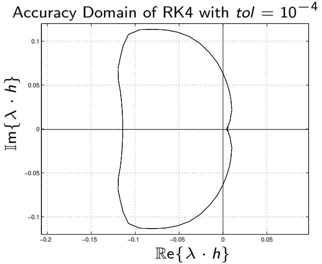


Figure: Accuracy domain of RK4 with $tol = 10^{-4}$

The Accuracy Domain IV

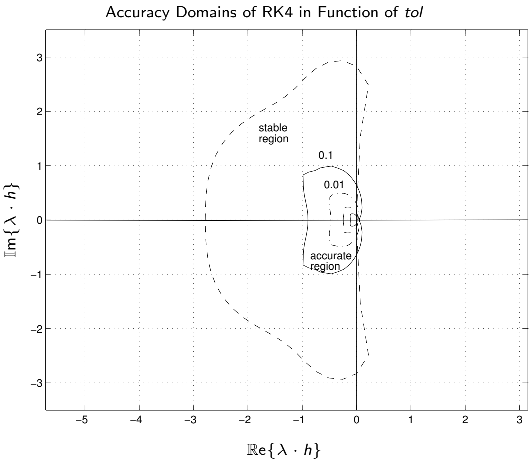


Figure: Accuracy domains of RK4

The Accuracy Domain V

Unfortunately, the accuracy domain is not independent of the performed experiment. It depends significantly on the *initial conditions* that we are using.

The Accuracy Domain V

Unfortunately, the accuracy domain is not independent of the performed experiment. It depends significantly on the *initial conditions* that we are using.

We need something better.

Simulation Efficiency

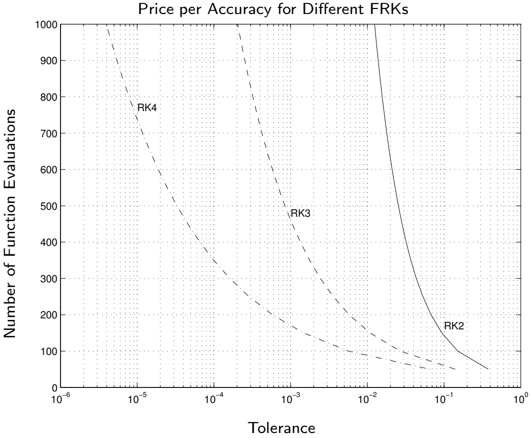


Figure: Simulation efficiency of different FRK algorithms

Damping Factor and Oscillation Frequency

Given the *linear continuous-time system*:

$$\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} \quad ; \quad \mathbf{x}(t_0) = \mathbf{x_0}$$

The *linear discrete-time system*:

$$\mathbf{x_{k+1}} = \mathbf{F_{anal}} \cdot \mathbf{x_k} \quad ; \quad \mathbf{x}(t_0) = \mathbf{x_0}$$

with:

$$\mathbf{F_{anal}} = \exp(\mathbf{A} \cdot h)$$

has the identical solution as the continuous-time system at the *sampling instants*, $k \cdot h$.

Therefore:

$$\mathrm{Eig}\{\mathbf{F_{anal}}\} = \exp(\mathrm{Eig}\{\mathbf{A}\} \cdot h)$$

Every eigenvalue of the discrete-time system corresponds to an eigenvalue of the continuous-time system:

$$\lambda_{disc} = \exp(\lambda_{cont} \cdot h) = \exp((-\sigma + j \cdot \omega) \cdot h) = \exp(-\sigma \cdot h) \cdot \exp(j \cdot \omega \cdot h)$$

Damping Factor and Oscillation Frequency II

We introduce a new complex plane:

$$z = \exp(\lambda \cdot h)$$

Control engineers know this plane very well.

Damping Factor and Oscillation Frequency II

We introduce a new complex plane:

$$z = \exp(\lambda \cdot h)$$

Control engineers know this plane very well.

We introduce also a *discrete damping factor*, σ_d, and a *discrete oscillation frequency*, ω_d:

$$\begin{aligned} \sigma_d &= \sigma \cdot h \\ \omega_d &= \omega \cdot h \end{aligned}$$

Damping Factor and Oscillation Frequency II

We introduce a new complex plane:

$$z = \exp(\lambda \cdot h)$$

Control engineers know this plane very well.

We introduce also a *discrete damping factor*, σ_d, and a *discrete oscillation frequency*, ω_d:

$$\begin{aligned} \sigma_d &= \sigma \cdot h \\ \omega_d &= \omega \cdot h \end{aligned}$$

Therefore:

$$\begin{aligned} |z| &= \exp(-\sigma_d) \\ \angle z &= \omega_d \end{aligned}$$

Damping Factor and Oscillation Frequency II

We introduce a new complex plane:

$$z = \exp(\lambda \cdot h)$$

Control engineers know this plane very well.

We introduce also a *discrete damping factor*, σ_d, and a *discrete oscillation frequency*, ω_d:

$$\begin{aligned}\sigma_d &= \sigma \cdot h \\ \omega_d &= \omega \cdot h\end{aligned}$$

Therefore:

$$\begin{aligned}|z| &= \exp(-\sigma_d) \\ \angle z &= \omega_d\end{aligned}$$

The values σ_d and ω_d are the discrete damping factor and the discrete oscillation frequency that we would expect to see if the simulation of the model were to be performed analytically.

Damping Factor and Oscillation Frequency III

In reality, we perform a numerical simulation. Its $\mathbf{F}_{\mathbf{simul}}$-matrix approximates the $\mathbf{F}_{\mathbf{anal}}$-matrix of the analytical solution.

Damping Factor and Oscillation Frequency III

In reality, we perform a numerical simulation. Its $\mathbf{F}_{\mathbf{simul}}$-matrix approximates the $\mathbf{F}_{\mathbf{anal}}$-matrix of the analytical solution.

Consequently, we can define for the $\mathbf{F}_{\mathbf{simul}}$-matrix:

$$\hat{z} = \exp(\hat{\lambda}_d)$$

with:

$$\hat{\lambda}_d = -\hat{\sigma}_d + j \cdot \hat{\omega}_d$$

and therefore:

$$\begin{aligned} |\hat{z}| &= \exp(-\hat{\sigma}_d) \\ \angle\hat{z} &= \hat{\omega}_d \end{aligned}$$

Damping Factor and Oscillation Frequency III

In reality, we perform a numerical simulation. Its $\mathbf{F}_{\mathbf{simul}}$-matrix approximates the $\mathbf{F}_{\mathbf{anal}}$-matrix of the analytical solution.

Consequently, we can define for the $\mathbf{F}_{\mathbf{simul}}$-matrix:

$$\hat{z} = \exp(\hat{\lambda}_d)$$

with:

$$\hat{\lambda}_d = -\hat{\sigma}_d + j \cdot \hat{\omega}_d$$

and therefore:

$$\begin{aligned} |\hat{z}| &= \exp(-\hat{\sigma}_d) \\ \angle\hat{z} &= \hat{\omega}_d \end{aligned}$$

The variable $\hat{\sigma}_d$ approximates σ_d, and the variable $\hat{\omega}_d$ approximates ω_d.

The Damping Plot

Consequently, it makes sense to introduce the *damping error*, ε_σ, and the *frequency error*, ε_ω:

$$\begin{aligned} \varepsilon_\sigma &= \sigma_d - \hat{\sigma}_d \\ \varepsilon_\omega &= \omega_d - \hat{\omega}_d \end{aligned}$$

The Damping Plot

Consequently, it makes sense to introduce the *damping error*, ε_σ, and the *frequency error*, ε_ω:

$$
\begin{aligned}
\varepsilon_\sigma &= \sigma_d - \hat{\sigma}_d \\
\varepsilon_\omega &= \omega_d - \hat{\omega}_d
\end{aligned}
$$

We can plot the *numerical damping*, $\hat{\sigma}_d$, in function of the *analytical damping*, σ_d. This graph is called *damping plot*.

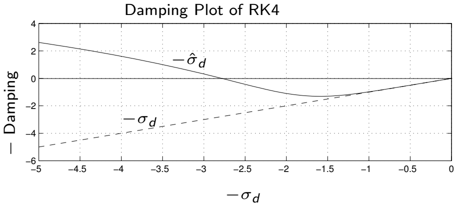


Figure: Damping plot of RK4

The Damping Plot II

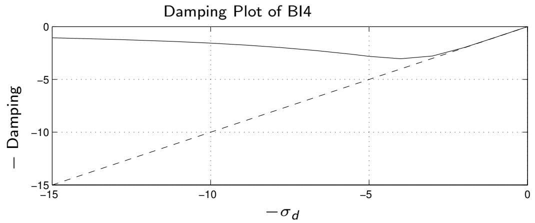
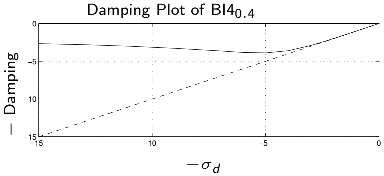


Figure: Damping plot of BI4

This algorithm doesn't lose its stability, but its damping factor at infinity assumes a value of zero instead of infinite.

This is an *F-stable* algorithm. All F-stable algorithms have this property in common.

The Damping Plot III

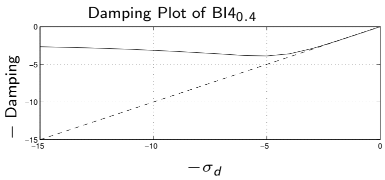


This algorithm doesn't lose its stability, but its damping factor at infinity is:

$$\hat{\sigma}_d(-\infty) = -4 \cdot \log\left(\frac{\vartheta}{1-\vartheta}\right)$$

This is an *A-stable but not L-stable* algorithm.

The Damping Plot III



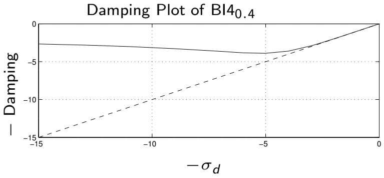
This algorithm doesn't lose its stability, but its damping factor at infinity is:

$$\hat{\sigma}_d(-\infty) = -4 \cdot \log\left(\frac{\vartheta}{1-\vartheta}\right)$$

This is an *A-stable but not L-stable* algorithm.

- The damping is infinite in the case of the BRK4 algorithm with $\vartheta = 0$. The BRK4 algorithm is *L-stable*.

The Damping Plot III



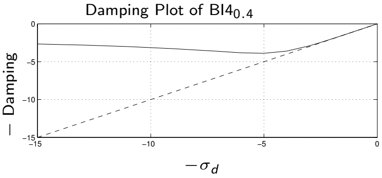
This algorithm doesn't lose its stability, but its damping factor at infinity is:

$$\hat{\sigma}_d(-\infty) = -4 \cdot \log\left(\frac{\vartheta}{1-\vartheta}\right)$$

This is an *A-stable but not L-stable* algorithm.

- The damping is infinite in the case of the BRK4 algorithm with $\vartheta = 0$. The BRK4 algorithm is *L-stable*.
- The damping is zero in the case of the BI4 algorithm with $\vartheta = 0.5$. The BI4 algorithm is *F-stable*.

The Damping Plot III



This algorithm doesn't lose its stability, but its damping factor at infinity is:

$$\hat{\sigma}_d(-\infty) = -4 \cdot \log\left(\frac{\vartheta}{1-\vartheta}\right)$$

This is an *A-stable but not L-stable* algorithm.

- The damping is infinite in the case of the BRK4 algorithm with $\vartheta = 0$. The BRK4 algorithm is *L-stable*.
- The damping is zero in the case of the BI4 algorithm with $\vartheta = 0.5$. The BI4 algorithm is *F-stable*.
- The damping is negative in the case of $\vartheta > 0.5$. These algorithms lose their numerical stability, i.e., their numerical stability domains loop in the left-half complex $\boxed{\lambda \cdot h}$ plane.

The Damping Plot IV

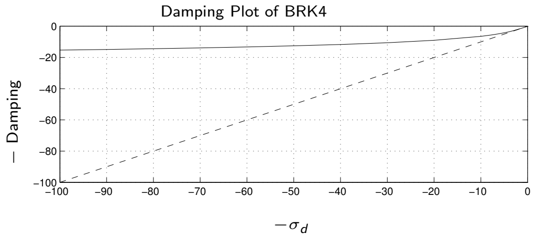


Figure: Damping plot of BRK4

The BRK4 algorithm is *L-stable*. Therefore, the damping grows to infinity.

However, the damping of the numerical simulation algorithm grows much more slowly than that of the analytical simulation.

The Damping Plot V

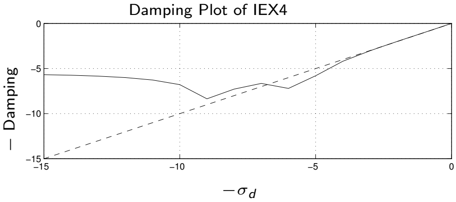


Figure: Damping plot of IEX4

In the case of the IEX4 algorithm, strange things happen that we need to understand better. Its **F**-matrix is:

$$\begin{aligned}\mathbf{F} \quad = \quad & -\frac{1}{6} \cdot [\mathbf{I^{(n)}} - \mathbf{A} \cdot h]^{-1} + 4 \cdot [\mathbf{I^{(n)}} - \frac{\mathbf{A} \cdot h}{2}]^{-2} \\ & - \frac{27}{2} \cdot [\mathbf{I^{(n)}} - \frac{\mathbf{A} \cdot h}{3}]^{-3} + \frac{32}{3} \cdot [\mathbf{I^{(n)}} - \frac{\mathbf{A} \cdot h}{4}]^{-4}\end{aligned}$$

The Damping Order Star

We can analyze the scalar case with:

$$q = \lambda \cdot h$$

We obtain:

$$f = -\frac{1}{6} \cdot \frac{1}{1-q} + 4 \cdot \frac{1}{(1-q/2)^2} - \frac{27}{2} \cdot \frac{1}{(1-q/3)^3} + \frac{32}{3} \cdot \frac{1}{(1-q/4)^4}$$

Therefore:

$$\hat{\sigma}_d = -\log(|f|)$$

This equation has a solution for all complex values of q, not only for $q = -\sigma_d$.

The Damping Order Star II

Let us draw the *damping error*, ε_σ, in function of σ_d and ω_d:

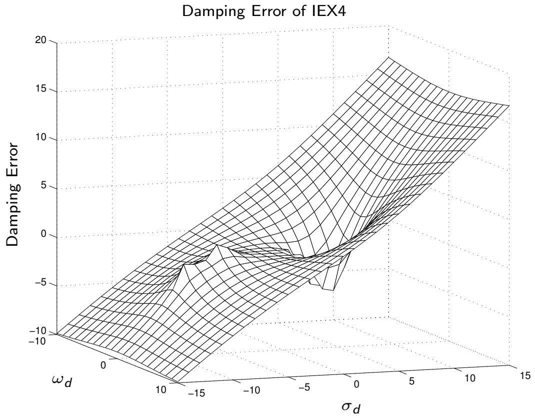
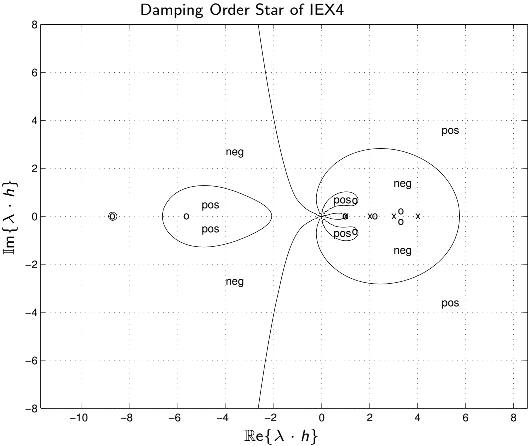


Figure: Damping error of IEX4

The Damping Order Star III

We can draw a graph with all the points, where the damping error is zero. This graph is called *"order star"*.



The Damping Order Star IV

The function:

$$f = -\frac{1}{6} \cdot \frac{1}{1-q} + 4 \cdot \frac{1}{(1-q/2)^2} - \frac{27}{2} \cdot \frac{1}{(1-q/3)^3} + \frac{32}{3} \cdot \frac{1}{(1-q/4)^4}$$

is a *strictly proper rational function*. It has 10 poles and 9 zeros.

The Damping Order Star IV

The function:

$$f = -\frac{1}{6} \cdot \frac{1}{1-q} + 4 \cdot \frac{1}{(1-q/2)^2} - \frac{27}{2} \cdot \frac{1}{(1-q/3)^3} + \frac{32}{3} \cdot \frac{1}{(1-q/4)^4}$$

is a *strictly proper rational function*. It has 10 poles and 9 zeros.

The damping of the poles is $-\infty$. For this reason, useful simulation methods must have all of their poles in the right-half complex $\boxed{\lambda \cdot h}$ plane.

The Damping Order Star IV

The function:

$$f = -\frac{1}{6} \cdot \frac{1}{1-q} + 4 \cdot \frac{1}{(1-q/2)^2} - \frac{27}{2} \cdot \frac{1}{(1-q/3)^3} + \frac{32}{3} \cdot \frac{1}{(1-q/4)^4}$$

is a *strictly proper rational function*. It has 10 poles and 9 zeros.

The damping of the poles is $-\infty$. For this reason, useful simulation methods must have all of their poles in the right-half complex $\boxed{\lambda \cdot h}$ plane.

The damping of the zeros is $+\infty$. Zeros are therefore useful in the left-half complex $\boxed{\lambda \cdot h}$ plane.

The Damping Order Star IV

The function:

$$f = -\frac{1}{6} \cdot \frac{1}{1-q} + 4 \cdot \frac{1}{(1-q/2)^2} - \frac{27}{2} \cdot \frac{1}{(1-q/3)^3} + \frac{32}{3} \cdot \frac{1}{(1-q/4)^4}$$

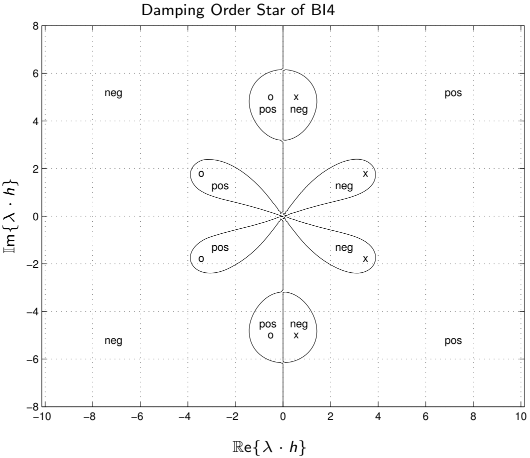
is a *strictly proper rational function*. It has 10 poles and 9 zeros.

The damping of the poles is $-\infty$. For this reason, useful simulation methods must have all of their poles in the right-half complex $\boxed{\lambda \cdot h}$ plane.

The damping of the zeros is $+\infty$. Zeros are therefore useful in the left-half complex $\boxed{\lambda \cdot h}$ plane.

In the proximity of the origin, we cannot accept either poles or zeros. At least, we cannot accept them in the left-half complex $\boxed{\lambda \cdot h}$ plane.

The Damping Order Star V



The damping order stars of F-stable methods are symmetric to the imaginary axis due to the symmetry of their poles and zeros.

The Frequency Plot

We can also analyze the frequency error. We can draw the *discrete numerical frequency*, $\hat{\omega}_d$, in function of the *discrete analytical frequency*, ω_d.

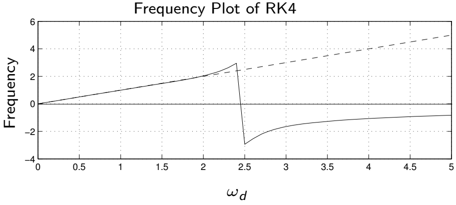
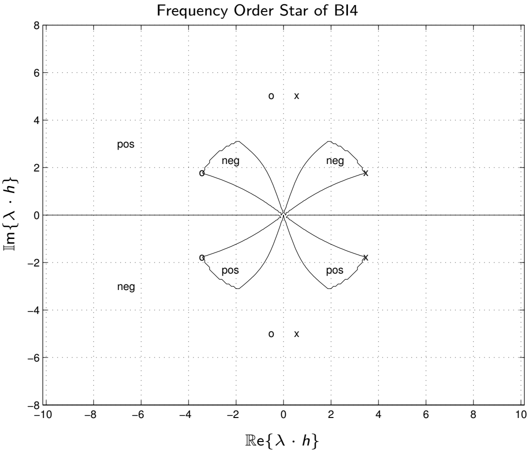


Figure: Frequency plot of RK4

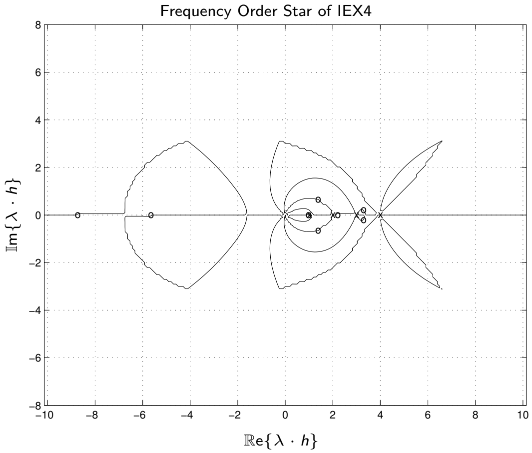
The frequency is 2π-periodic. Yet, the frequency error is only of interest in the proximity of the origin. Therefore, its periodicity doesn't bother us much.

The Frequency Order Star

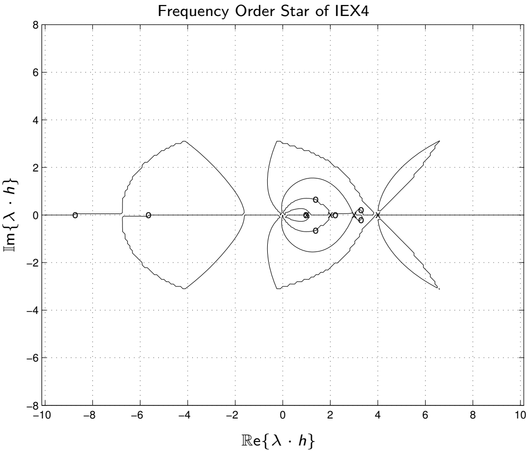
It is also possible to draw the *frequency error*, ε_ω, in function of σ_d and ω_d. We thus can draw a *frequency order star*:



The Frequency Order Star II



The Frequency Order Star II



I drew these frequency order stars . . . because they are beautiful.

The Asymptotic Regions

Let us look once more at the damping and frequency plots. There are regions, where the damping and frequency errors are very small. These regions are called *asymptotic regions*.

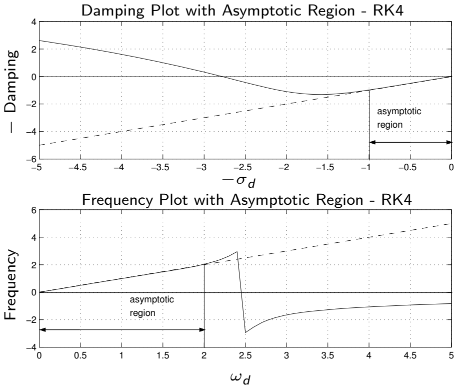


Figure: Asymptotic regions of RK4

The Accuracy Domain VI

It may make sense to define a combined error consisting of the damping and frequency errors:

$$\mathit{os}_{err} = |\sigma_d - \hat{\sigma}_d| + |\omega_d - \hat{\omega}_d|$$

The Accuracy Domain VI

It may make sense to define a combined error consisting of the damping and frequency errors:

$$os_{err} = |\sigma_d - \hat{\sigma}_d| + |\omega_d - \hat{\omega}_d|$$

We can encounter, in the two order stars (of damping and frequency), a region around the origin, where os_{err} stays smaller than a given threshold, tol:

$$os_{err} \leq tol$$

The Accuracy Domain VI

It may make sense to define a combined error consisting of the damping and frequency errors:

$$os_{err} = |\sigma_d - \hat{\sigma}_d| + |\omega_d - \hat{\omega}_d|$$

We can encounter, in the two order stars (of damping and frequency), a region around the origin, where os_{err} stays smaller than a given threshold, tol:

$$os_{err} \leq tol$$

This region can also be used as an *accuracy domain*.

The Accuracy Domain VI

It may make sense to define a combined error consisting of the damping and frequency errors:

$$os_{err} = |\sigma_d - \hat{\sigma}_d| + |\omega_d - \hat{\omega}_d|$$

We can encounter, in the two order stars (of damping and frequency), a region around the origin, where os_{err} stays smaller than a given threshold, tol:

$$os_{err} \leq tol$$

This region can also be used as an *accuracy domain*.

However, this new definition of the accuracy domain is much more useful than the one offered previously, because it doesn't depend on any experiment.

The Accuracy Domain VII

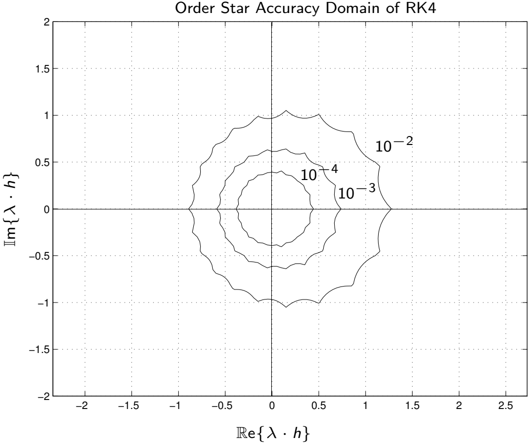


Figure: Order star accuracy domain of RK4

The Accuracy Domain VIII

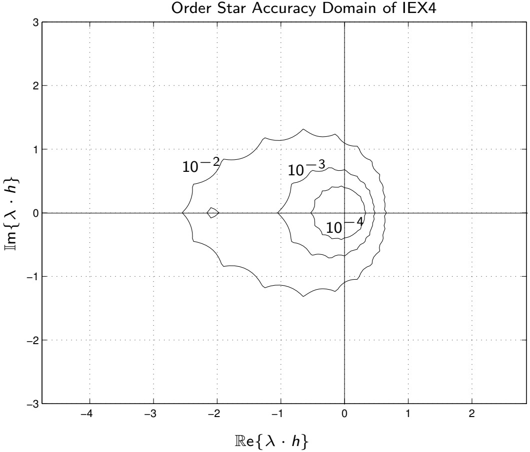


Figure: Order star accuracy domain of IEX4

Integration Step-size Control

To guarantee the quality of the numerical simulation results, we have the choice of controlling either the *integration step size* or the *order of approximation accuracy*. It is more common, in the case of the RK algorithms, to control the integration step size.

Integration Step-size Control

To guarantee the quality of the numerical simulation results, we have the choice of controlling either the *integration step size* or the *order of approximation accuracy*. It is more common, in the case of the RK algorithms, to control the integration step size.

In order to control the integration step size, we need to estimate the *local integration error*.

Integration Step-size Control

To guarantee the quality of the numerical simulation results, we have the choice of controlling either the *integration step size* or the *order of approximation accuracy*. It is more common, in the case of the RK algorithms, to control the integration step size.

In order to control the integration step size, we need to estimate the *local integration error*.

One way of accomplishing this is to repeat the same step twice using two different integration algorithms.

Integration Step-size Control

To guarantee the quality of the numerical simulation results, we have the choice of controlling either the *integration step size* or the *order of approximation accuracy*. It is more common, in the case of the RK algorithms, to control the integration step size.

In order to control the integration step size, we need to estimate the *local integration error*.

One way of accomplishing this is to repeat the same step twice using two different integration algorithms.

Assuming that the two algorithms don't produce (by chance) the same erroneous result, we may implement the following algorithm:

$$\varepsilon_{\mathrm{rel}} = \frac{|x_1 - x_2|}{|x_1|}$$

$$\textbf{if } \varepsilon_{\mathrm{rel}} > tol_{\mathrm{rel}} \quad \Rightarrow \quad h_{\mathrm{new}} = 0.5 \cdot h$$

$$\textbf{if } \varepsilon_{\mathrm{rel}} < 0.5 \cdot tol_{\mathrm{rel}} \quad \text{during four steps} \quad \Rightarrow \quad h_{\mathrm{new}} = 1.5 \cdot h$$

Integration Step-size Control II

The above algorithm is a bit risky, because it may happen that $x_1 = 0$ at a given instant of time.

Integration Step-size Control II

The above algorithm is a bit risky, because it may happen that $x_1 = 0$ at a given instant of time.

Therefore, it may be better to use an improved formula for estimating the relative error, $\varepsilon_{\mathrm{rel}}$:

$$\varepsilon_{\mathrm{rel}} = \frac{|x_1 - x_2|}{\max(|x_1|, |x_2|, \delta)}$$

where $\delta = 10^{-10}$ is a very small (fudge) constant.

Integration Step-size Control II

The above algorithm is a bit risky, because it may happen that $x_1 = 0$ at a given instant of time.

Therefore, it may be better to use an improved formula for estimating the relative error, $\varepsilon_{\mathrm{rel}}$:

$$\varepsilon_{\mathrm{rel}} = \frac{|x_1 - x_2|}{\max(|x_1|, |x_2|, \delta)}$$

where $\delta = 10^{-10}$ is a very small (fudge) constant.

However, it is not very efficient to repeat the entire step twice just for the purpose of obtaining an estimate of the local integration error.

Integration Step-size Control II

The above algorithm is a bit risky, because it may happen that $x_1 = 0$ at a given instant of time.

Therefore, it may be better to use an improved formula for estimating the relative error, $\varepsilon_{\mathrm{rel}}$:

$$\varepsilon_{\mathrm{rel}} = \frac{|x_1 - x_2|}{\max(|x_1|, |x_2|, \delta)}$$

where $\delta = 10^{-10}$ is a very small (fudge) constant.

However, it is not very efficient to repeat the entire step twice just for the purpose of obtaining an estimate of the local integration error.

Meanwhile there exist *encapsulated codes*, where two different algorithms share a number of stages.

The Code RKF4/5

The code *Runge-Kutta-Fehlberg 4/5 (RKF4/5)* is one of these encapsulated codes. The method is characterized by the Butcher table:

0	0	0	0	0	0	0
1/4	1/4	0	0	0	0	0
3/8	3/32	9/32	0	0	0	0
12/13	1932/2197	−7200/2197	7296/2197	0	0	0
1	439/216	-8	3680/513	−845/4104	0	0
1/2	−8/27	2	−3544/2565	1859/4104	−11/40	0
x_1	25/216	0	1408/2565	2197/4104	−1/5	0
x_2	16/135	0	6656/12825	28561/56430	−9/50	2/55

The code contains an RK4 algorithm in five stages and another RK5 algorithm in six stages:

$$f_1(q) = 1 + q + \frac{1}{2}q^2 + \frac{1}{6}q^3 + \frac{1}{24}q^4 + \frac{1}{104}q^5$$

$$f_2(q) = 1 + q + \frac{1}{2}q^2 + \frac{1}{6}q^3 + \frac{1}{24}q^4 + \frac{1}{120}q^5 + \frac{1}{2080}q^6$$

The Code RKF4/5 II

Therefore:

$$\varepsilon(q) = f_1(q) - f_2(q) = \frac{1}{780} q^5 - \frac{1}{2080} q^6$$

and consequently:

$$\varepsilon \sim h^5$$

We conclude:

$$h \sim \sqrt[5]{\varepsilon}$$

The Code RKF4/5 II

Therefore:

$$\varepsilon(q) = f_1(q) - f_2(q) = \frac{1}{780}q^5 - \frac{1}{2080}q^6$$

and consequently:

$$\varepsilon \sim h^5$$

We conclude:

$$h \sim \sqrt[5]{\varepsilon}$$

We want:

$$tol_{\mathrm{rel}} = \frac{|x_1 - x_2|}{\max(|x_1|, |x_2|, \delta)}$$

and thus, it makes sense to propose:

$$h_{\mathrm{new}} = \sqrt[5]{\frac{tol_{\mathrm{rel}} \cdot \max(|x_1|, |x_2|, \delta)}{|x_1 - x_2|}} \cdot h_{\mathrm{old}}$$

The Code RKF4/5 II

Therefore:

$$\varepsilon(q) = f_1(q) - f_2(q) = \frac{1}{780}q^5 - \frac{1}{2080}q^6$$

and consequently:

$$\varepsilon \sim h^5$$

We conclude:

$$h \sim \sqrt[5]{\varepsilon}$$

We want:

$$tol_{\mathrm{rel}} = \frac{|x_1 - x_2|}{\max(|x_1|, |x_2|, \delta)}$$

and thus, it makes sense to propose:

$$h_{\mathrm{new}} = \sqrt[5]{\frac{tol_{\mathrm{rel}} \cdot \max(|x_1|, |x_2|, \delta)}{|x_1 - x_2|}} \cdot h_{\mathrm{old}}$$

In this way, if the error is too large, the *next step* is reduced, and if the error is unnecessarily small, the next step is increased. Steps are never repeated, even if the error is excessively large.

The Code RKF4/5 II

Therefore:

$$\varepsilon(q) = f_1(q) - f_2(q) = \frac{1}{780} q^5 - \frac{1}{2080} q^6$$

and consequently:

$$\varepsilon \sim h^5$$

We conclude:

$$h \sim \sqrt[5]{\varepsilon}$$

We want:

$$tol_{\mathrm{rel}} = \frac{|x_1 - x_2|}{\max(|x_1|, |x_2|, \delta)}$$

and thus, it makes sense to propose:

$$h_{\mathrm{new}} = \sqrt[5]{\frac{tol_{\mathrm{rel}} \cdot \max(|x_1|, |x_2|, \delta)}{|x_1 - x_2|}} \cdot h_{\mathrm{old}}$$

In this way, if the error is too large, the *next step* is reduced, and if the error is unnecessarily small, the next step is increased. Steps are never repeated, even if the error is excessively large.

RKF4/5 embraces thus an *optimistic control strategy*.

Integration Step-size Control III

We can interpret the problem of controlling the integration step size as a *discrete control problem*.

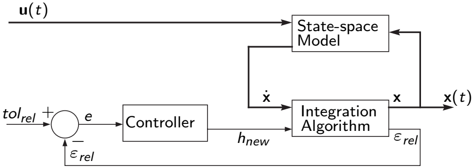


Figure: Step-size control viewed as a control problem

Integration Step-size Control IV

A *PI controller* was developed by Kjell Gustafsson in his Ph.D. dissertation:

$$h_{\text{new}} = \left(\frac{0.8 \cdot tol_{\text{rel}}}{\varepsilon_{\text{rel}_{\text{new}}}} \right)^{\frac{0.3}{n}} \cdot \left(\frac{\varepsilon_{\text{rel}_{\text{old}}}}{\varepsilon_{\text{rel}_{\text{new}}}} \right)^{\frac{0.4}{n}} \cdot h_{\text{old}}$$

where:

$$\begin{aligned} \varepsilon_{\text{rel}_{\text{new}}} &= \frac{\|\mathbf{x_1} - \mathbf{x_2}\|_\infty}{\max(\|\mathbf{x_1}\|_2, \|\mathbf{x_2}\|_2, \delta)} \\ \varepsilon_{\text{rel}_{\text{old}}} &= \text{same quantity one time step back} \end{aligned}$$

Conclusions

- In this presentation, we developed a *numerical accuracy treatment* for ODE solvers that is as solid as that of the *numerical stability treatment* introduced earlier.

Conclusions

- In this presentation, we developed a *numerical accuracy treatment* for ODE solvers that is as solid as that of the *numerical stability treatment* introduced earlier.

- We introduced a *frequency domain analysis* similar to the approach taken by control engineers in the discussion of discrete-time linear control systems.

Conclusions

- In this presentation, we developed a *numerical accuracy treatment* for ODE solvers that is as solid as that of the *numerical stability treatment* introduced earlier.
- We introduced a *frequency domain analysis* similar to the approach taken by control engineers in the discussion of discrete-time linear control systems.
- We designed *visualization methods for accuracy properties* of an ODE solver using *damping and frequency plots*. Furthermore, we showed the beautiful *damping and frequency order stars* of ODE solvers.

Conclusions

- In this presentation, we developed a *numerical accuracy treatment* for ODE solvers that is as solid as that of the *numerical stability treatment* introduced earlier.
- We introduced a *frequency domain analysis* similar to the approach taken by control engineers in the discussion of discrete-time linear control systems.
- We designed *visualization methods for accuracy properties* of an ODE solver using *damping and frequency plots*. Furthermore, we showed the beautiful *damping and frequency order stars* of ODE solvers.
- The presentation ended with a discussion of *integration step-size control* methods.