Computational Physics

Stability and operator exponential

Residual and error

Consider the initial value problem (IVP)

$$\mathbf{y}'(t) = A(t)\mathbf{y}(t) \qquad \mathbf{y}(0) = \mathbf{y}_0.$$

A numerical solver provides an approximate solution

$$\tilde{\mathbf{y}}(t) \approx \mathbf{y}(t).$$

The residual is the difference between true and numerical solution:

$$\mathbf{r}(t) = \mathbf{y}(t) - \tilde{\mathbf{y}}(t).$$

The norm of the residual is usually called the (absolute) error:

$$E = ||\mathbf{r}(t)|| = ||\mathbf{y}(t) - \tilde{\mathbf{y}}(t)||.$$

Boundedness of the error in time – stability

We now focus specifically on ODE-solvers and chose h as the (fixed) time step.

We assume the scalar problem

$$y'(t) = \alpha y(t) \qquad \text{with } y(0) = y_0 \quad \text{and} \quad \mathrm{Re}\{\alpha\} \leq 0.$$

Is it possible to guarantee that the absolute error remains finite with a fixed time step h even as $t \to \infty$?

Yes, in the given problem, we find that the solution is monotonically decaying: $|y(t + \Delta t)| \leq |y(t)|$.

Therefore the absolute error is bounded $|E(t)| \leq |y_0|$ whenever the numerical solution remains finite.

The fact that the *absolute* error remains finite is called *stability*.
Note: The *relative* error can still explode, the result might be useless!

Example: Euler's method

We assume that the ODE $y'(t) = \alpha y(t)$ is solved with Euler's method:

$$\tilde{y}(t_{n+1}) = \tilde{y}(t_n) + \underbrace{\alpha h}_{=z} \tilde{y}(t_n), \qquad \text{with } t_n = nh \quad \text{and} \quad \tilde{y}(0) = y_0.$$

Therefore, after n time steps, we find:

$$\tilde{y}(t_n) = (1+z)^n y_0.$$

This remains finite exactly if $|1+z| \leq 1$.

Euler's method is stable whenever $|1+z| = |1+h\alpha| \leq 1$.

Stability regions

This analysis os Euler's method can be generalized for any ODE solver that is based on doing n time steps with identical update.

The ODE is linear, therefore the update equation must be linear in $\tilde{y}(t)$ and can be written as:

$$\tilde{y}(t_{n+1}) = R(z)\tilde{y}(t_n),$$

with some function $R(z)$, that we call *stability function*.
The function $R(z)$ depends on the exact type of ODE solver.

After n time steps, we find:

$$\tilde{y}(t_n) = R(z)^n y_0.$$

Using the same argument as before, the method is stable whenever $|R(z)| \leq 1$.

Stability regions (cont'd)

Each numerical ODE solver had a characteristic contour in the complex plane that defines the values z for which it is stable.

Some examples (Euler's method and 2-4 order Runge-Kutta):

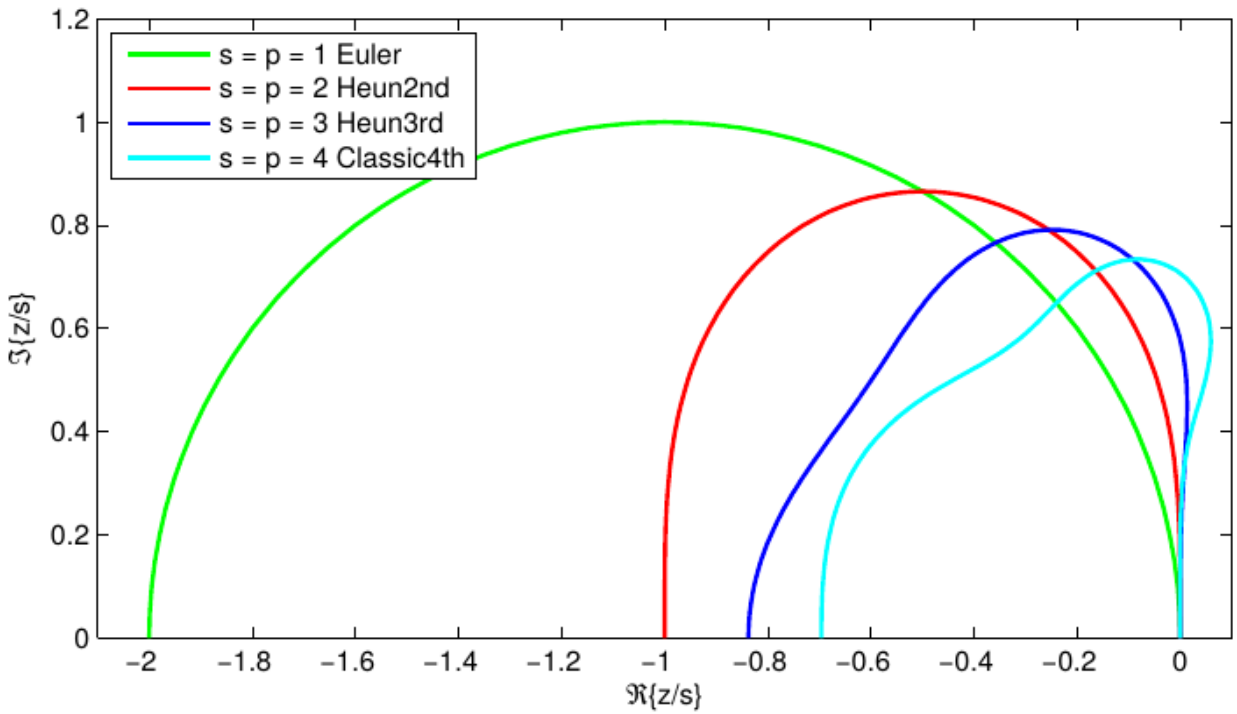
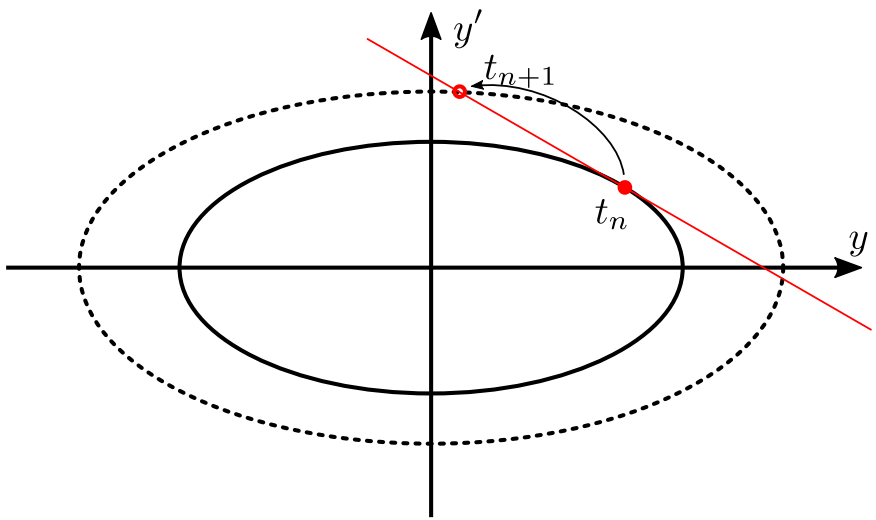


image: M.Sc. thesis Richard Diehl, KIT 2009

Visualisation of the instability for imaginary z

The instability of Euler's method for the undamped harmonic oscillator $y'' + y = 0$ can be nicely visualized in phase space.

The correct solution lives on a curve of constant energy in phase space. This curve is an ellipse.



Within each time step, both y and y' are updated by going a little along the tangent of the constant-energy contour. This lets us spiral aways from the origin, i.e. the total energy increases exponentially.

Time evolution of ODE systems with constant coefficients

Consider the vector-valued IVP

$$\mathbf{y}'(t) = A\mathbf{y}(t), \qquad \mathbf{y}(0) = \mathbf{y}_0,$$

with constant matrix A.

This is the form of problems without explicit time-dependence, i.e. in most linear problems in physics.

It is easily solved by diagonalizing A:

$$A = R^{-1}\Lambda R, \qquad \text{with } \Lambda \text{ diagonal; eigenvalues } \lambda_i.$$

$$\Rightarrow \begin{pmatrix} u_1'(t) \\ u_2'(t) \\ \vdots \\ u_n'(t) \end{pmatrix} = \begin{pmatrix} \lambda_1 u_1(t) \\ \lambda_2 u_2(t) \\ \vdots \\ \lambda_n u_n(t) \end{pmatrix} \qquad \text{with } \mathbf{u}(t) = R\mathbf{y}(t),$$

we obtain n decoupled scalar ODEs.

Time evolution of ODE systems with constant coefficients

$$\begin{pmatrix} u_1'(t) \\ u_2'(t) \\ \vdots \\ u_n'(t) \end{pmatrix} = \begin{pmatrix} \lambda_1 u_1(t) \\ \lambda_2 u_2(t) \\ \vdots \\ \lambda_n u_n(t) \end{pmatrix}, \qquad \text{with } \lambda_1 \text{ EVs of } A \text{ and } \mathbf{u}(t) = R\mathbf{y}(t).$$

The functions $u_i(t)$ describe the time-evolution of the amplitudes of the eigenvectors of A, i.e. of the eigenmodes of the physical system.

Total residual: $\mathbf{r} = R^{-1}[\tilde{\mathbf{u}}(t) - \mathbf{u}(t)]$, so overall error:

$$E = ||R^{-1}\tilde{\mathbf{u}}(t) - \mathbf{u}(t)|| \quad \leq \quad ||R^{-1}|| \cdot ||\tilde{\mathbf{u}}(t) - \mathbf{u}(t)||.$$

One single unstable eigenmode ruins the calculation.

Normally all eigenmodes are excited to some extent by numerical noise.
$\Rightarrow$ All eigenvalues of A must lie within the stability contour.

Illustation

DGTD-simulation of empty space with locally refined mesh.

Distribution of typical DGTD operator spectrum within Runge-Kutta stability contour at maximal stable time step:

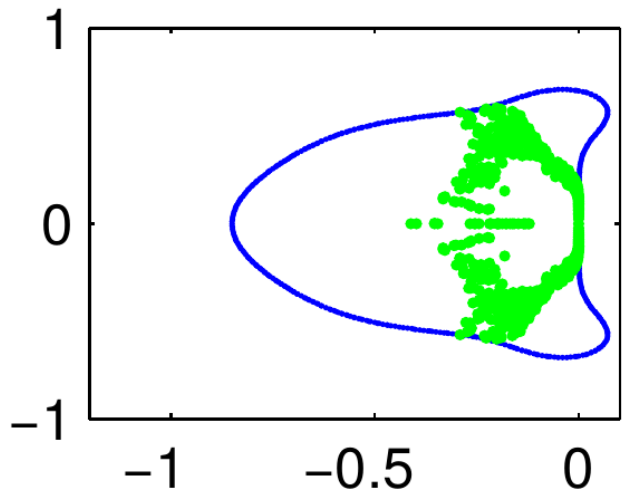


image: M.Sc. thesis Richard Diehl, KIT 2009

Inverse Euler method

Euler's method has the update equation
$\mathbf{y}_{n+1} = y_n + \underbrace{hA\mathbf{y}_n}_{\approx \mathbf{y}' \text{ across time step}}$

An equally good (but harder to evaluate) approximation is $\mathbf{y}' \approx hA\mathbf{y}_{n+1}$.

Using this, we find the update equation
$\mathbf{y}_{n+1} = (1 - hA)^{-1}\mathbf{y}_n$.

This is called an *implicit* method, because every time step requires the solution to a linear problem (expensive for large matrices);
Euler's method is an *explicit* method.

The stability analysis shows that this method is stable for all $|h\alpha - 1|^{-1} \leq 1$, i.e. it is stable everywhere except a circle in the *right* half of the complex plane.
$\Rightarrow$ unconditionally (L-)stable.

Implicit methods (e.g. inverse Euler or implicit Runge-Kutta) are good for small problems that are ill-conditioned or very nonlinear.

Leapfrog integrator

Is there an easier fix for the instability, perhaps just for undamped problems?

$$(y', u') = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} (y, u)$$

We must modify the update method such that we end up on the constant energy circle again.

This can be accomplished by updating y and u not at once, but one after the other:

$$\begin{aligned} u(t_{n+1}) &= u(t_n) + hy(t_n) \\ y(t_{n+1}) &= y(t_n) + hu(\underbrace{t_{n+1}}_{!!!}). \end{aligned}$$

This is known as the leapfrog method and well suited for lossless oscillatory problems.

Byproduct: Time-evolution matrix

Homogeneous, time-independent, linear IVPs of the form

$$\mathbf{y}'(t) = A\mathbf{y}(t), \qquad \mathbf{y}(0) = \mathbf{y}_0$$

can be solved by the time-evolution matrix

$$\mathcal{U}(t) = \exp(At) : \qquad \mathbf{y}(t) = \mathcal{U}(t)\mathbf{y}_0.$$

This is an analogy to quantum mechanics. It is (in principle) exact and a single-step method; therefore it is *unconditionally stable*.

The time-evolution matrix is defined as:

$$\mathcal{U}(t) = R^{-1} \begin{pmatrix} \exp(\lambda_1 t) & 0 & \cdots & 0 \\ 0 & \exp(\lambda_2 t) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \exp(\lambda_n t) \end{pmatrix} R.$$

This expression is not very useful in practice, because it requires a complete matrix diagonalization.

There are approximations to $\mathcal{U}(t)$ that retain the unconditional stability.

Taylor series of matrix functions

Exponential function of a complex variable:

$$\exp(z) = 1 + \sum_{n=1}^{\infty} \frac{z^n}{n!}.$$

If the matrix has no defect, its exponential can also be represented as a Taylor series (proof as homework):

$$\exp(A) = \mathbb{I} + \sum_{n=1}^{\infty} \frac{A^n}{n!}.$$

One possibility to approximate the exponential is to directly truncate the Taylor series (see homework):

$$\exp(At) \approx \mathbb{I} + \sum_{n=1}^{N} \frac{(At)^n}{n!}.$$

Disadvantages of Taylor series

The truncated Taylor expansion requires the powers of the matrix A:

$$\exp(At) \approx \mathbb{I} + \sum_{n=1}^{N} \frac{(At)^n}{n!}.$$

- Requires many matrix-matrix products, which require $\mathcal{O}(n^3)$ operations.
- Often most elements of A are zero, which can be used to save memory and time. The powers of A lose this property (*fill-in*).
- Many especially efficient numerical methods avoid setting up A at all, but only have a recipe how to get $A\mathbf{y}$ from $\mathbf{y}$.
- For time-evolution we really need only $\mathcal{U}(t)\mathbf{y}_0$, not the whole matrix $\mathcal{U}(t)$ itself.
- It is possible to expand the result $\mathbf{y}(t) = \mathcal{U}(t)\mathbf{y}_0$ in the Krylov space spanned by $\{\mathbf{y}_0, A\mathbf{y}_0, A^2\mathbf{y}_0, \ldots\}$.

Homework

Investigate the convergence and stability of the various integration methods (problem sheet).