Hochschule
Kaiserslautern
University of
Applied Sciences

Numerical Methods
Von der Pol-Oscillator

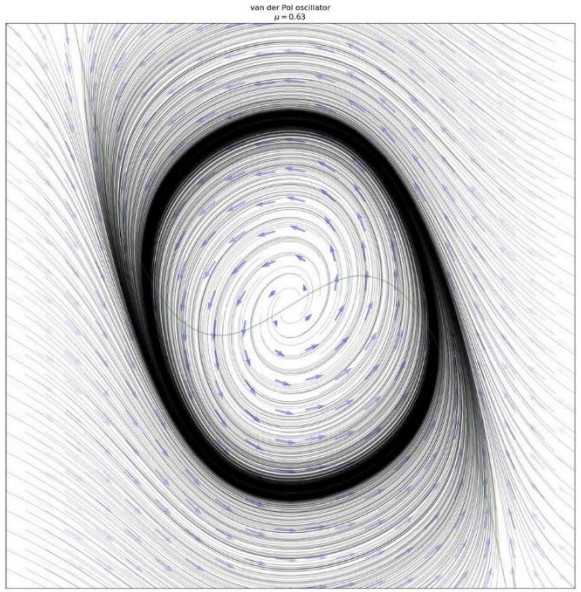
Ganesh Dola (887797)
M.S.c in Mechatronics Engineering

Table of contents:

Hochschule
Kaiserslautern
University of
Applied Sciences

Introduction

- The Van der Pol oscillator is a classical example of a nonlinear second-order differential equation.
- In 1920, ***Balthasar van der Pol*** created it first to simulate oscillations in vacuum tube-based electrical circuits with vacuum tubes, it has since become a fundamental system in nonlinear dynamics.
- The Van der Pol oscillator has several real-world applications across engineering, biology, and physics, mainly because it models nonlinear oscillations with self-sustaining (limit cycle) behavior and nonlinear damping.
- Damping behavior creates the nonlinear scenario in this study.



Governing Equation

Hochschule Kaiserslautern
University of Applied Sciences

- Mathematical Expression

  PDE = $y'' - \mu(1-y^2)y' + y = 0$

Where:

- y = system state (like voltage)
- μ = controls the non linear damping.

  when $\mu = 0$ (weak damping) → it behaves like a simple harmonic motion.

  $\mu > 0$ (relaxation oscillations) → it introduces damping that depends on 'y'.

- The Van der Pol Oscillator models nonlinear damping using a 2nd order ODE.
- Its damping term adds or removes energy based on amplitude, creating a limit cycle.
- Rewriting it as a 1st order system makes it solvable using MATLAB.

Solver Selection

Hochschule Kaiserslautern University of Applied Sciences

- While solving nonlinear systems like van der pol oscillator, the behavior of the solution totally depends on stiffness of system, so solver selection is important.
- The Most important properties of solver to be considered here are stability, accuracy, performance and numerical behavior particularly for the nonlinear system.
- Here to fulfil task requirement different solvers (ODE45, ODE23, ODE15s, Euler) are tested and compared for robustness to solve von der pol oscillator.

ODE45

- It is a MATLAB function used to numerically solve ordinary differential equations (ODEs).
- It's based on the **Runge-Kutta(4,5-order solution) method**, which is a powerful and adaptive technique for solving **non-stiff** ODE's.

Key features of the solver:

- Automatically adjusts the time step for accuracy and efficiency.
- Balances speed and precision well for most problems.
- Ideal for non-stiff problem (If your system doesn't have rapidly changing dynamics).

Basic syntax:

[t, y] = ode45(odefun, tspan, y0)
[t, y] = ode45(odefun, tspan, yo, options)
[t, y, te, ye, ie] = ode45(odefun, tspan, yo, options)
Sol = ode45(odefun, tspan, y0)

Where,

tspan : Time range
y0 : Initial conditions of the system
[t, y] : time , solution values
te,ye,ie : Events
sol : returns a structure
Options : RelTol, AbsTol, MaxStep (for fine tuning)

Ref : *https://ch.mathworks.com/help/matlab/ref/ode45.html*

ODE15s

Hochschule Kaiserslautern University of Applied Sciences

- It is a MATLAB function designed to solve stiff ordinary differential equations using variable-order methods.
- It uses **backward differentiation formulas (BDFs)** or **numerical differentiation formulas (NDFs)** depending on the configuration.
- This solver is particularly effective for problems where rapid changes in solution require adaptive step sizing.
- It is ideal for large systems or models with discontinuities, such as chemical kinetics or biological systems.

Basic syntax:

[t, y] = ode15s(odefun, tspan, y0)
[t, y] = ode15s(odefun, tspan, yo, options)
[t, y, te, ye, ie] = ode15s(odefun, tspan, yo, options)
Sol = ode15s(odefun, tspan, y0)

Where,

tspan : Time range
y0 : Initial conditions of the system
[t, y] : time , solution values
te,ye,ie : Events
sol : returns a structure
Options : RelTol, AbsTol, MaxStep (for fine tuning)

Ref : *https://ch.mathworks.com/help/matlab/ref/ode15s.html*

ODE23

Hochschule Kaiserslautern University of Applied Sciences

- It is a MATLAB function used to solve non-stiff ordinary differential equations using a low-order Runge-Kutta method.
- It compares second- and third-order methods to automatically adjust the step size for accuracy control.
- This solver is ideal for problems that don't require high precision, such as simulations or visualizations in computer graphics.
- Compared to ode45, ode23 is more efficient for coarse error tolerances and may perform better on moderately stiff problems

Basic syntax:

[t, y] = ode23(odefun, tspan, y0)

[t, y] = ode23(odefun, tspan, yo, options)

[t, y, te, ye, ie] = ode23(odefun, tspan, yo, options)

Sol = ode23(odefun, tspan, y0)

Where,

tspan : Time range

y0 : Initial conditions of the system

[t, y] : time , solution values

te,ye,ie : Events

sol : returns a structure

Options : RelTol, AbsTol, MaxStep (for fine tuning)

Ref : https://ch.mathworks.com/help/matlab/ref/ode23.html

Implicit Euler

Hochschule Kaiserslautern
University of Applied Sciences

- Implicit Euler is a simple but powerful solver for stiff differential equations.
- It is also known **as the backward Euler method.**
- It is an implicit method, meaning that a generally nonlinear equation must be solved in each step.

Key features of this solver:

- It is unconditionally stable for linear problems, excellent for stiff systems.
- It can handle large time steps without becoming unstable.
- It is easy to implement conceptually, but may require iterative solvers.

General structure:

$$y(i+1) = y(i) + h * f(t(i+1), y(i+1))$$

where:
y(n) : Approximate solution
h: step size
f: function definition of ODE
y(i+1) : next value of the solution
We have to use '**fsolve**' in MATLAB or Newton's method to estimate the solution at each time step

Ref : https://de.wikipedia.org/wiki/Implizites_Euler-Verfahren

Explicit Euler

Hochschule Kaiserslautern
University of Applied Sciences

- Explicit Euler is a simple but powerful solver for non-stiff differential equations.
- It is also known **as the forward Euler method.**
- This method estimates the next value by adding the derivative at the current point multiplied by the time step

Key features of this solver:

- It is best suited for non-stiff equations due to its conditional stability.
- By stepping forward using the current slope, the method offers a simple yet effective solution technique.

General structure:

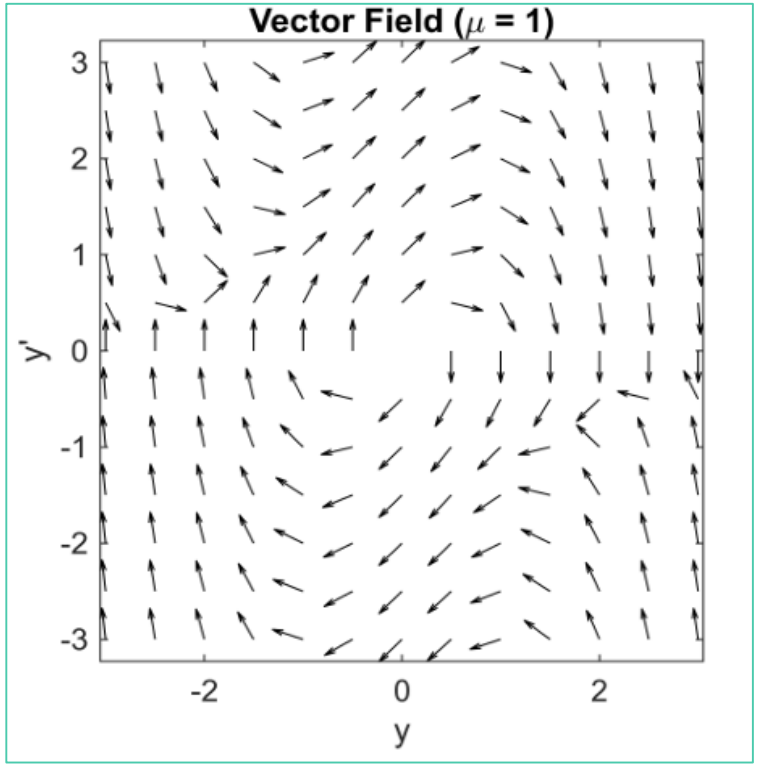
$$y(i+1) = y(i) + h * f(t(i), y(i))$$

Ref : https://de.wikipedia.org/wiki/Explizites_Euler-Verfahren

Simulation & Stiffness

Vector Field for Non-Stiff Case (μ = 1)

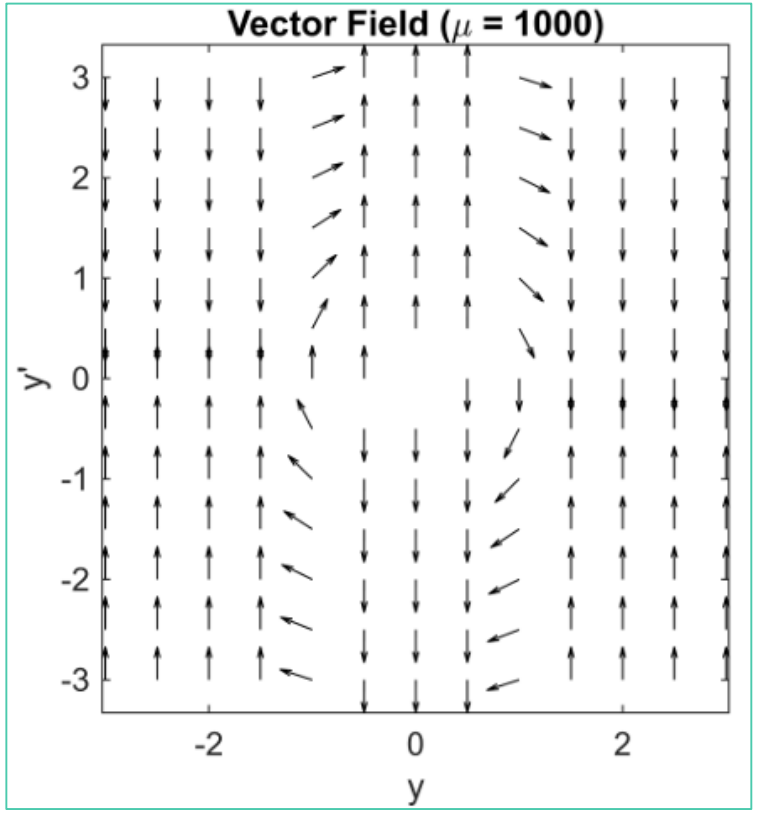
- The arrows show smooth rotational flow around the origin.
- Near the origin: the system spirals outward - this means the origin is unstable.
- As the path moves outward, arrows begin to curve and bend into a circular loop.
- This loop is called a limit cycle - the system settles into this self-sustaining oscillation no matter where you start (unless it's exactly at [0; 0]).
- The motion is mild and symmetric, meaning energy is gradually fed into and then regulated by damping.
- Insight: For $\mu = 1$, the field is relatively smooth, suggesting the system is predictable and easy for solvers to handle.



Simulation & Stiffness

Vector Field for Stiff Case (μ = 1000)

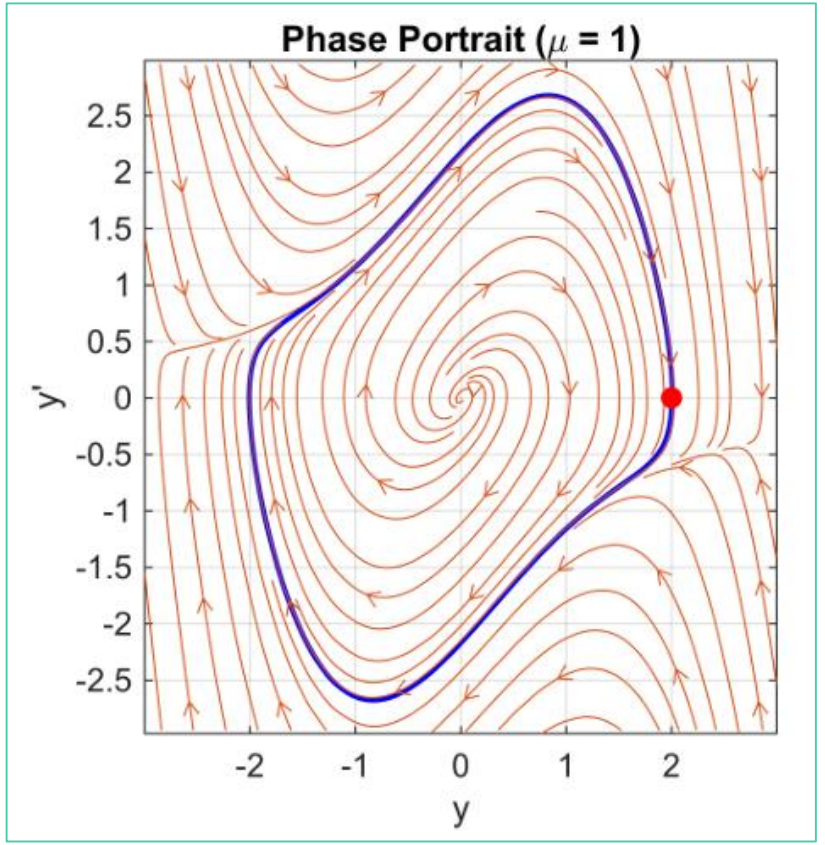
- Arrows change direction drastically and abruptly near the horizontal axis (where y' is approx. 0).
- Near the y-axis, arrows are nearly vertical, indicating slow, “creeping” behavior.
- Farther from the axis, arrows suddenly flip direction and shoot out, showing very fast transitions. This is the hallmark of stiff dynamics.
- It looks like the system is pausing, building tension, and then snapping to the other side.
- This results in a distorted vector field with sharp gradients, illustrating why stiff solvers like ODE15s are necessary.
- They can handle these fast transitions efficiently without becoming unstable.



Simulation & Stiffness

Phase Portrait for Non-Stiff Case (μ = 1)

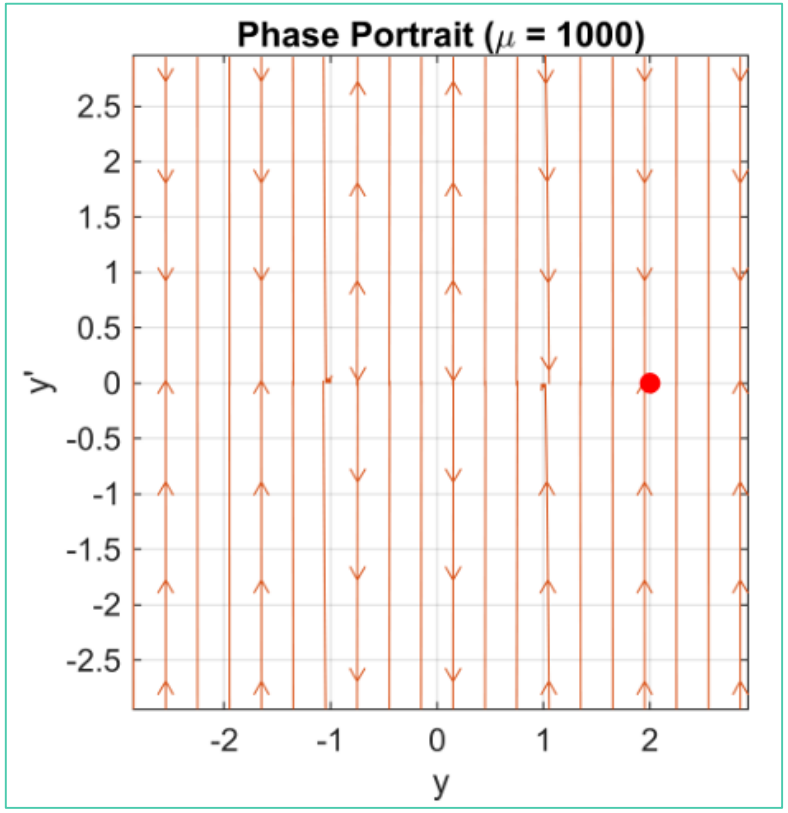
- The phase portrait displays a trajectory starting at [2, 0] (large displacement, no velocity
- The phase portrait shows the system spiraling inward toward a stable limit cycle-a hallmark of self-sustained, nonlinear oscillation.
- This signifies self-sustained oscillations with moderate damping.
- Eventually, the system locks into periodic behavior, even without external forcing
- The damping is moderate, so the system's energy decays smoothly until it locks into a periodic motion.
- The damping parameter μ = 1 ensures non-stiff dynamics, captured well by solvers like ODE45.



Simulation & Stiffness

Phase Portrait for Stiff Case (μ = 1000)

- For μ = 1000, the Van der Pol oscillator exhibits a stiff limit cycle with a rectangular phase portrait.
- The system slowly evolves on the horizontal segments and rapidly transitions on the vertical segments, producing sharp corners in the trajectory.
- This visual signature of stiffness makes it clear why specialized solvers like ODE15s are essential for accurate and efficient simulation.

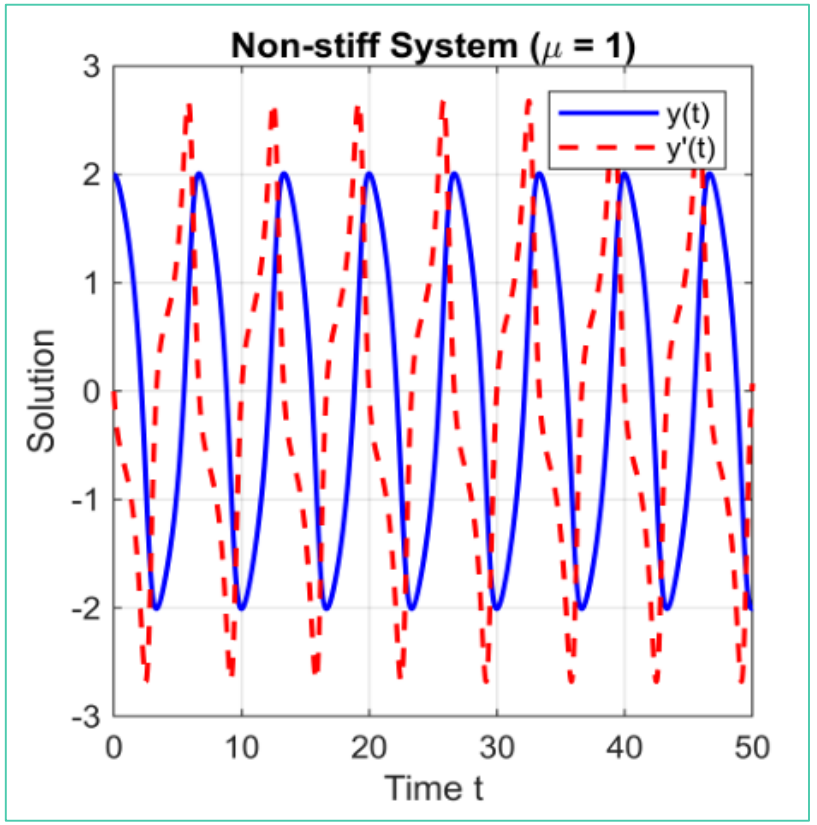


Simulation & Stiffness

Hochschule Kaiserslautern
University of Applied Sciences

Solution vs Time for Non-Stiff Case (μ = 1)

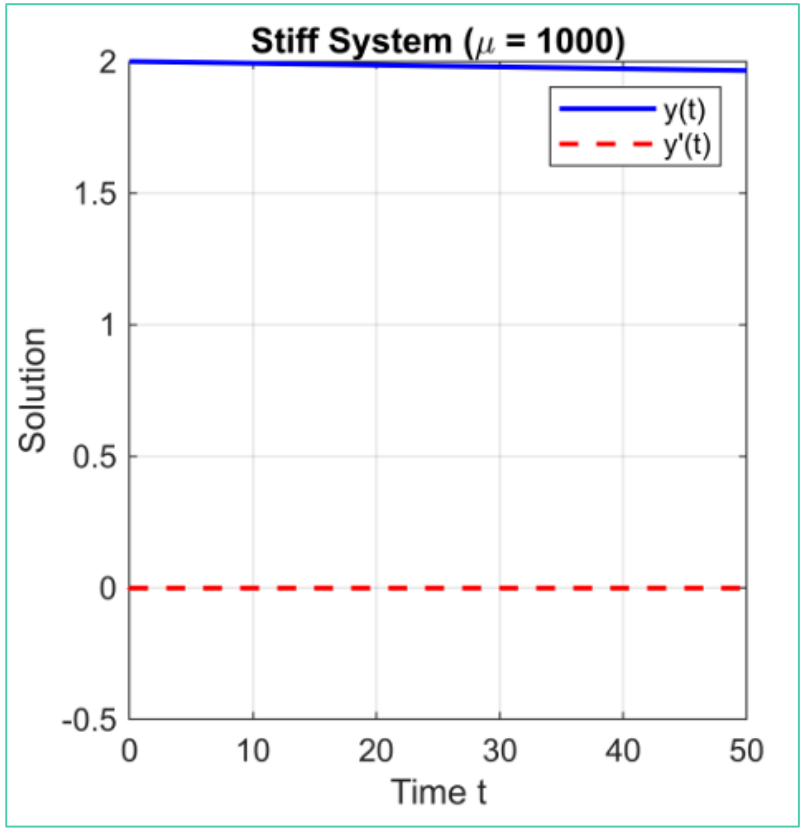
- Plots of y(t) represents the position of the oscillator over time and y'(t) represents the velocity over time.
- This oscillator behaves like a pendulum swinging smoothly back and forth.
- The amplitude (maximum height of the wave) stays mostly constant
- The two curves (position and velocity) look like smooth sine waves, indicating stable oscillations.



Simulation & Stiffness

Solution vs Time for Stiff Case (μ = 1000)

- Both lines are almost completely flat, There are no visible oscillations or spikes. Oscillator is recharging slowly, and not doing any dramatic movements.
- a fast transient can be observed at a point in time the system has already completed its rapid changes and settled into a slow recovery phase.
- Stiff equations have some components that change very fast and others very slowly. this makes them tricky to simulate.
- non-stiff solver like ode45, it will miss the spikes or take forever to capture them and stiff solver like ode15s, which handles these sudden changes and smooth stretches efficiently.



Solver Performance

Non-Stiff system($\mu = 1$)

Hochschule Kaiserslautern
University of Applied Sciences

Solver	Time (s)	Steps	Description
ode45 (explicit)	0.0156	2005	**Strikes a great balance of speed, step size, and accuracy and Requires fewer steps than ODE23 or Euler, with faster runtime**
ode23 (explicit)	0.3721	4503	**More steps and longer runtime due to being more conservative in error control and Slightly better for coarse tolerance, but inefficient here.**
ode15s (implicit)	0.0461	**1573**	**Still accurate, but less efficient than ODE45 in non-stiff dynamics and Takes unnecessary steps trying to handle stiffness that does not exist.**
Explicit Euler	0.0191	3000	**Fast, but requires small fixed step sizes for acceptable accuracy and Accuracy suffers, especially for nonlinear or oscillatory systems.**

Solver Performance

Stiff system(μ = 1000)

Hochschule Kaiserslautern
University of Applied Sciences

Solver	Time (s)	Steps	Description
ode45 (explicit)	0.5021	176841	**Explicit solver not suited for stiff systems Takes many small steps, long runtime and potential instability.**
ode23 (explicit)	0.0104	149	**Faster than ODE45, but still not stiff-specialized Low step count is misleading possible loss of accuracy.**
ode15s (implicit)	0.0028	**106**	**Optimized for stiff systems, Uses adaptive time stepping, handling rapid changes efficiently, Lowest execution time and fewest steps with reliable results.**
Implicit Euler	0.6399	300	**Handles stiffness well but is computationally expensive, Requires solving equations at each step to slower in practice.**

Error Analysis

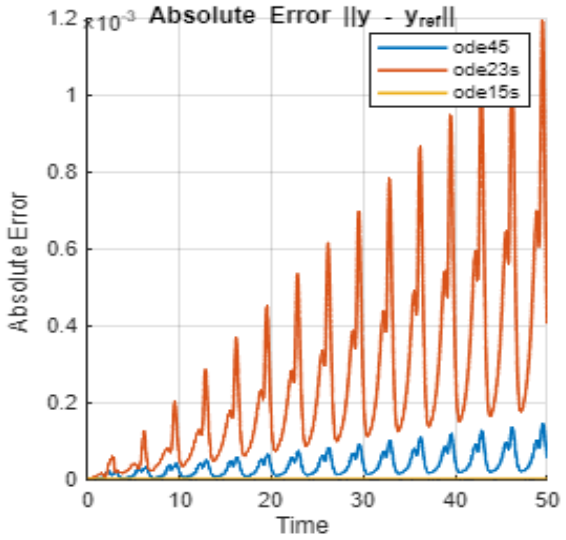
Hochschule Kaiserslautern University of Applied Sciences

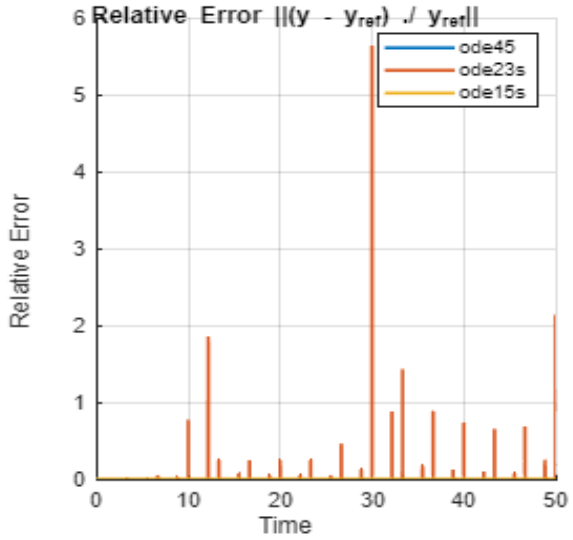
Error analysis is essential for validating solver performance, ensuring that numerical solutions truly represent the system behavior, and guiding the optimal selection of solvers for different stiffness conditions.

There are two types of error:

- Absolute Error: The **absolute difference** between the numerical result and the reference value.
- Relative Error: The **absolute error** divided by the magnitude of the reference value. It measures error proportionally.

Absolute Error & Relative Error





Solver	Max abs error	Max Rel error
ode45	1.4386e-04	9.2402e-03
ode23s	1.1942e-03	5.6410e+0
ode15s	4.4409e-16	3.7513e-12

- ode15s has nearly zero error it's the reference and hence perfectly accurate.
- ode45 has small bumps generally low error, but not perfect.
- ode23s (orange) has the highest error here. It's not ideal for non-stiff systems.

- Measures the error relative to the magnitude of the solution.
- ode15s stays at nearly zero and ode45 shows moderate fluctuations.
- ode23s shows extremely high spikes.
- ODE45 and ODE15s are performing well as both solvers have high numerical accuracy, efficient error control.

Conclusion

Hochschule Kaiserslautern
University of Applied Sciences

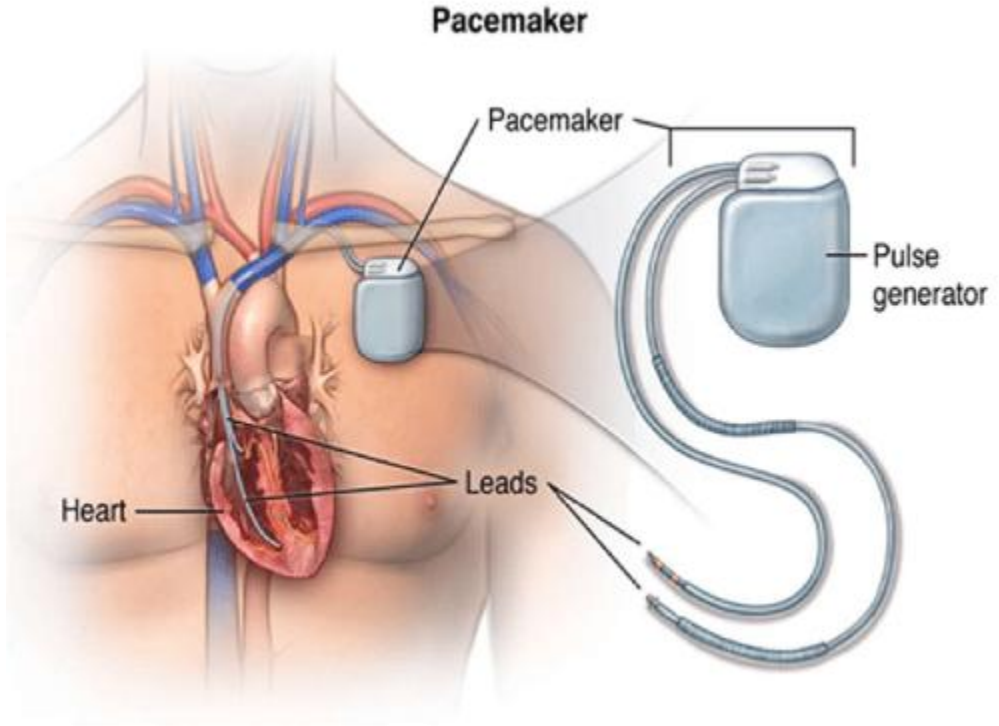
- All tested solvers performed adequately for solving the Van der Pol oscillator in the non-stiff regime. Among them, ODE45 emerged as the preferred choice for its excellent balance between accuracy, efficiency, and ease of use when stiffness is low.
- However, as the system transitions to a stiff regime (e.g., large ‘μ’), ODE15s drastically outperforms all other methods. Its ability to handle stiffness with adaptive step sizes and variable order makes it the most robust and efficient solver for such nonlinear dynamics.
- This comparison clearly highlights the importance of selecting a solver based on the problem & stiffness, confirming that ODE15s is the most reliable option across both stiff and non-stiff scenarios for the Van der Pol oscillator.

Real World Applications

Hochschule Kaiserslautern
University of Applied Sciences

Heart Pacemaker

- **Heart Pacemaker** is developed by van der pol system.
- The heart does not beat like perfect sine wave, its behavior is completely nonlinear, self-sustaining electrical pulses. The van der Pol oscillator mimics this by generating oscillations on its own.
- It generates the required pulse in veins for person who have heart dieses.
- There are also other real-world equipment's of bioengineering like Impendence sensors, some displacement and position sensors works on principal of van der pol system.



Ref: https://arxiv.org/pdf/2309.07010#:~:text=The%20Van%20der%20Pol%20oscillator%20has%20found%20applications%20in%20diverse,the%20damping%20coefficient%20is%20large.

Wind turbines

- **Wind turbines** operates under variable wind conditions so the conditions are pretty nonlinear due to wind gusts, rotational imbalance and aero elastic effects.
- These oscillations can cause fatigue and structural damage if not properly damped.
- So that VDP system helps here to analyze and designing of the components.

Ref: *https://www.mdpi.com/2073-8994/14/1/88#:~:text=Hafien%20et%20al.,on%20the%20unsteady%20lift%20coefficient.*

Formula1 race cars

- **Formula1 race cars** experience airflow-induced oscillations (like buffeting, flutter, or vortex shedding).
- It affects on Downforce stability, Driver control and Structural integrity of body components.
- In aerodynamic design of racecars, the Van der Pol oscillator models self-sustained oscillations caused by airflow interactions, helping engineers predict and reduce unwanted vibrations that compromise performance and safety.

Ref: *https://www.scielo.org.mx/scielo.php?pid=S0035-001X2022000100009&script=sci_arttext.*

Hochschule
Kaiserslautern
University of
Applied Sciences

Thank you