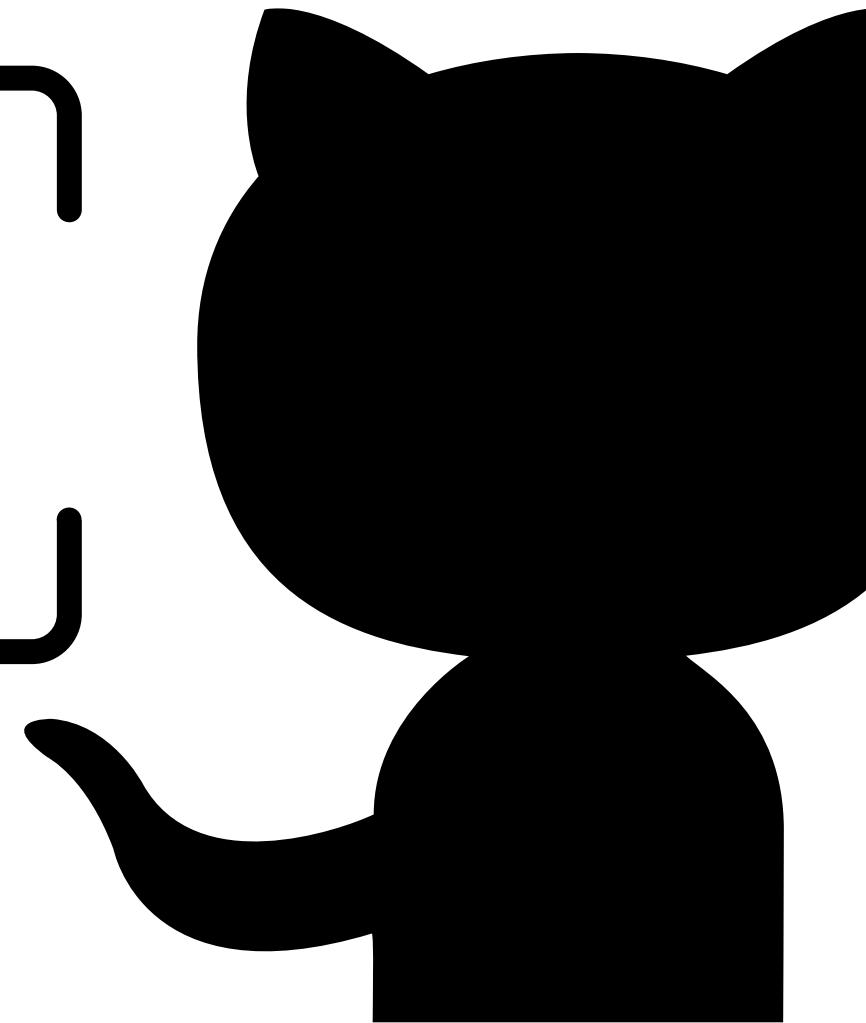


A Warm Welcome to our Presentation

Slides can be found at

<https://is.gd/stiff>

A question earns a
contributor badge
in our GitHub
Repo 



Numerical Approaches for Solving (Stiff) Differential Equations

Introduction to basic and commonly used methods and their limitations.

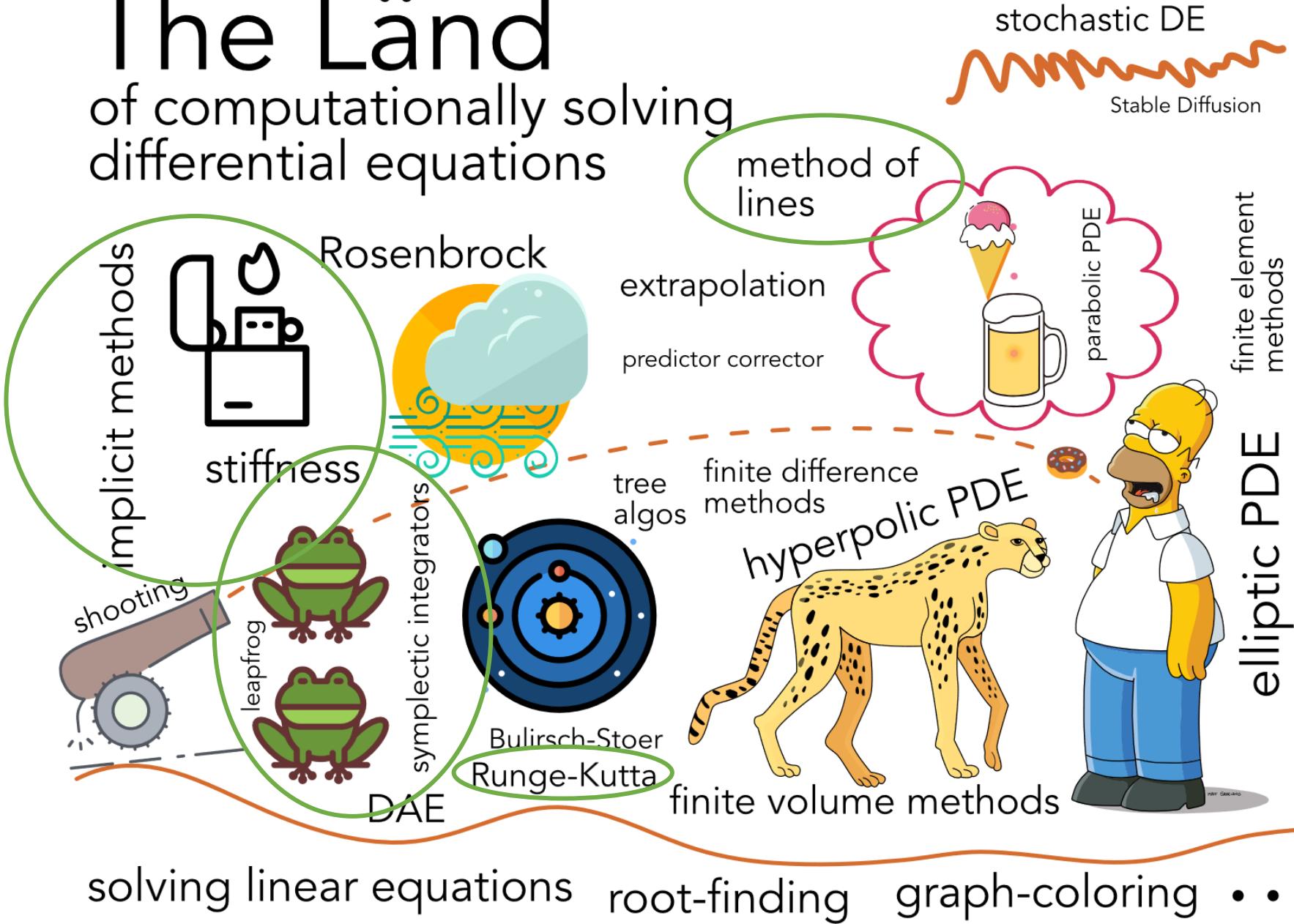
Seminar talk in the Scientific Machine Learning Seminar by Dr. Tobias Buck, AstroAI-Lab
Heidelberg University, Department for Physics and Astronomy

by Daniel Richter and Leonard Storcks, 23.05.2023

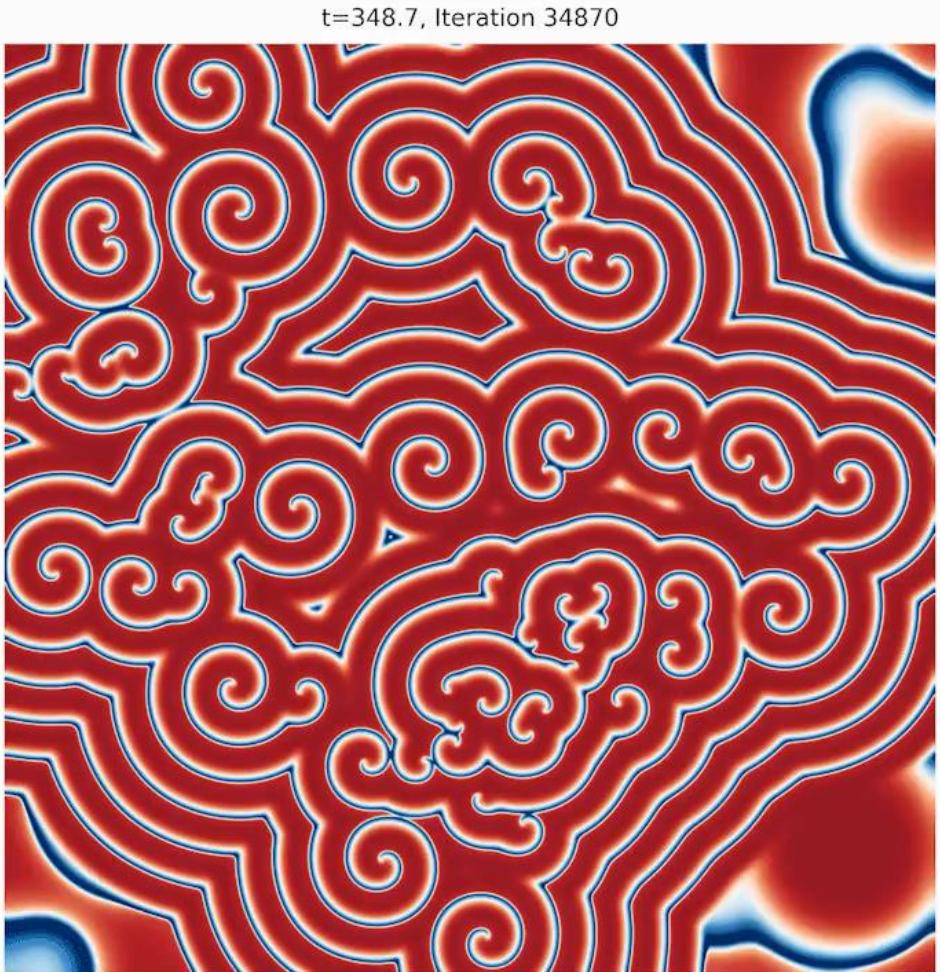
bō
hi
si
+

The Länd

of computationally solving
differential equations



Motivation



Source: [H-Psi-is-E-Psi](#), Brusselator Oscillations HQ Render, CC BY-SA 4.0

A beautiful oscillatory system involving chemistry and diffusion
the Brusselator.

... and its stiff

Context of SciML

»Scientific machine learning is the burgeoning field combining techniques of machine learning into traditional scientific computing and mechanistic modeling.« [e. g. Zubov et al, 2021]

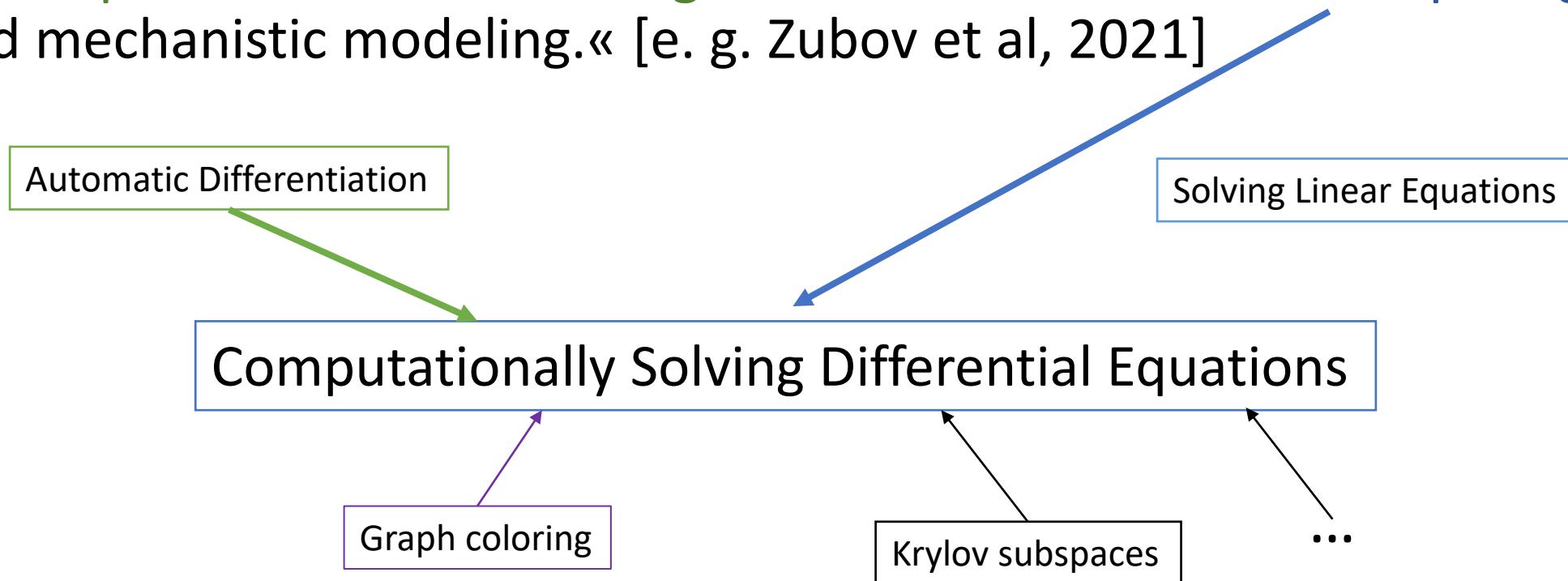


Table of Contents

- Some Classical Methods for Solving ODEs
- The Problem of Conserved Quantities
- The Problem of Stiffness
- Efficiently Solving Stiff ODEs
- Breakdown of Classical Methods & Outlook
- Key insights to take away

main part

Some Classical Methods for Solving ODEs

Explicit Euler Method

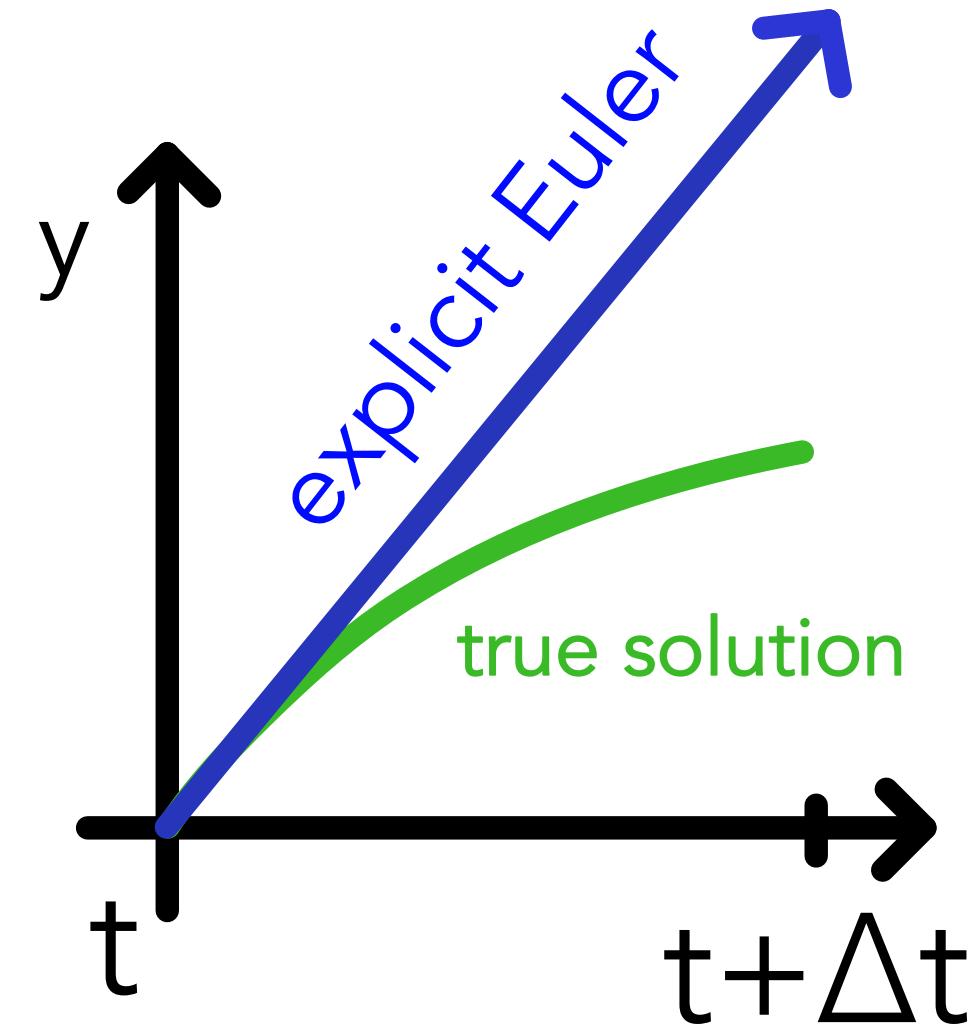
Euler method for solving ODEs $\partial_t \underline{y} = f(\underline{y})$

$$\underline{y}^{(n+1)} = \underline{y}^{(n)} + f(\underline{y}^{(n)}) \Delta t$$

Live coding stability analysis example

$$\frac{dy}{dt} = -\alpha y, \quad \alpha > 0, \quad y(0) = y_0$$

known solution $y(t) = y_0 e^{-\alpha t}$



What is the Order of Explicit Euler?

Taylor Expansion

$$\underline{y}_{n+1} = \underline{y}_n + \underline{y}'_n \Delta t + \mathcal{O}_s(\Delta t^2)$$

To simulate over a time T , we need $N_s = \frac{T}{\Delta t}$ steps, so
the error over time scales with

$$N_s \mathcal{O}_s(\Delta t^2) = \mathcal{O}_T(\Delta t)$$

- first order accuracy.

Runge-Kutta-Method

Q: How would you construct a higher order scheme?

Runge-Kutta Idea: Weighted combination of simple derivatives

$$\underline{y}_{n+1} = \underline{y}_n + h \sum_{i=1}^m \beta_i \underline{k}_i$$

$$\underline{k}_i = f\left(\left(\underline{y}_n + h \sum_{l=1}^{m-1} \alpha_{i,l} \underline{k}_l\right), t_n + \gamma_i h\right), \quad i = 1, \dots, m$$

$$\sum_{l=1}^m \alpha_{i,l} = \gamma_i, \quad \alpha_{i,l} = 0 \text{ for } l \geq i \rightarrow \text{explicit}$$

RK2

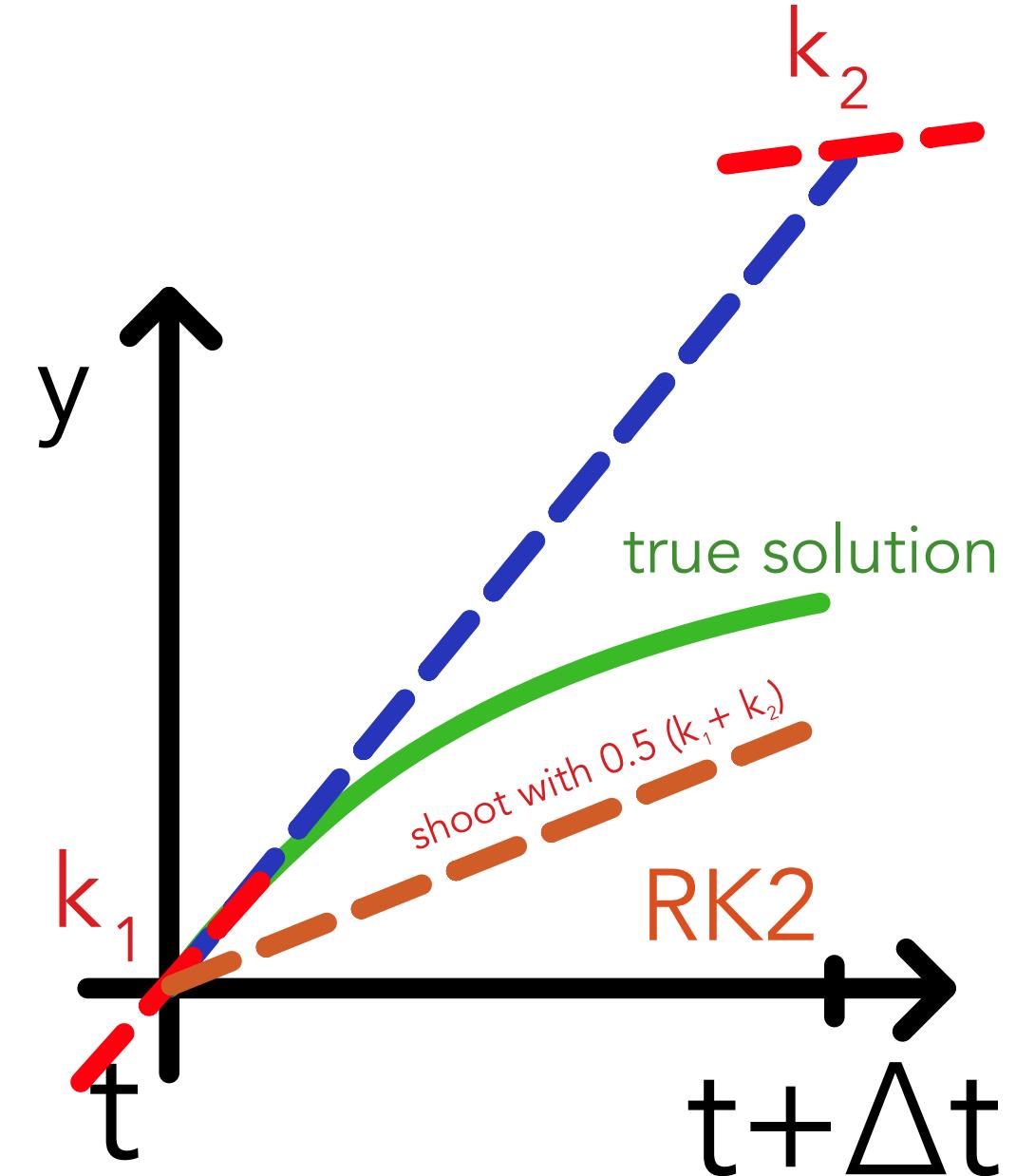
$$k_1 = f(y_n, t_n)$$

$$k_2 = f(y_n + \Delta t k_1, t_n + \Delta t)$$

$$y_{n+1} = y_n + \frac{\Delta t}{2} (k_1 + k_2) + \mathcal{O}(\Delta t^3)$$

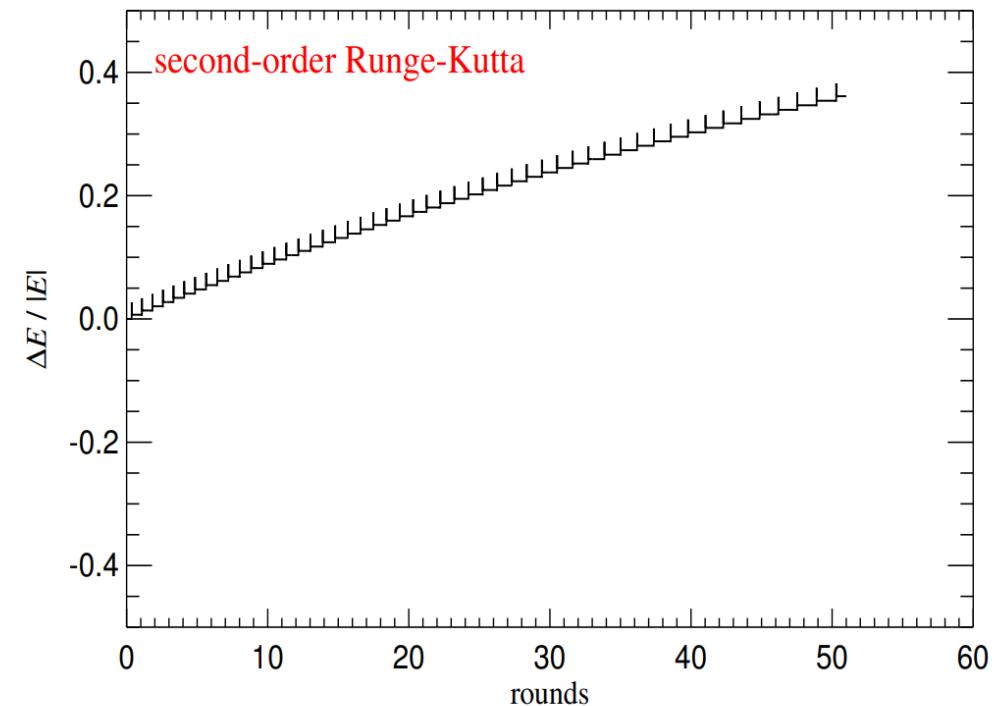
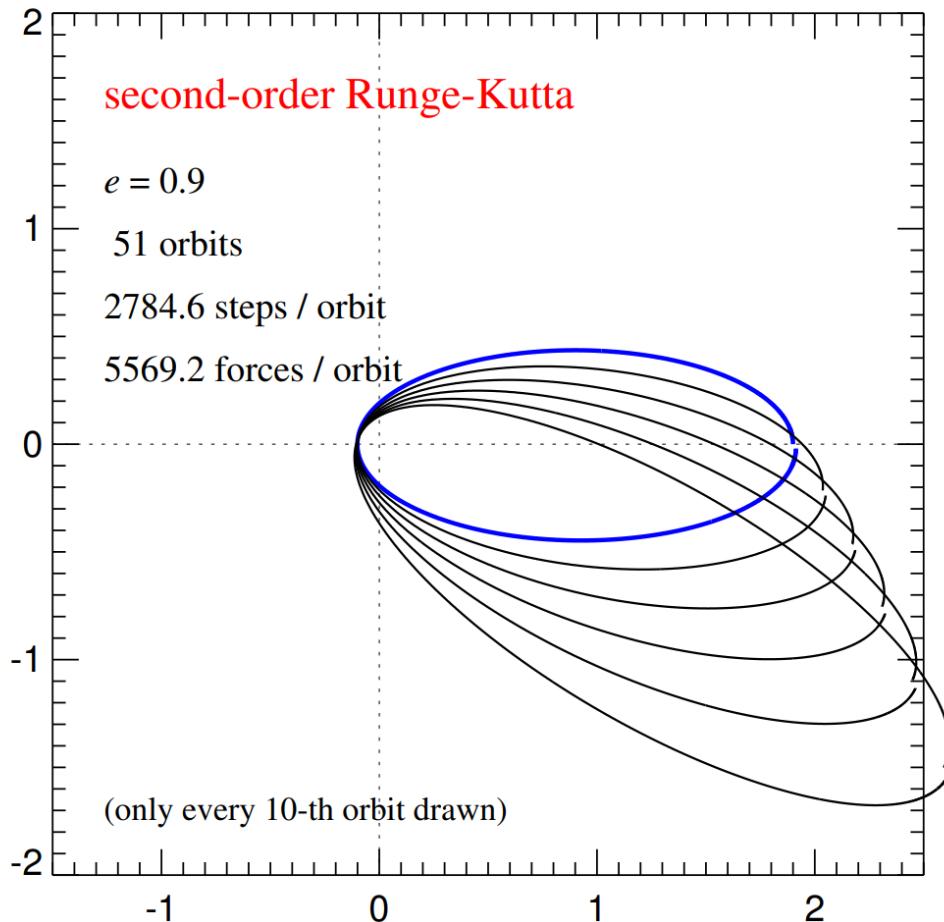
Outlook on further RK methods

- e. g. RK Fehlberg 4/5 with included error estimation



The Problem of Conserved Quantities

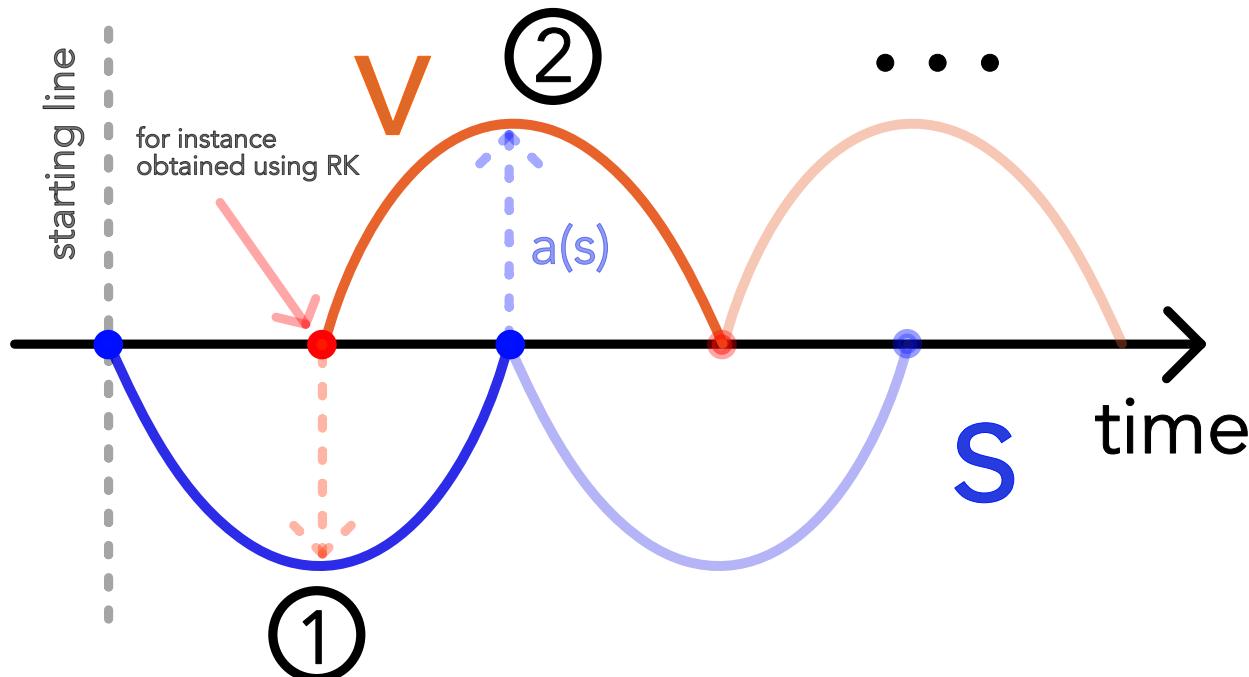
Solution of the two-body Problem



Energy error in each step!

[from the Fundamentals of Simulation Methods lecture notes by Springel et al., version 2022/2023]

Leapfrog



Symplectic integrator with good energy conservation properties; time reversibility; exact angular momentum conservation

1. update location:

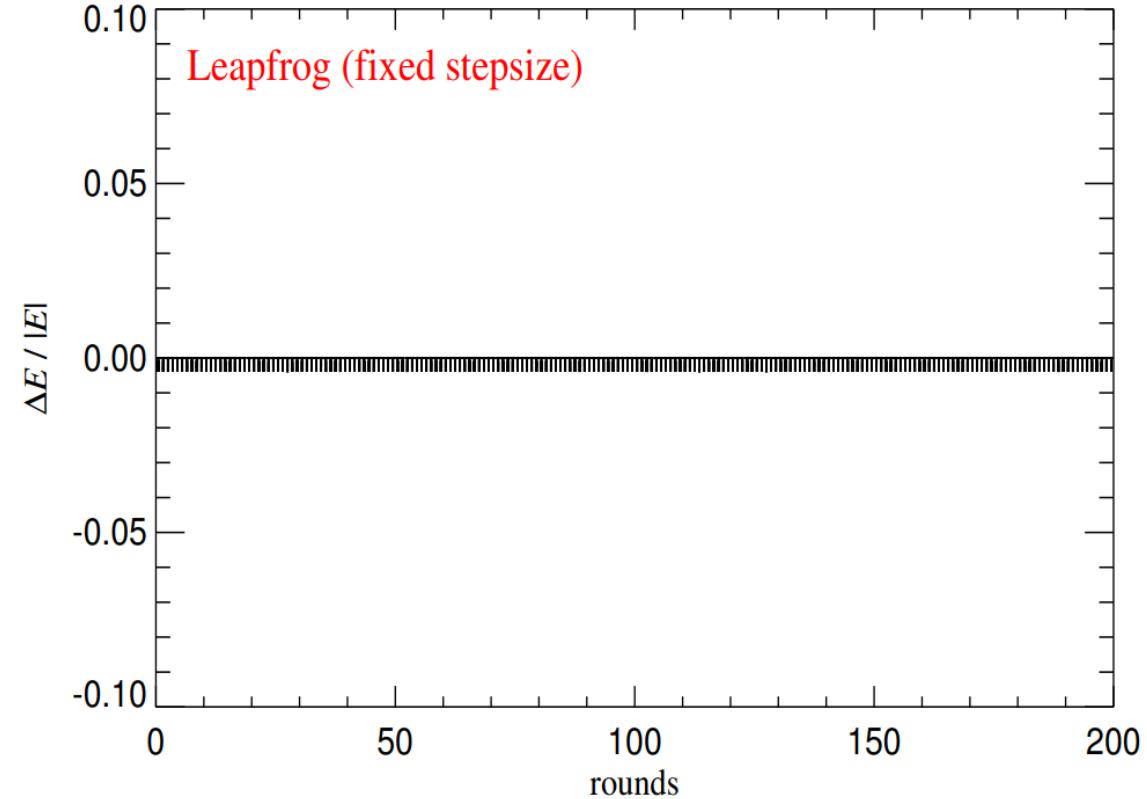
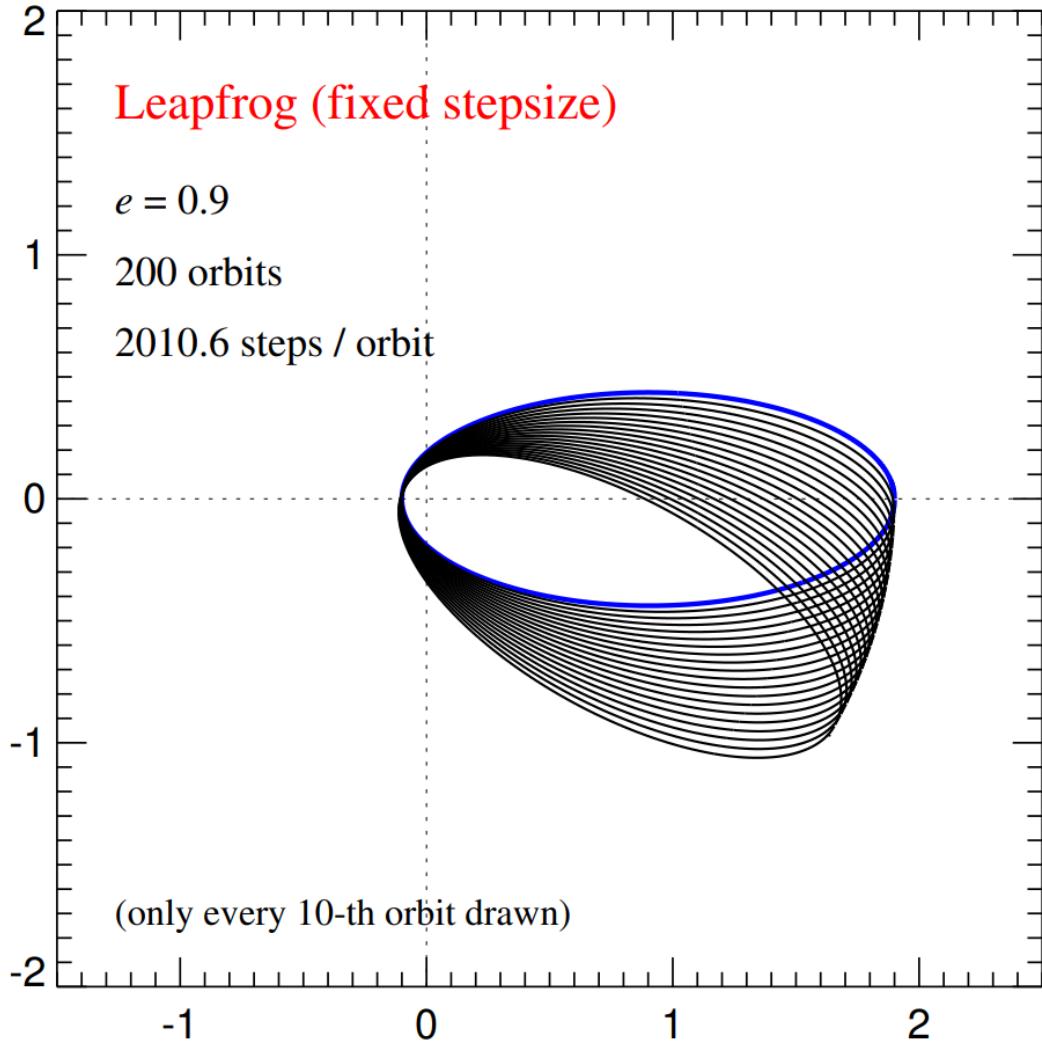
$$\begin{aligned}s\left(t + \frac{1}{2}\Delta t\right) &= \\ s\left(t - \frac{1}{2}\Delta t\right) + \underline{v}(t)\Delta t + \\ \mathcal{O}(\Delta t^3)\end{aligned}$$

2. update velocity:

$$\begin{aligned}\underline{v}(t + \Delta t) &= \underline{v}(t) + \\ \underline{a}\left(t + \frac{1}{2}\Delta t\right)\Delta t + \\ \mathcal{O}(\Delta t^3)\end{aligned}$$

Exact solution to $H_{\text{leap}} = H + H_{\text{err}}$,

$$H_{\text{err}} \propto \frac{\Delta t^2}{12} \left\{ \{H_{\text{kin}}, H_{\text{pot}}\}, H_{\text{kin}} + \frac{1}{2} H_{\text{pot}} \right\} + \mathcal{O}(\Delta t^3)$$



[from the “Fundamentals of Simulation Methods” lecture notes by Springel et al., version 2022/2023]

Intuition of Symplecticity in 2D

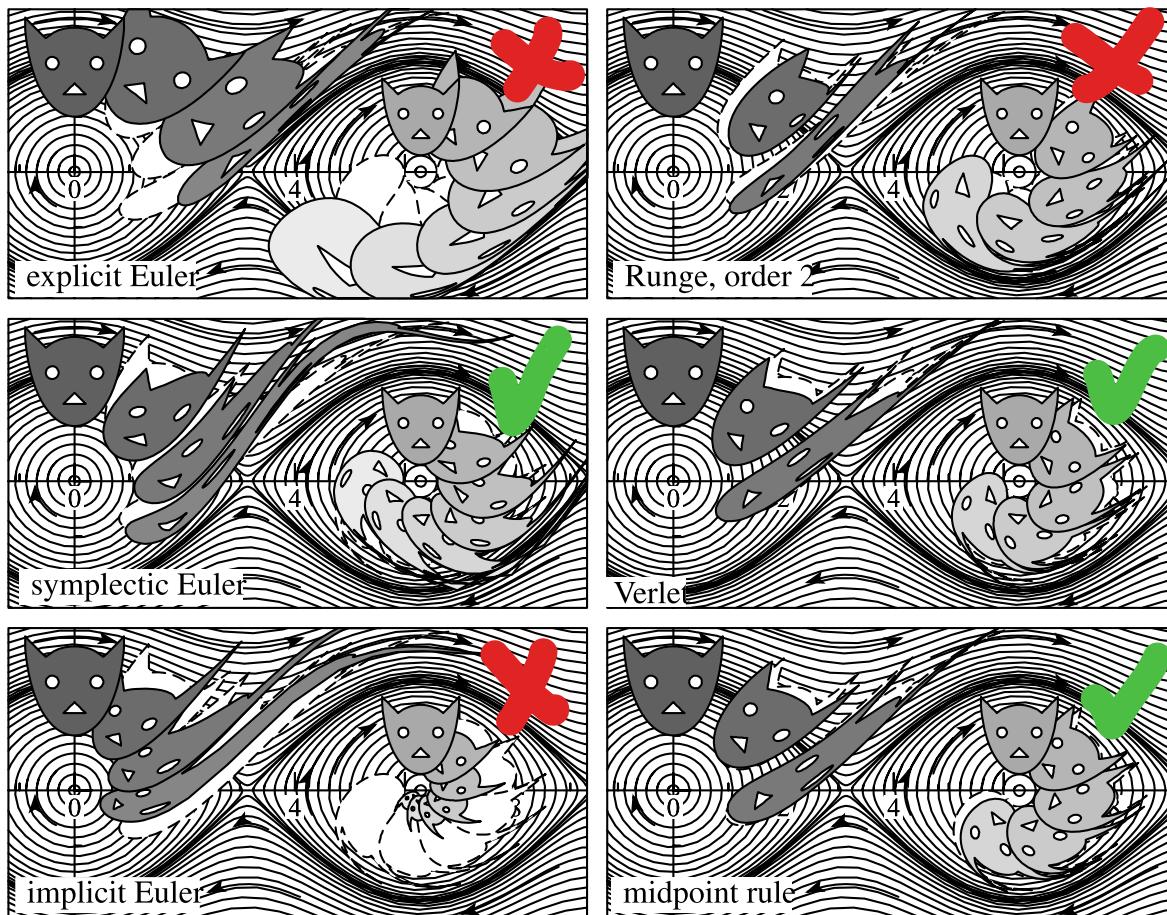
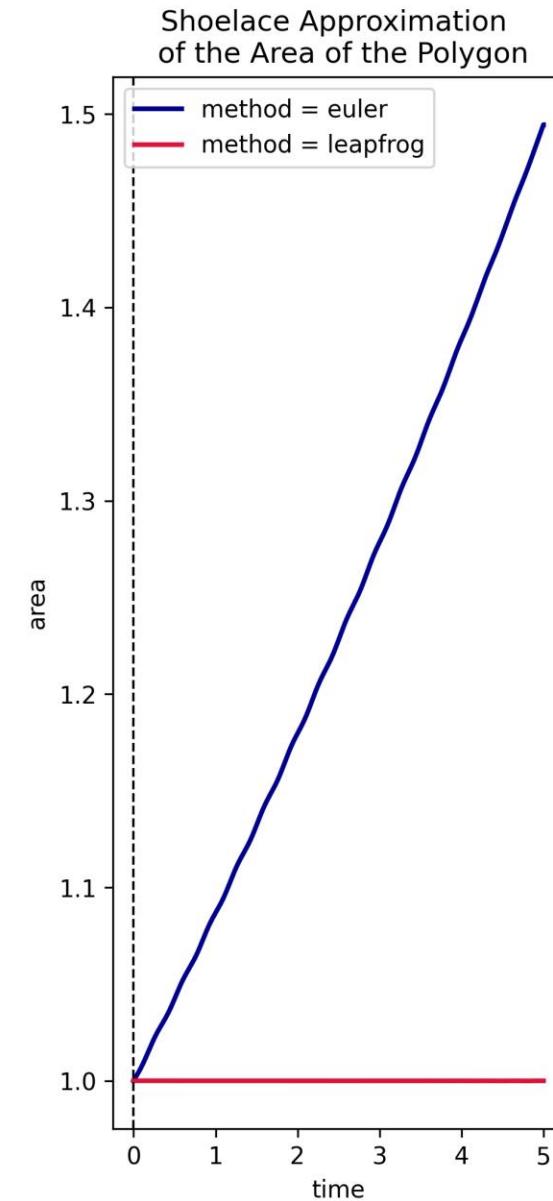
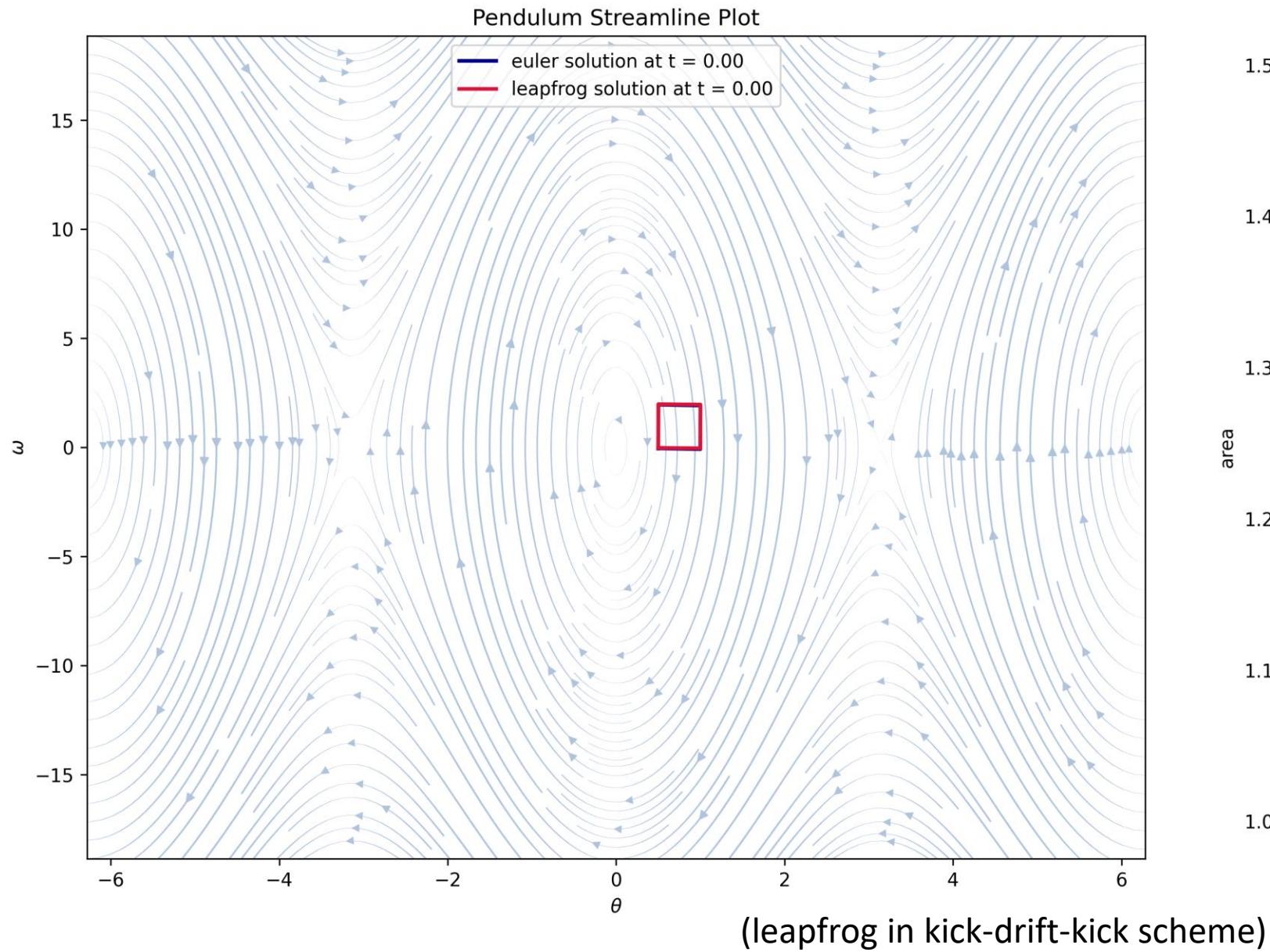


Fig. 3.1. Area preservation of numerical methods for the pendulum; same initial sets as in Fig. 2.2; first order methods (left column): $h = \pi/4$; second order methods (right column): $h = \pi/3$; dashed: exact flow

A symplectic transformation is area preserving.

- Note: No exact energy conservation, e. g. leapfrog exactly solves perturbed Hamiltonian
- Phase space conservation \Leftrightarrow long term energy error bounded, but floating-point error \rightarrow small energy drift
- recommended for long-term integration, e. g. molecular dynamics, planetary orbitals, ...

[from chapter 6 in “Geometrical Numerical Integration”, Hairer, Lubich, Wanner, 2005]



The Problem of Stiffness

Linear ODE example

$$\partial_t u = 998u + 1998v, \quad \partial_t v = -999u - 1999v, \quad u(0) = 1, \quad v(0) = 0$$

$$\partial_t \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} 998 & 1998 \\ -999 & -1999 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}, \quad \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Analytic solution

$$\begin{pmatrix} u \\ v \end{pmatrix} = \exp\left(\begin{pmatrix} 998 & 1998 \\ -999 & -1999 \end{pmatrix} t\right) \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$
$$= \begin{pmatrix} 2 \\ -1 \end{pmatrix} \exp(-1t) + \begin{pmatrix} -1 \\ 1 \end{pmatrix} \exp(-1000t)$$

ill-conditioned matrix

$$\text{stiffness ratio} = \frac{\max |\operatorname{Re} \lambda_i|}{\min |\operatorname{Re} \lambda_i|} = \frac{1000}{1}$$

decay on vastly different time-scales

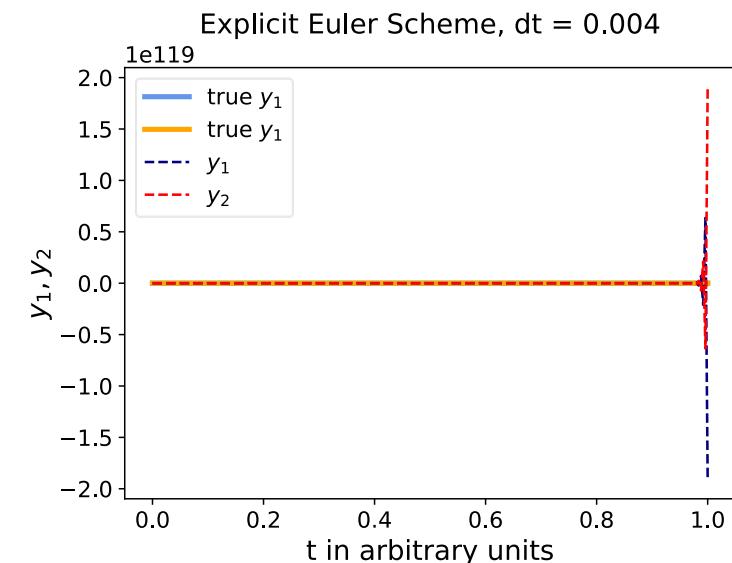
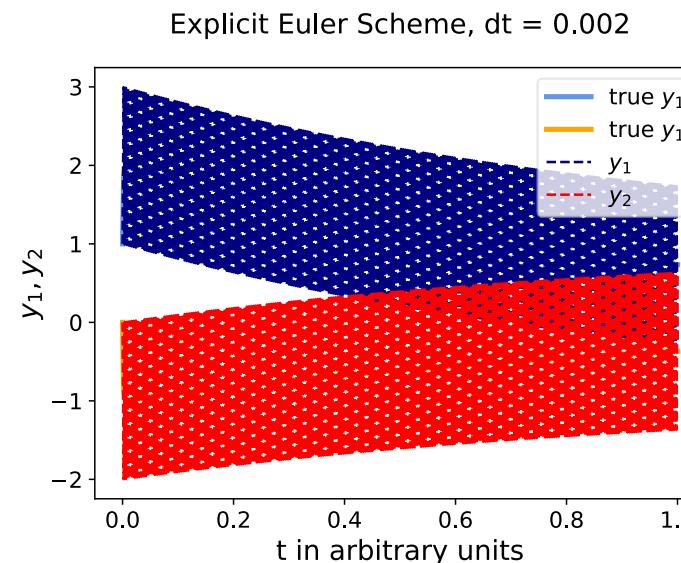
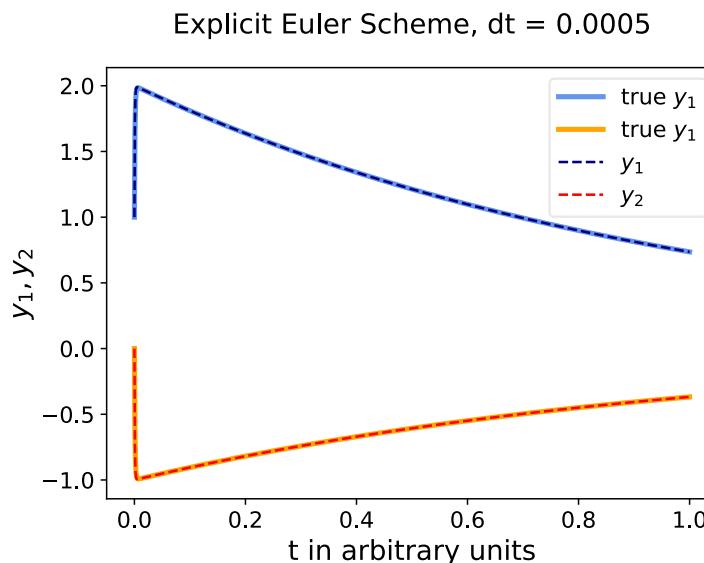
Example from the
“Numerical Recipes” book

Explicit Schemes in Failure Mode

$$\Delta t < \frac{1}{1000} \rightarrow \text{stable}$$

$$\frac{1}{1000} < \Delta t \leq \frac{2}{1000} \\ \rightarrow \text{oscillations}$$

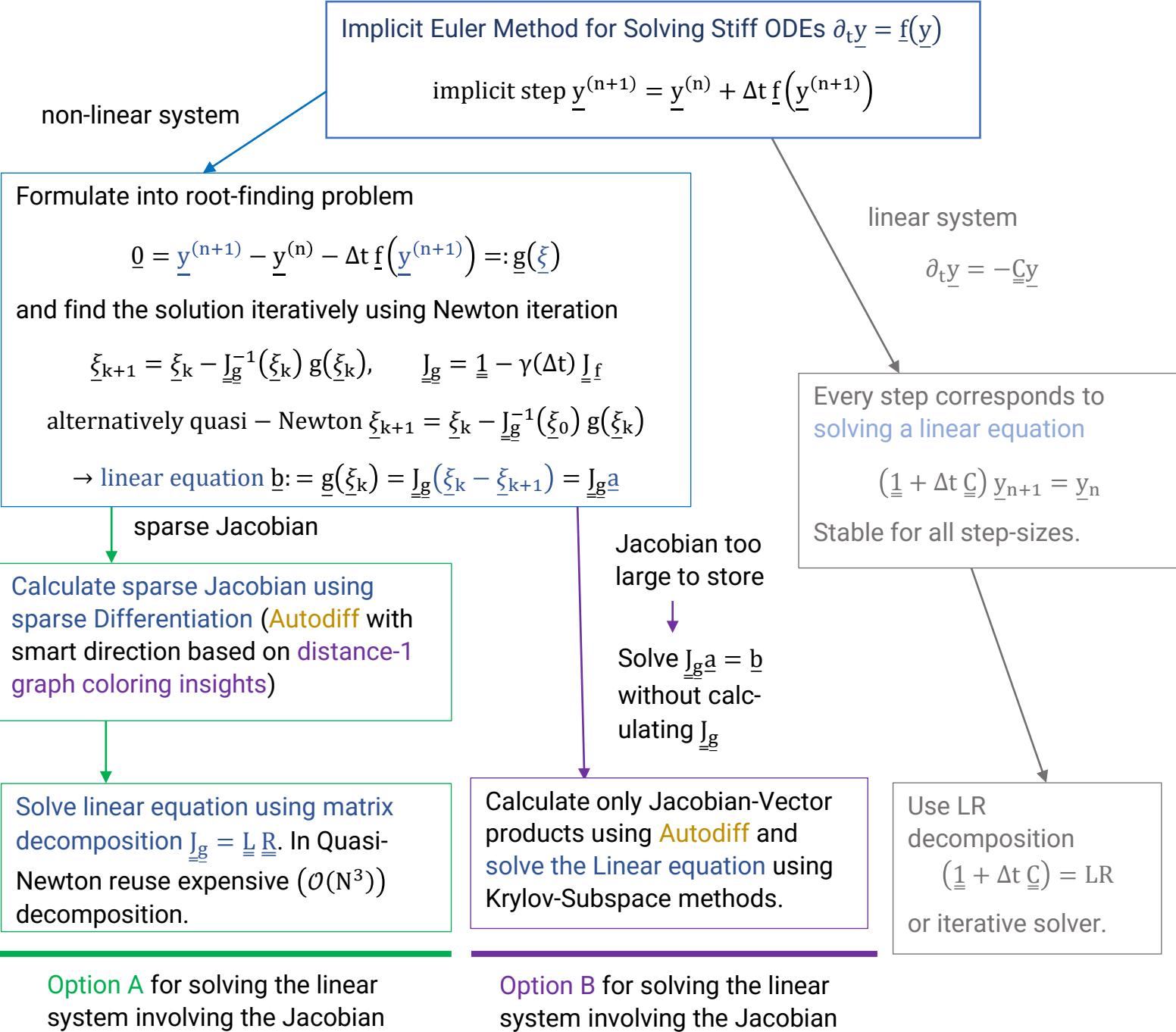
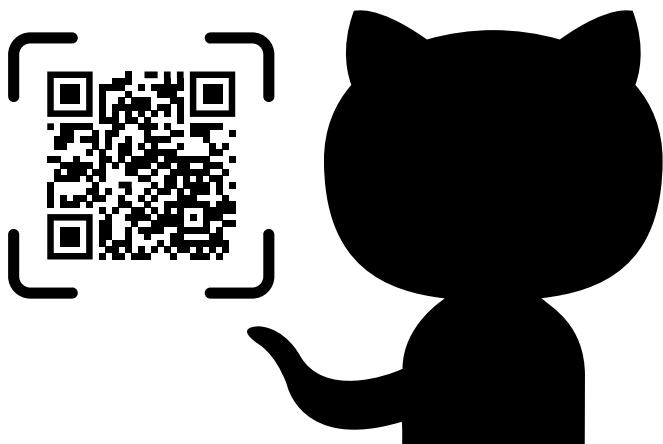
$$\Delta t > \frac{2}{1000} \\ \rightarrow \text{unstable}$$



Timestep excessively small to large scale solution smoothness necessary in explicit method → **Stiffness** → more efficient approach?

Efficiently Solving Stiff ODEs

Overview on Solving Stiff ODEs



Introduction of the Brusselator | Note on PDEs

Non-linear chemical-reaction-diffusion partial differential equation in 1D

$$\frac{\partial u}{\partial t} = A + u^2 v - (B + 1)u + \alpha \frac{\partial^2 u}{\partial x^2}$$

$$\frac{\partial v}{\partial t} = Bu - u^2 v + \alpha \frac{\partial^2 v}{\partial x^2}$$

fixed concentrations of substances A, B
x and t dependent concentrations u, v

diffusion constant α

Turn into ODE by method of lines

discretize all partial derivatives but 1

$$\partial_t u_i = 1 + u_i^2 v_i - 4u_i + \frac{\alpha}{(\Delta x)^2} (u_{i-1} - 2u_i + u_{i+1})$$

$$\partial_t v_i = 3u_i - u_i^2 v_i + \frac{\alpha}{(\Delta x)^2} (v_{i-1} - 2v_i + v_{i+1})$$

$$\text{grid index } i = 0, \dots, N + 1, \Delta x = \frac{1}{N + 1}$$

write compactly as

$$\underline{y} = f(\underline{y}), \quad \underline{y} = \begin{pmatrix} u_0 \\ \vdots \\ u_{N+1} \\ v_0 \\ \vdots \\ v_{N+1} \end{pmatrix}$$

$f: \mathbb{R}^{2(N+2)} \rightarrow \mathbb{R}^{2(N+2)}$ from above

Boundary values and textbook solution

Boundary values

$$u_0(t) = u_{N+1}(t) = 1$$

$$v_0(t) = v_{N+1}(t) = 3$$

Initial conditions

$$u_i(0) = 1 + \sin(2\pi x_i)$$

$$v_i(0) = 3, i = 1, \dots, N$$

Textbook solution from “Solving Stiff ODEs II”

by E. Hairer and G. Wanner

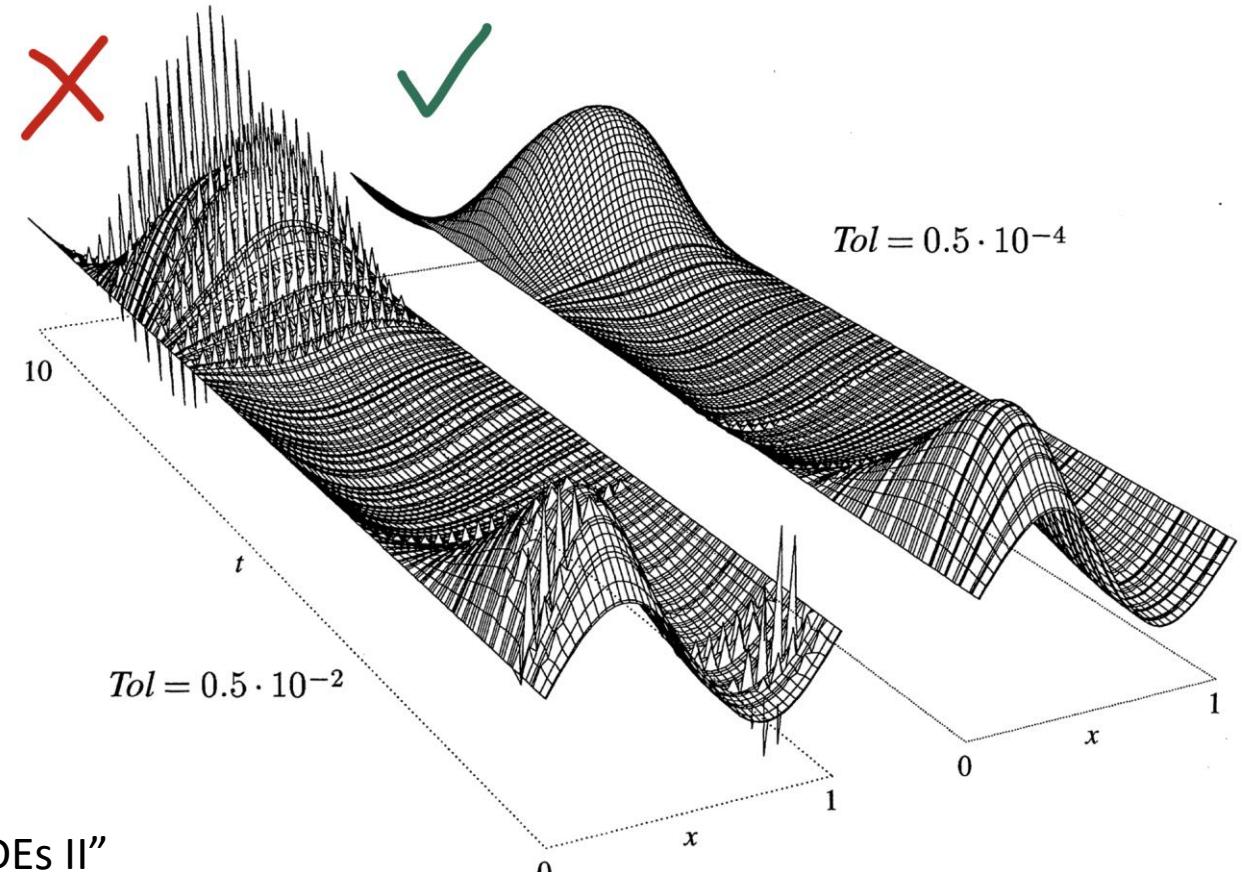


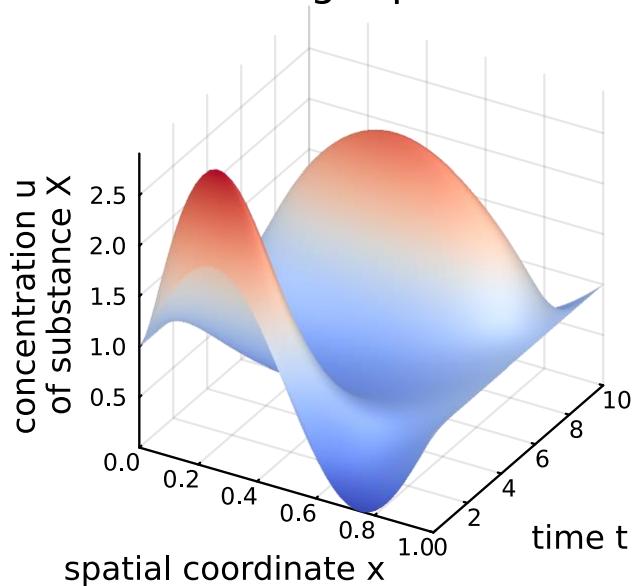
Fig. 1.7. Solution $u(x, t)$ of (1.6') with $N = 40$ using ODEX

Can we use an explicit method? | On Stiffness

Explicit Euler Method: $\underline{y}^{(n+1)} = \underline{y}^{(n)} + \Delta t \underline{f}(\underline{y}^{(n)})$

↓

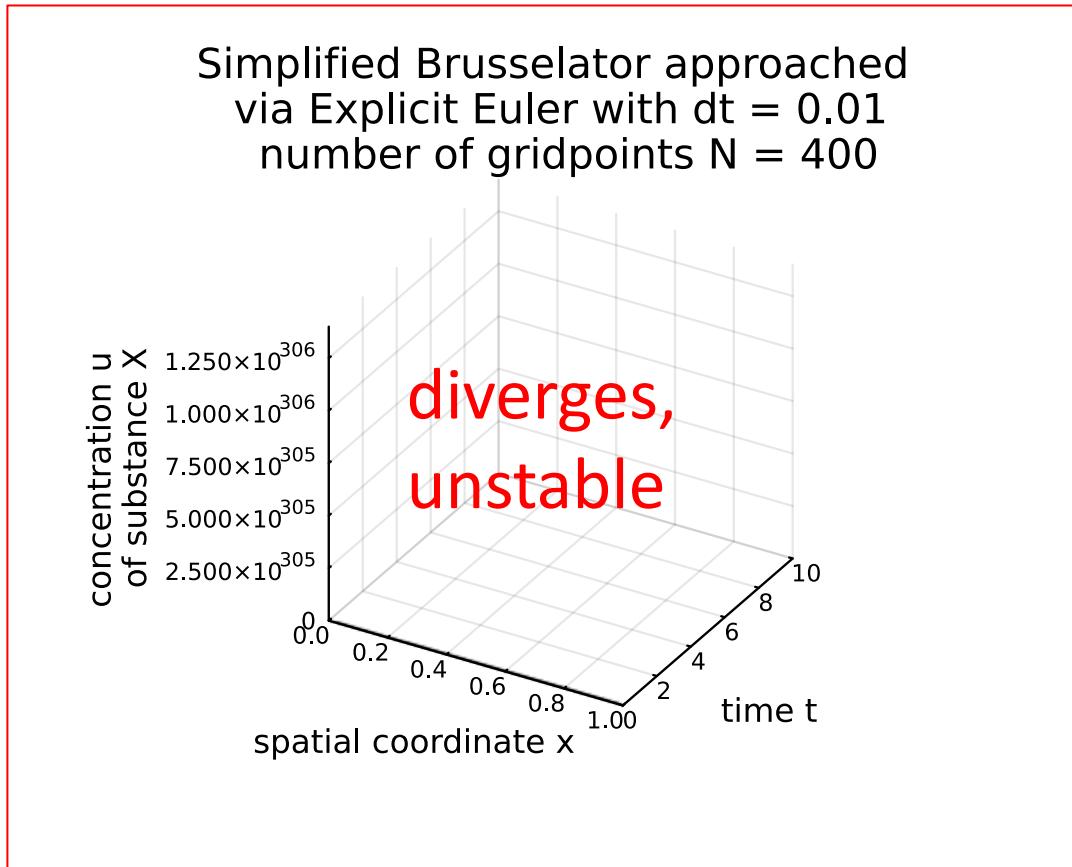
Simplified Brusselator approached
via Explicit Euler with $dt = 0.01$
number of gridpoints $N = 40$



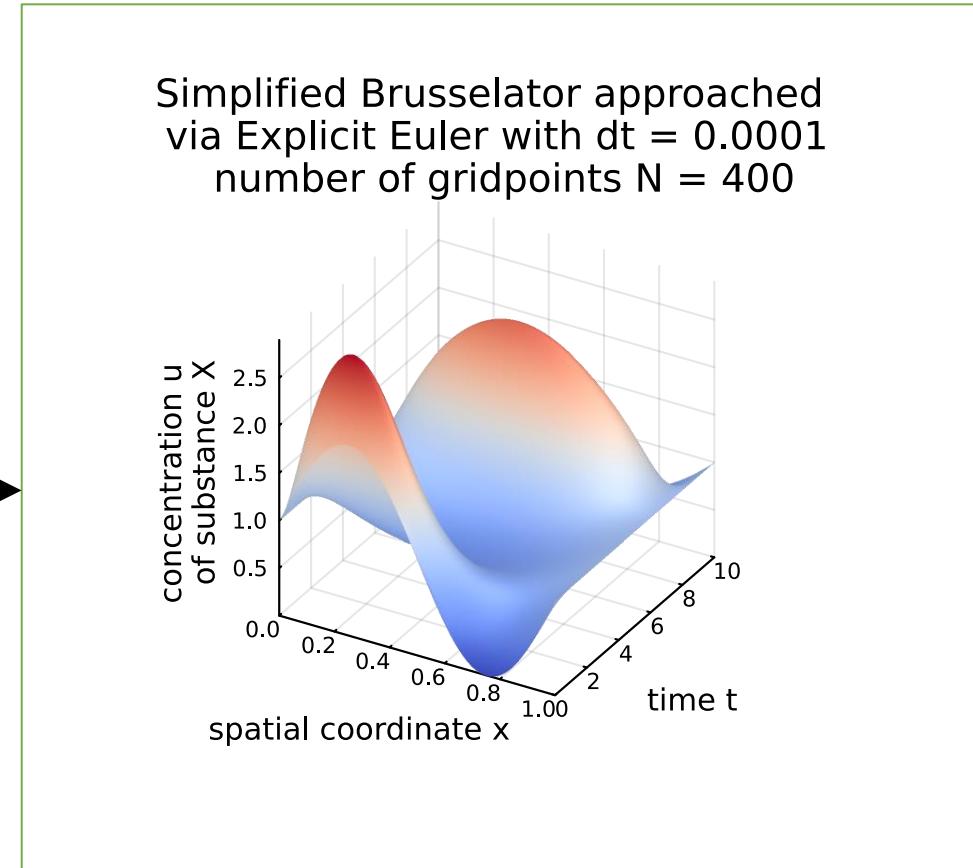
For $N = 40$ explicit Euler
gives the correct result for a
reasonably large stepsize.

The Occurrence of Stiffness

Increase N to 400



Decrease step-size



Default DifferentialEquations.jl Tsitouras 5/4
Runge-Kutta method: 220047 evaluations of f

A Practical Definition of Stiffness

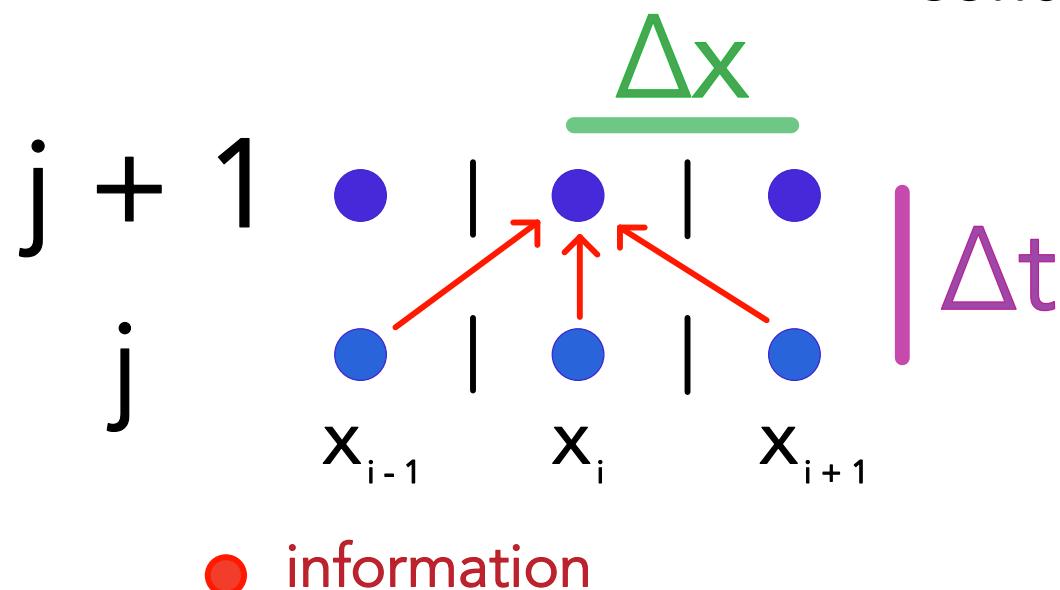
e. g. explicit Euler Method

»If a numerical method with a finite region of absolute stability, applied to a system with any initial conditions, is forced to use in a certain interval of integration a step length which is excessively small in relation to the smoothness of the exact solution in that interval, then the system is said to be stiff in that interval.«

J. D. Lambert, “Numerical Methods for ordinary Differential Systems”, chapter 6

Understanding Stiffness in a Diffusive Context

In every explicit step, only neighboring cells communicate.



Concentration spreads with $\sigma = \sqrt{2\alpha t}$

$$\Delta x > \sigma(\Delta t) = \sqrt{2\alpha\Delta t}$$

diffusion constant α

$$\Delta t < \frac{\Delta x^2}{2\alpha}, \quad \Delta x = \frac{1}{N + 1}$$

number N of gridpoints in $[0,1]$

Problem: Double resolution $\frac{\Delta x}{2} \rightarrow$ need $\mathcal{O}(N^2)$ more steps to cover the timespan, every step $\mathcal{O}(N)$ more complex $\rightarrow \mathcal{O}(N^3)$

The Implicit Approach | Implicit Euler Method

Implicit Euler Method for Solving Stiff ODEs $\partial_t \underline{y} = f(\underline{y})$

$$\text{implicit step } \underline{y}^{(n+1)} = \underline{y}^{(n)} + \Delta t f(\underline{y}^{(n+1)})$$

Formulate taking **one timestep** as root-finding problem

$$0 = \underline{y}^{(n+1)} - \underline{y}^{(n)} - \Delta t f(\underline{y}^{(n+1)}) =: g(\xi)$$

Solve **one timestep** iteratively using Newton iteration

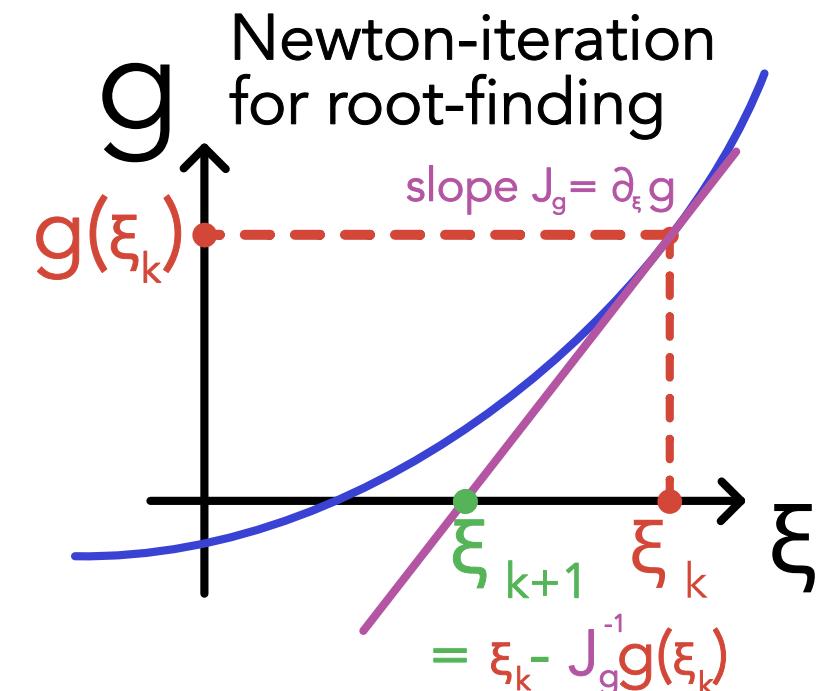
$$\xi_{k+1} = \xi_k - J_g^{-1}(\xi_k) g(\xi_k), \quad J_g = \underline{I} - \gamma(\Delta t) \underline{J}_f$$

$$\xi_0 = \underline{y}^{(n)}, \quad \xi_m \xrightarrow[m \rightarrow \infty]{} \underline{y}^{(n+1)}$$

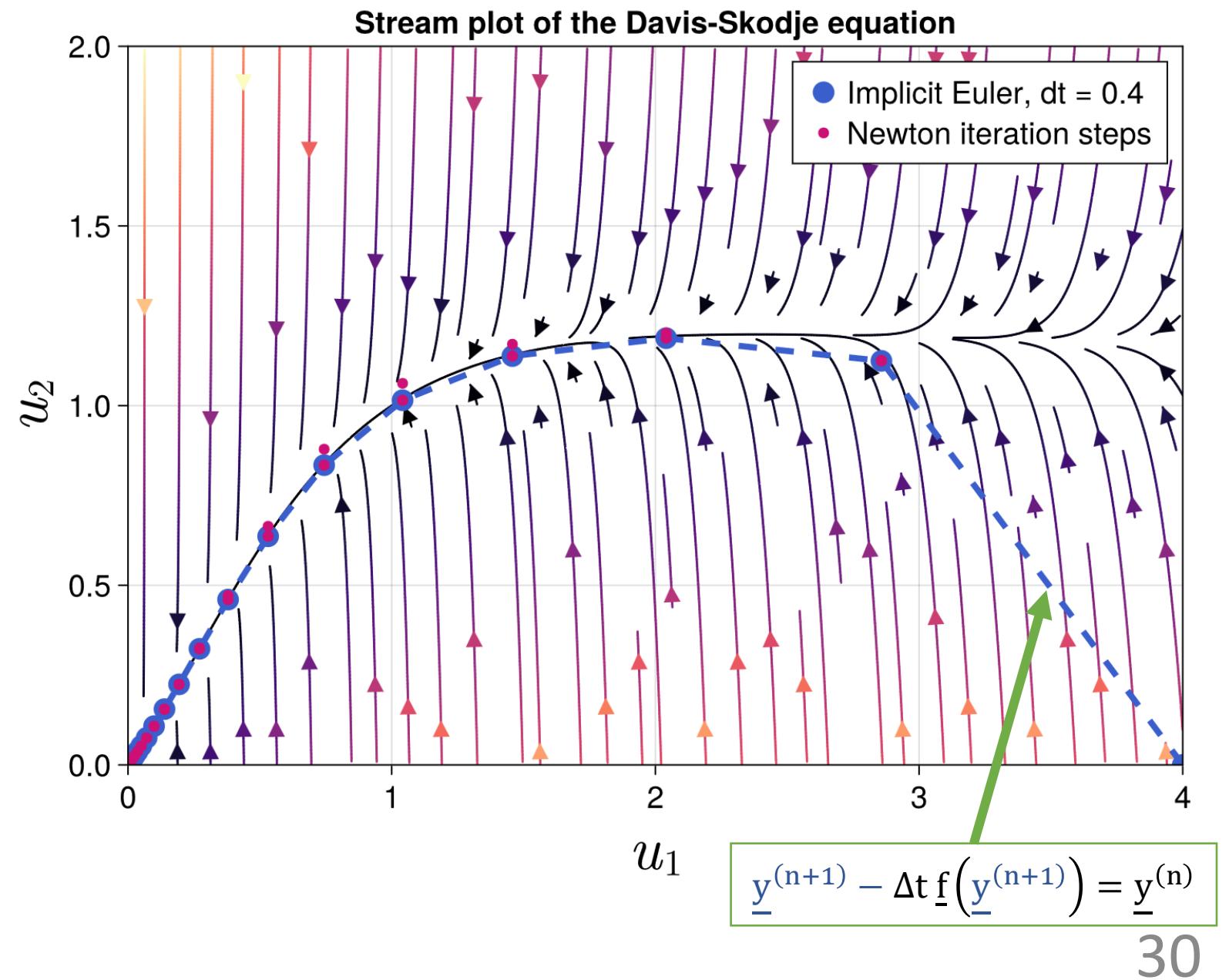
$$\text{alternatively quasi - Newton } \xi_{k+1} = \xi_k - J_g^{-1}(\xi_0) g(\xi_k)$$

In **each step of the Newton-iteration** calculate ξ_{k+1} from

$$\text{linear equation } b := g(\xi_k) = J_g(\xi_k - \xi_{k+1}) = J_g a$$



Goal: Make implicit step using (Quasi-) Newton



How to solve the linear system $\underline{g}(\xi_k) = \underline{J}_g \underline{a}$?

Remember: Calculate a Jacobian Vector product (directional derivative)

$$\underline{J}_g \underline{v} = (\nabla \cdot \underline{g}) \underline{v}, \quad \underline{J}_g \hat{\underline{e}}_j \text{ calculates } j - \text{th column of Jacobian}$$

to machine precision using one forward mode AD calculation

$$\underline{g}(\xi + \epsilon \underline{v}) = \underline{g}(\xi) + \epsilon \underline{J}_g \underline{v}, \quad \text{dual number } \xi + \epsilon \underline{v}, \quad \epsilon^2 = 0$$

Calculate \underline{J}_g efficiently and solve the linear system based on matrix decomposition.

Option A for solving the linear system involving the Jacobian

Solve the system only based on Jacobian-vector-products.

Option B for solving the linear system involving the Jacobian

Option A: Jacobi Calculation and Matrix

Decomposition for $\underline{g}(\underline{\xi}_k) = \underline{\underline{J}}_g \underline{a}$

1. Calculate all columns $\underline{\underline{J}}_g \hat{\underline{e}}_j$ of the Jacobian

$$\underline{g}(\underline{\xi} + \epsilon \hat{\underline{e}}_j) = \underline{g}(\underline{\xi}) + \epsilon \underline{\underline{J}}_g \hat{\underline{e}}_j$$
$$j = 1, \dots, 2(N+2)$$

$\rightarrow \mathcal{O}(N)$ evaluations of \underline{f}

$\rightarrow \mathcal{O}(N^2)$ complexity

in Newton iteration in every iterative step

$$\underline{\xi} = \underline{\xi}_k$$

in Quasi-Newton once per timestep

$$\underline{\xi} = \underline{y}_n$$

2. Do matrix decomposition, e. g.

$$\underline{\underline{P}} \underline{\underline{J}}_g = \underline{\underline{L}} \underline{\underline{R}} \quad (\text{in } \mathcal{O}(N^3))$$

$$\underline{\underline{J}}_g \underline{a} = \underline{g} \Leftrightarrow \underline{\underline{L}} \underline{\underline{R}} \underline{a} = \underline{g}$$

Q: Why can't we simply invert $\underline{\underline{J}}_g$?

3. Solve for \underline{a}

1. solve $\underline{\underline{L}} \underline{y} = \underline{g}$ for \underline{y}
(forward substitution)
2. solve $\underline{\underline{R}} \underline{a} = \underline{y}$ for \underline{y} by forward
(backward substitution)

Lower – left matrix $\underline{\underline{L}}$,

Upper – right matrix $\underline{\underline{R}}$,

permutation matrix $\underline{\underline{P}}$

Comparison between Newton and Quasi-Newton

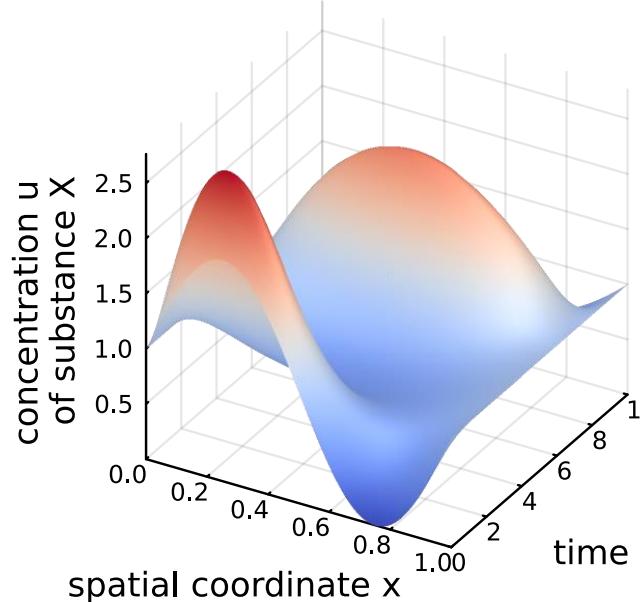
```
function newtons_method(f!, xi0, dt, tol, max_iter)
"""
The root of g_zero (and therefore the next step
in time in the implicit Euler scheme) is found
using Newton's method, which is implemented here.
"""

xi = copy(xi0)
for i in 1:max_iter
    #####
    # Here the LU decomposition is performed O(N^3)
    # the full Jacobian is constructed in every step
    luJ = lu(Jg(f!, xi, dt)) # Jg constructs
    #                               the Jacobian
    #####
    a = luJ \ g_zero(f!, xi, xi0, dt) # O(N^2)
    xi .= xi .- a
    if norm(g_zero(f!, xi, xi0, dt)) < tol
        return xi
    end
end
return xi
end
```

```
function quasi_newtons_method(f!, xi0, dt, tol, max_iter)
"""
The root of g_zero (and therefore the next step
in time in the implicit Euler scheme) is found using Newton's method,
which is implemented here.
"""

xi = copy(xi0)
#####
# Here a matrix LU decomposition is performed once
# per implicit Euler timestep, O(N^3)
luJ = lu(Jg(f!, xi, dt))
#####
for i in 1:max_iter
    a = luJ \ g_zero(f!, xi, xi0, dt) # O(N^2)
    xi .= xi .- a
    if norm(g_zero(f!, xi, xi0, dt)) < tol
        return xi
    end
end
return xi
end
```

Simplified Brusselator approached
via Implicit Euler $dt = 0.1$
number of gridpoints $N = 400$



Looks good and that at $\Delta t = 0.1$

Efficiency of our simple implementations

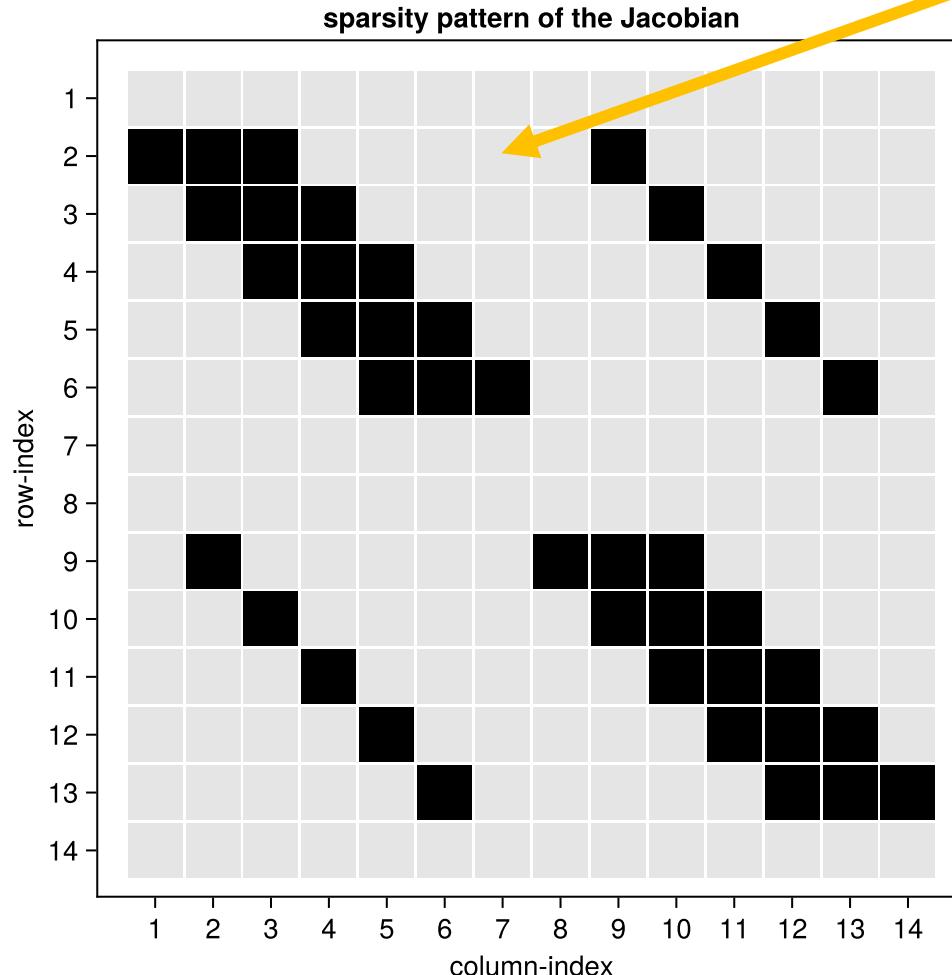
Explicit Euler, $dt = 0.0001$	Implicit Euler; Newton, $dt = 0.1$	Implicit Euler; Quasi- Newton, $dt = 0.1$
1.848 s (1447185 allocations: 5.95 GiB)	6.737 s (396055 allocations: 11.48 GiB)	3.144 s (197980 allocations: 5.57 GiB)

Problem I: Calculation of the Jacobian column-by-column is very expensive.

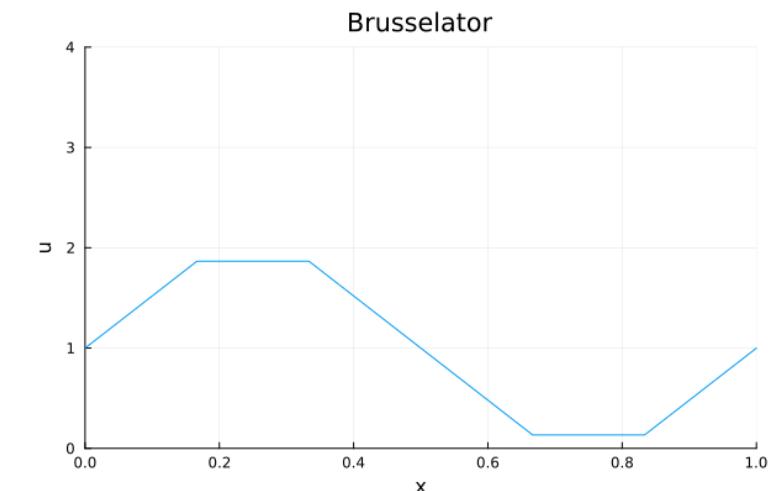
Problem II: The matrix factorization with $\mathcal{O}(N^3)$ is also expensive but a highly optimized problem (\rightarrow LAPACK)

Sparsity of the Jacobian

Examples for $N = 5$ for explainability.



$$\frac{\partial g_2}{\partial \xi_7} = 0$$



$$N = 400: \frac{\# \text{ non-zero elements}}{\# \text{ all elements}} \approx 6 \cdot 10^{-5}$$

Q: How can we use this sparsity?

Intermezzo: Non-Overlapping differentiations

Consider e. g.

Twiggling at x_1 and x_2 has no cross effects thus separates additively

$$\underline{f}(\underline{x}) = \begin{pmatrix} x_1 + x_3 \\ x_2 x_3 \\ x_1 \end{pmatrix}, \quad \underline{\underline{J}}_f = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} \end{pmatrix}, \quad \text{sparsity } \underline{\underline{S}}_f = \begin{pmatrix} & & \\ & & \\ & & \end{pmatrix}$$

where

$$\frac{\underline{f}(\underline{x} + h\underline{\hat{e}}_1) - \underline{f}(\underline{x})}{h} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \quad \frac{\underline{f}(\underline{x} + h\underline{\hat{e}}_2) - \underline{f}(\underline{x})}{h} = \begin{pmatrix} 0 \\ x_3 \\ 0 \end{pmatrix}$$

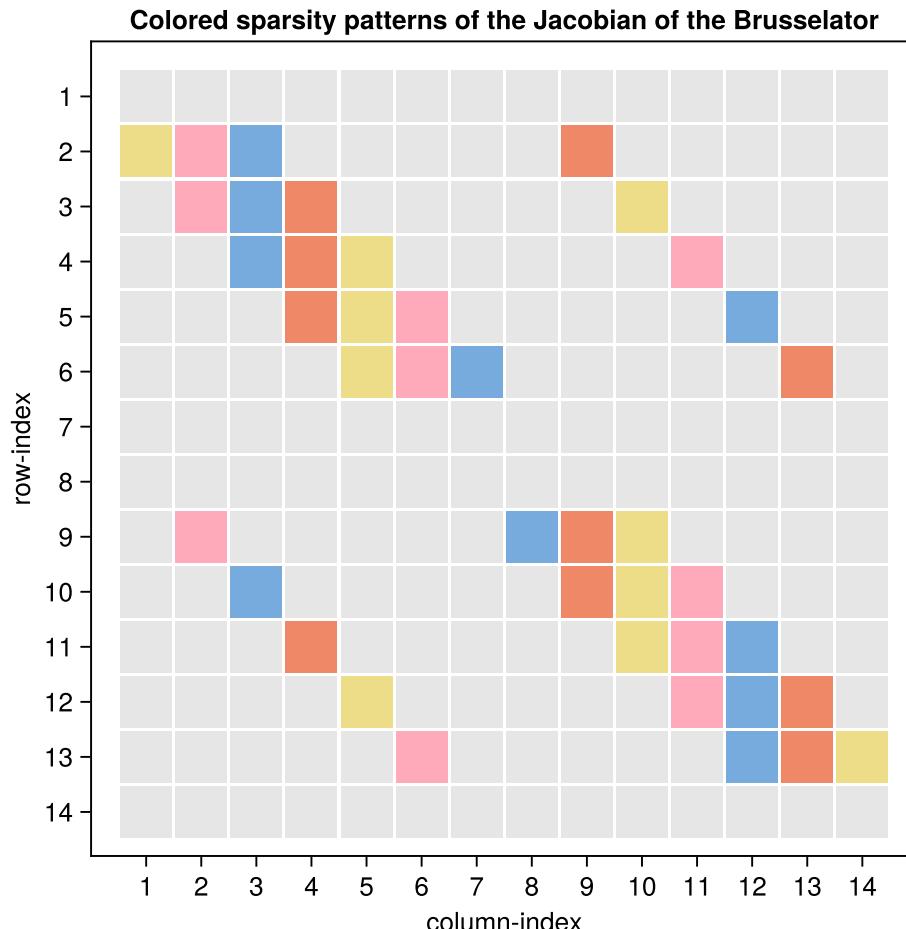
and

$$\frac{\underline{f}(\underline{x} + h\underline{\hat{e}}_1 + h\underline{\hat{e}}_2) - \underline{f}(\underline{x})}{h} = \begin{pmatrix} 1 \\ x_3 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ x_3 \\ 0 \end{pmatrix} \xrightarrow{\text{using } \underline{\underline{S}}_f} \underline{\underline{J}}_f = \begin{pmatrix} 1 & 0 & \vdots \\ 0 & x_3 & \vdots \\ 1 & 0 & \vdots \end{pmatrix}$$

Changing x_2 neither effects f_1 nor f_3

Idea: Calculate columns of the Jacobian in non-overlapping groups

Calculate e. g. the blue group in one forward pass $\underline{J}_g(\hat{e}_3 + \hat{e}_7 + \hat{e}_8 + \hat{e}_{12}) =: \underline{\text{JVP}}$



$$= \begin{pmatrix} 0 \\ (\text{JVP})_2 \\ (\text{JVP})_3 \\ (\text{JVP})_4 \\ (\text{JVP})_5 \\ (\text{JVP})_6 \\ 0 \\ 0 \\ (\text{JVP})_9 \\ (\text{JVP})_{10} \\ (\text{JVP})_{11} \\ (\text{JVP})_{12} \\ (\text{JVP})_{13} \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ (\text{JVP})_2 \\ (\text{JVP})_3 \\ (\text{JVP})_4 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ (\text{JVP})_{10} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Only 4 forward passes!

37

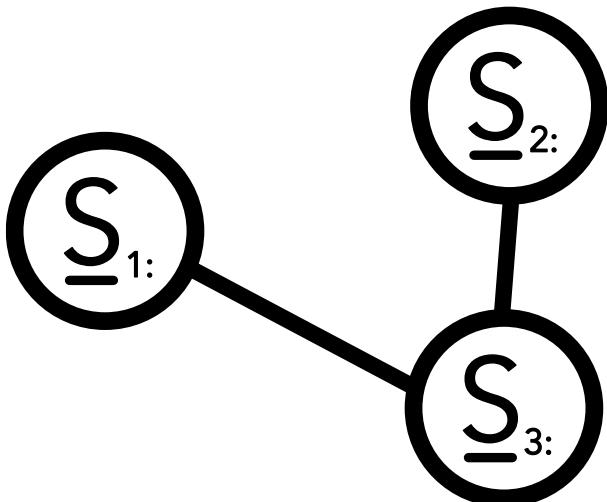
But how to find the grouping (coloring)?

Well known distance-1 graph coloring problem in disguise:

Nodes: columns of the sparsity of the Jacobian

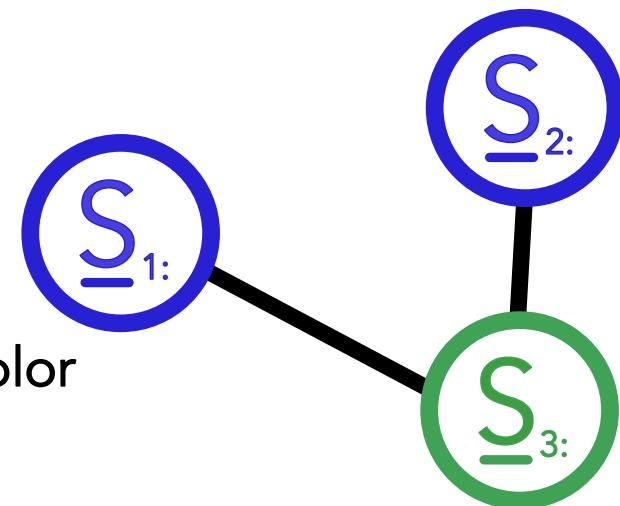
Edges: between overlapping columns

Simple example

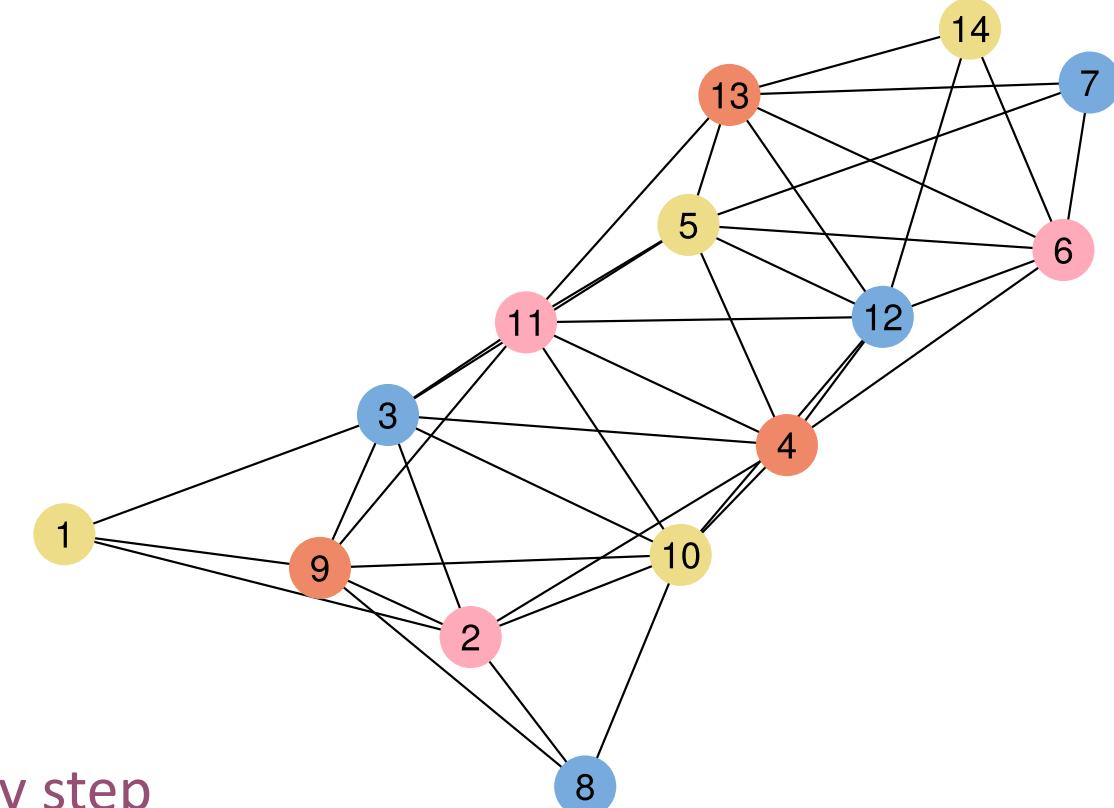
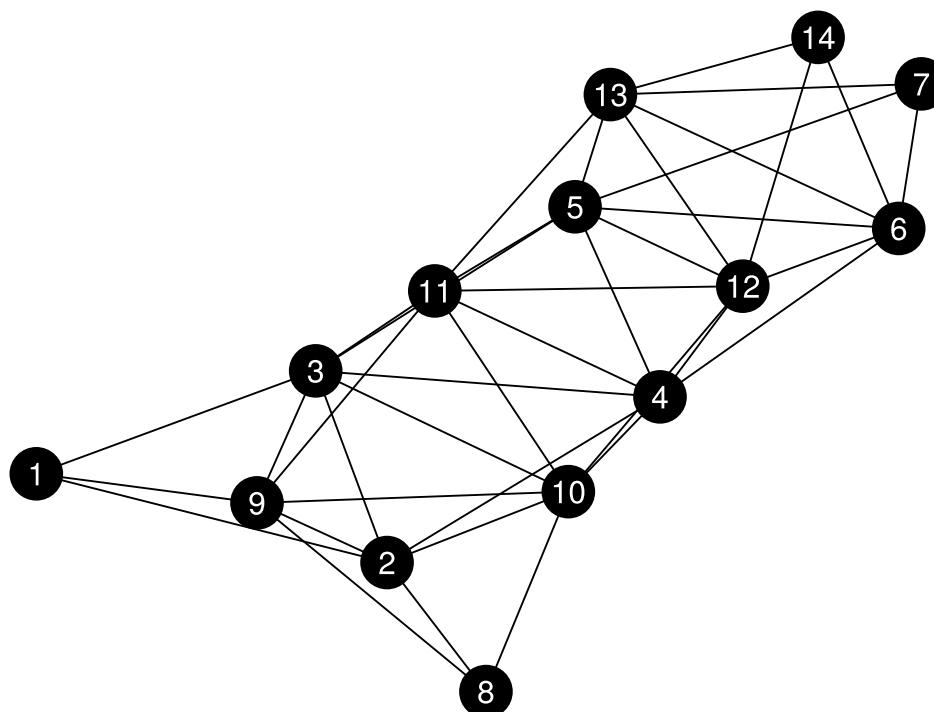


distance-1 coloring problem

color the nodes so that
no nodes directly connected
by an edge are of the same color
with as few colors as
possible



NP-complete → use approximate Greedy algorithm



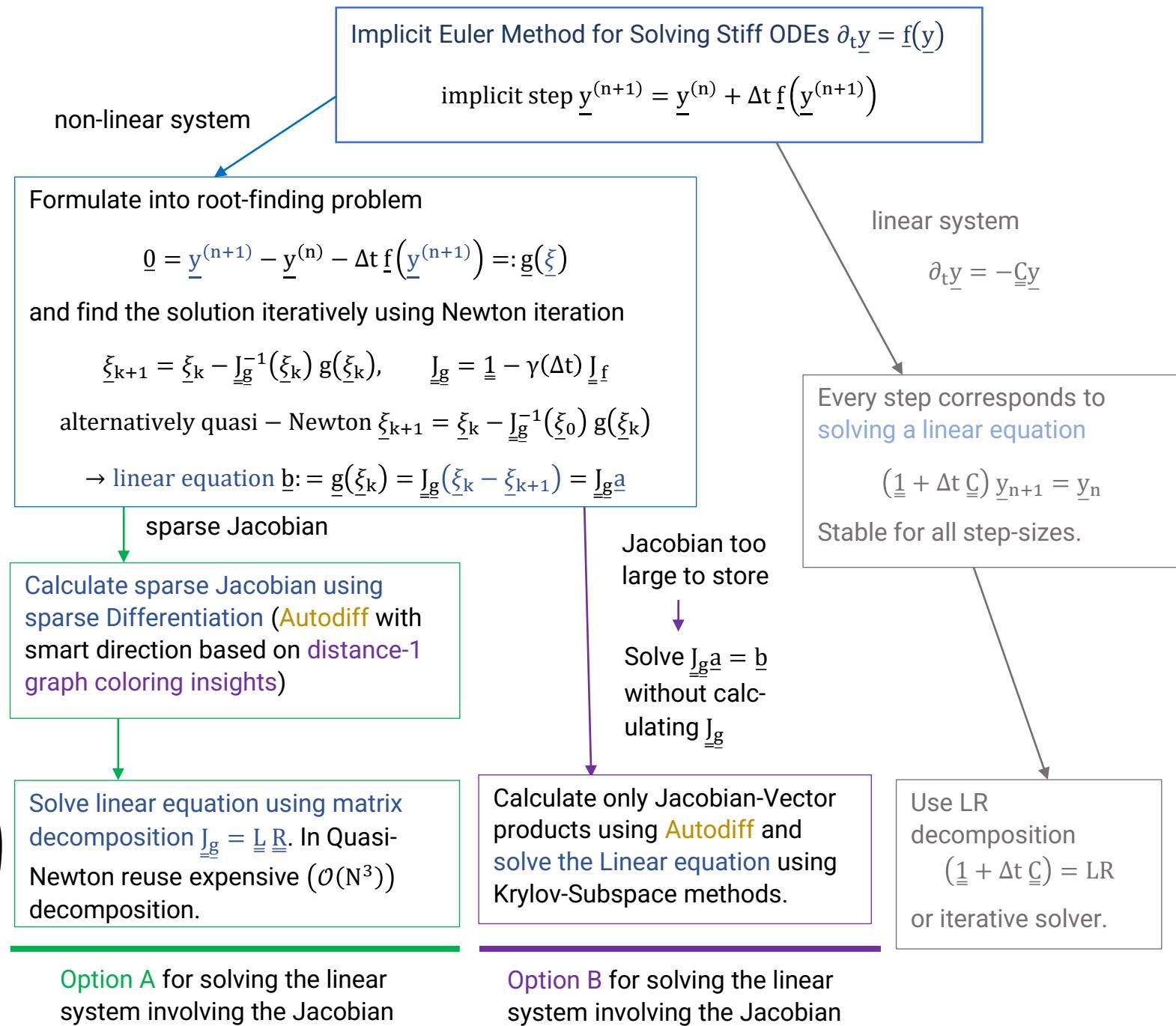
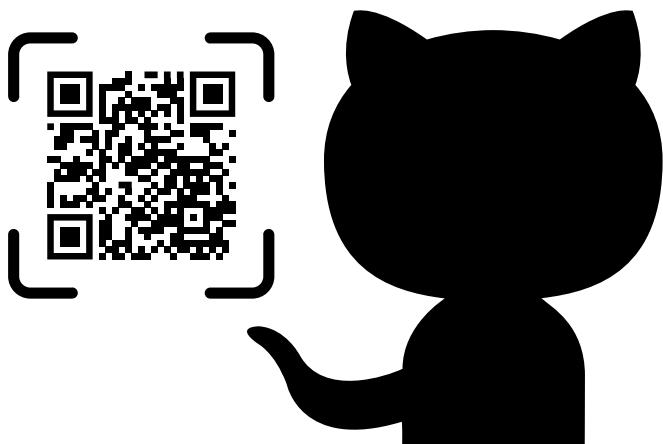
Greedy algorithm we used and step by step
application in appendix :)

Results

Implicit Euler; Quasi-Newton, dt = 0.1	Implicit Euler; Quasi-Newton, coloring, dt = 0.1	Implicit Euler; Quasi-Newton, coloring, sparse factorization , dt = 0.1
3.144 s (197980 allocations: 5.57 GiB)	877.994 ms (17404 allocations: 1.51 GiB)	88.977 ms (25192 allocations: 200.19 MiB)



Overview on Solving Stiff ODEs



Why should I care? – my library does that all automatically

Give the solver more information on your problem.

DifferentialEquations.jl solve method without sparsity pattern given	DifferentialEquations.jl solve method with sparsity pattern given
251.900 ms (63311 allocations: 211.75 MiB)	14.961 ms (33879 allocations: 27.41 MiB)

Further improvements

- General
 - Adaptive timestep control in combination with higher order methods like Rosenbrock or Semi-Implicit Extrapolation
- For Jacobi-free variant
 - Preconditioning → decrease the condition number of the matrix being solved → faster convergence of GMRES

Breakdown of Classical Methods & Outlook

- **Problem of strong turbulence:** Reynolds number limited for turbulent flow simulations (unreasonably small grid would be necessary) → Idea: handle small scale phenomena based on insights from Machine Learning [e. g. Kochkov et al is a nice read]
- **Problem of different scales:** Different scales can be very problematic (although there are adaptive methods, mesh free methods, tree algorithms etc.)
- **Curse of dimensionality for high-dimensional PDEs** (e. g. Schrödinger in Physics, Black-Scholes in finance): The computational cost for solving them goes up exponentially with the dimensionality. [Han et al, 2018]
- Need for inverse modeling
- How to blend differential equations with the vast data sets?
- ...

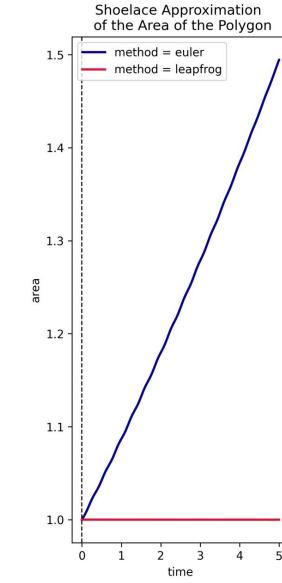
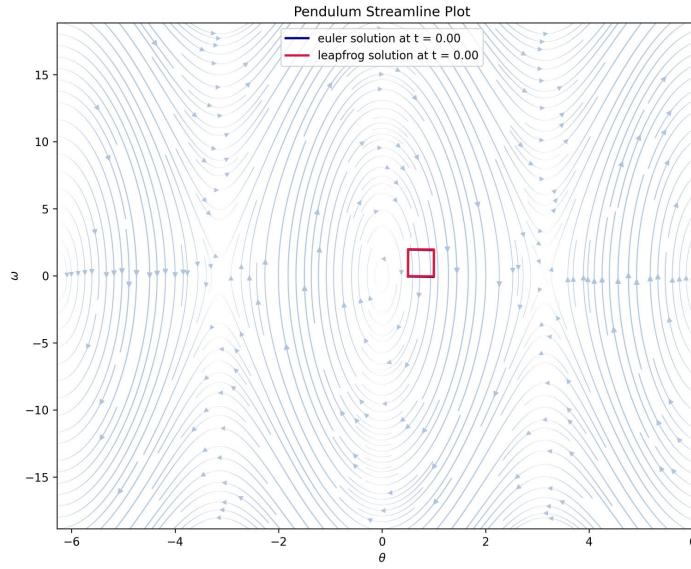
TODO

Key insights

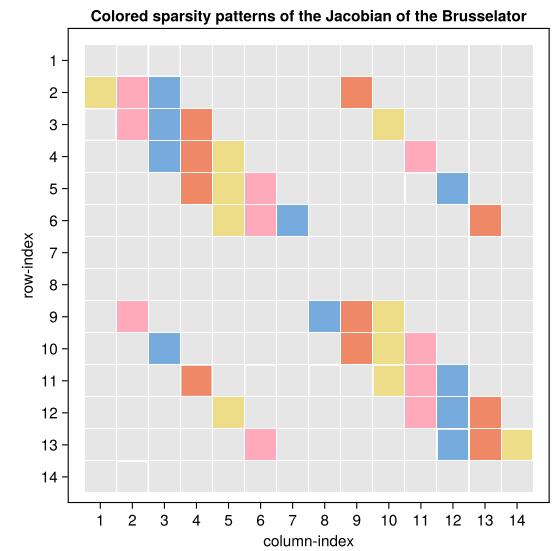
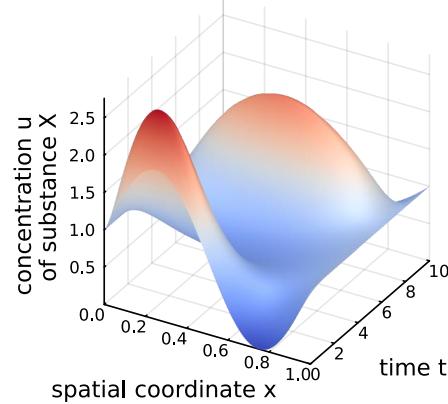
For integrations over long timespan where conservation properties should be obeyed, use symplectic integrators.

Stiffness is an efficiency problem.

Stiff problems require special approaches where you need to mind the sparsity of your Jacobian.



Simplified Brusselator approached via Implicit Euler $dt = 0.1$
number of gridpoints $N = 400$



References

General Literature used

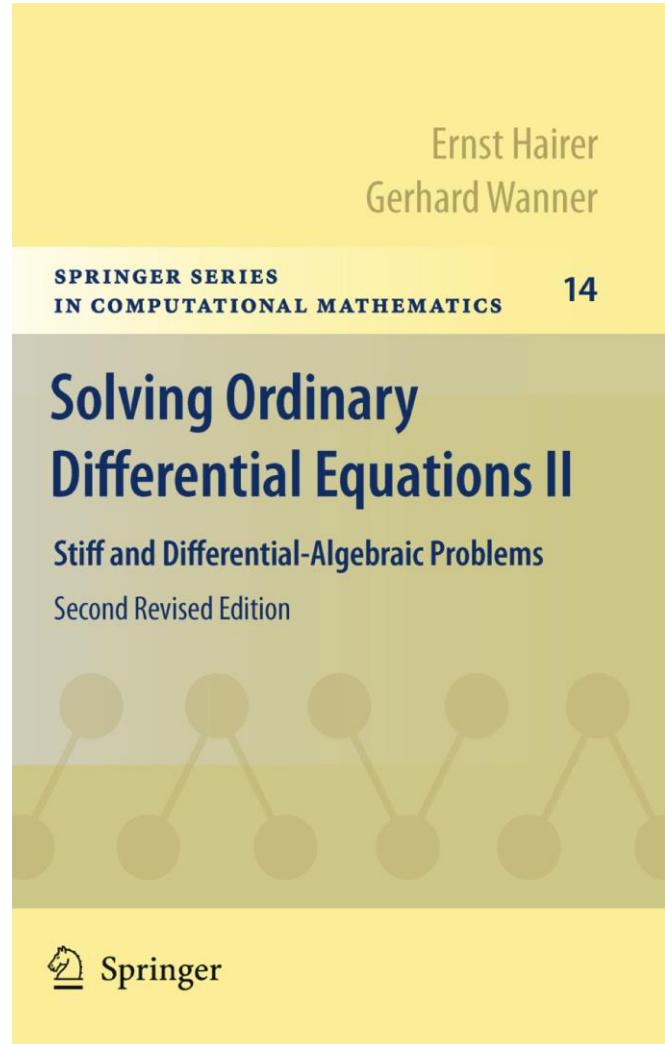
(see [criticism](#))

E. Hairer
S. P. Nørsett
G. Wanner

Solving Ordinary Differential Equations I

Nonstiff Problems

Second Revised Edition
With 135 Figures



NUMERICAL RECIPES

The Art of Scientific Computing
Third Edition

William H. Press

Raymer Chair in Computer Sciences and Integrative Biology
The University of Texas at Austin

Saul A. Teukolsky

Hans A. Bethe Professor of Physics and Astrophysics
Cornell University

William T. Vetterling

Research Fellow and Director of Image Science
ZINK Imaging, LLC

Brian P. Flannery

Science, Strategy and Programs Manager
Exxon Mobil Corporation



Ernst Hairer
Christian Lubich
Gerhard Wanner

Geometric Numerical Integration

Structure-Preserving Algorithms
for Ordinary Differential Equations

Second Edition

With 146 Figures



NUMERICAL METHODS FOR ORDINARY DIFFERENTIAL SYSTEMS

The Initial Value Problem

J. D. Lambert

*Professor of Numerical Analysis,
University of Dundee, Scotland*

JOHN WILEY & SONS
Chichester · New York · Brisbane · Toronto · Singapore

Lecture Notes for MVComp 1
(winter term 2022/2023)

Fundamentals of Simulation Methods

originally by Volker Springel (MPA)

modified and extended by

Christoph Pfrommer (AIP), Philipp Girichidis (ZAH),
Ralf Klessen (ZAH), Cornelis Dullemond (ZAH),
Frauke Gräter (HITS), Rüdiger Pakmor (MPA),
and Fritz Röpke (HITS/ZAH)

February 3, 2023

Main general resource

[“Parallel Computing and Scientific Machine Learning \(SciML\): Methods and Applications”](#), mainly chapter 9., lecture notes by Chris Rackauckas

Differential equations in Julia

[Documentation of the DifferentialEquations.jl package](#)

Papers

- Han, J., Jentzen, A., & E, W. (2018). Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34), 8505–8510.
doi:10.1073/pnas.1718942115
- Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., & Hoyer, S. (2021). Machine learning–accelerated computational fluid dynamics. In *Proceedings of the National Academy of Sciences* (Vol. 118, Issue 21). *Proceedings of the National Academy of Sciences*.
<https://doi.org/10.1073/pnas.2101784118>

Appendix

Option B: “Jacobian-free” method

Problem of option A: What if the Jacobian is too large to store?

Q: How to solve a linear system

$$\underline{A}\underline{x} = \underline{b}$$

in our case $\underline{J}_g\underline{a} = \underline{g}$

only based on Matrix
(Jacobian)-Vector products?

Idea 1: Gradient descent on the quadratic form

$$\underline{J}_g\underline{a} = \underline{g} \Leftrightarrow G(\underline{a}) \doteq \frac{1}{2} \underline{a}^T \underline{J}_g \underline{a} - \underline{a}^T \underline{g}$$

is minimal for $\underline{A} \in \mathbb{R}^{n \times n}$ symmetric and positive definite,

$$\text{so } \underline{a}^T \underline{J}_g \underline{a} > 0 \quad \forall \underline{a} \in \mathbb{K}^n \setminus \{0\}$$

→ unique minimizer

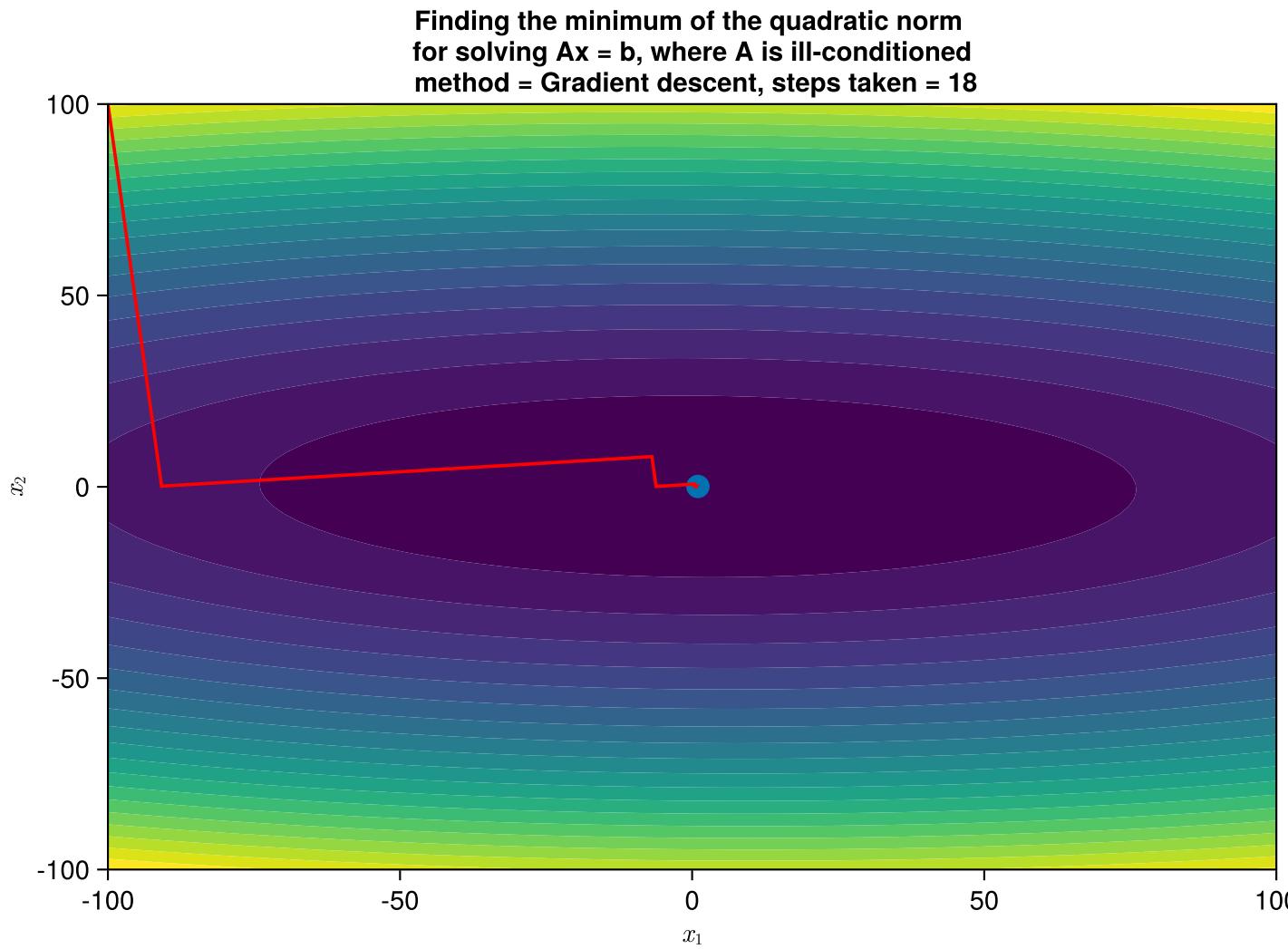
Gradient descent

$$\underline{a}_{k+1} = \underline{a}_k + \alpha(\underline{a}_k) \underline{r}_k$$

$$\text{residual } \underline{r}_k = \underline{g} - \underline{J}_g \underline{a}_k$$

with smart step – size α

Problem of gradient descent: Non-optimal sequence of search directions.



Consider the steps in Gradient descent for $\alpha = 1$, $\underline{a}_0 = \underline{0}$, $\underline{a}_1 = \underline{g}$

$$\underline{a}_2 = \underline{a}_1 + (\underline{g} - \underline{\underline{J}}_g \underline{a}_1) = 2\underline{g} - \underline{\underline{J}}_g \underline{g}$$

$$\underline{a}_3 = \underline{a}_2 + (\underline{g} - \underline{\underline{J}}_g \underline{a}_2) = 3\underline{g} - 3\underline{\underline{J}}_g \underline{g} + \underline{\underline{J}}_g^2 \underline{g}$$

⋮

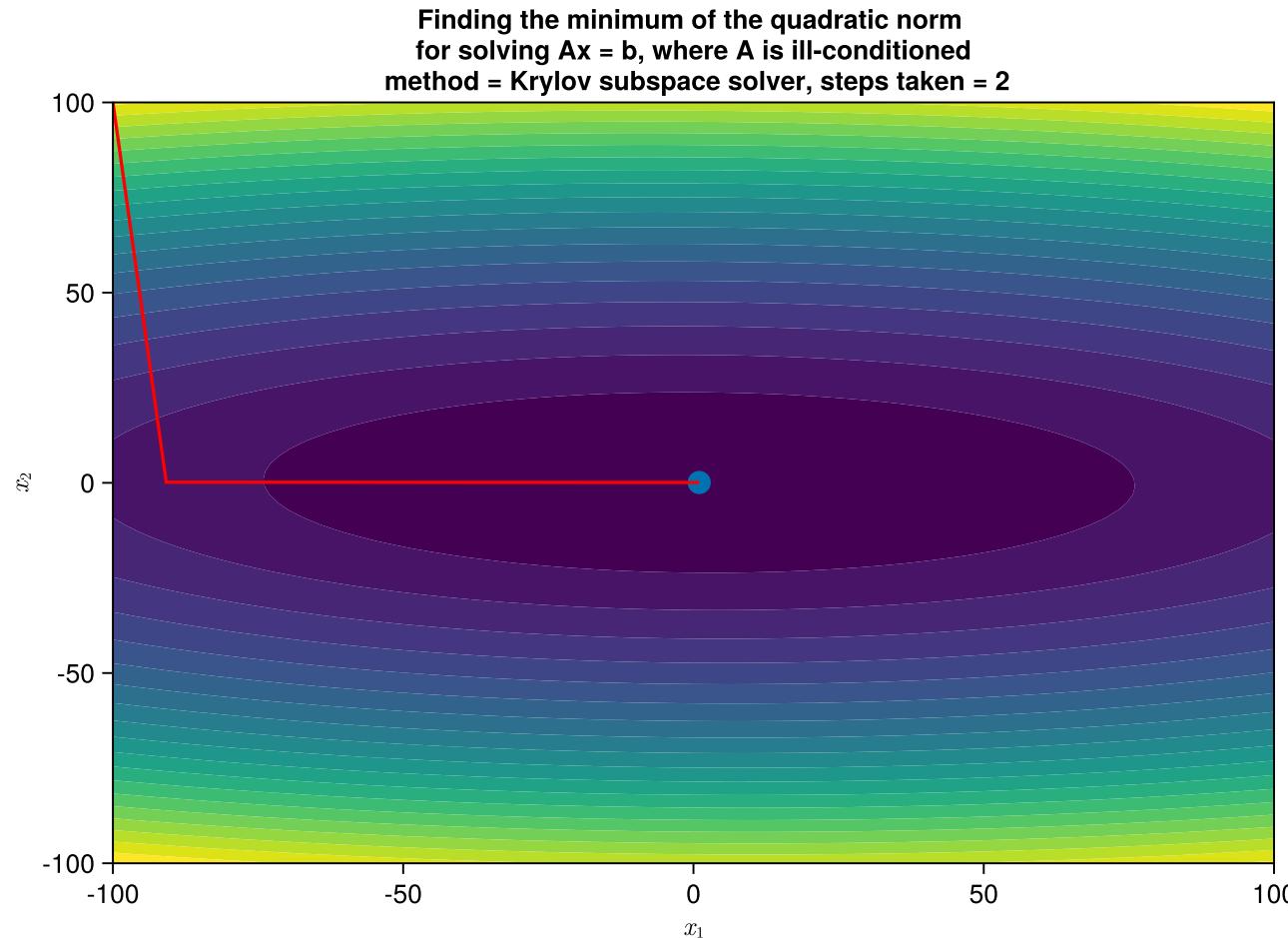


$$\text{Krylov subspace } \mathcal{K}_r := \mathcal{K}_r(\underline{\underline{J}}_g, \underline{g}) := \text{span}(\underline{b}, \underline{\underline{J}}_g \underline{g}, \underline{\underline{J}}_g^2 \underline{g}, \dots, \underline{\underline{J}}_g^{r-1} \underline{g})$$

$$\mathcal{K}_0 \subseteq \mathcal{K}_1 \subseteq \mathcal{K}_2 \subseteq \dots, \quad \mathcal{K}_n \text{ spans } \mathbb{R}^n \text{ if } \underline{\underline{J}}_g \text{ is invertible}$$

Krylov subspace can be easily constructed! → Are there better directions from \mathcal{K}_r ? 55

Teaser: Conjugate Gradient method



Number of
steps at most
size of linear
system.

Krylov subspace methods

Choose \underline{a}_{k+1} so that residual $\underline{r}_{k+1} = \underline{g} - \underline{\underline{J}}_g \underline{a}_{k+1}$

- a. is orthogonal to $\mathcal{K}_{k+1} \rightarrow$ Conjugate Gradient method
minimizing the quadratic form
- b. has minimum norm for $\underline{a}_k \in \mathcal{K}_{k+1} \rightarrow$ GMRES

 works generally for $\underline{\underline{J}}_g$ invertible

Conjugate Gradient method only works if $\underline{\underline{J}}_g$ is symmetric positive definite \rightarrow use GMRES for Brusselator problem

Proof: Jacobian-Vector products are directional derivatives

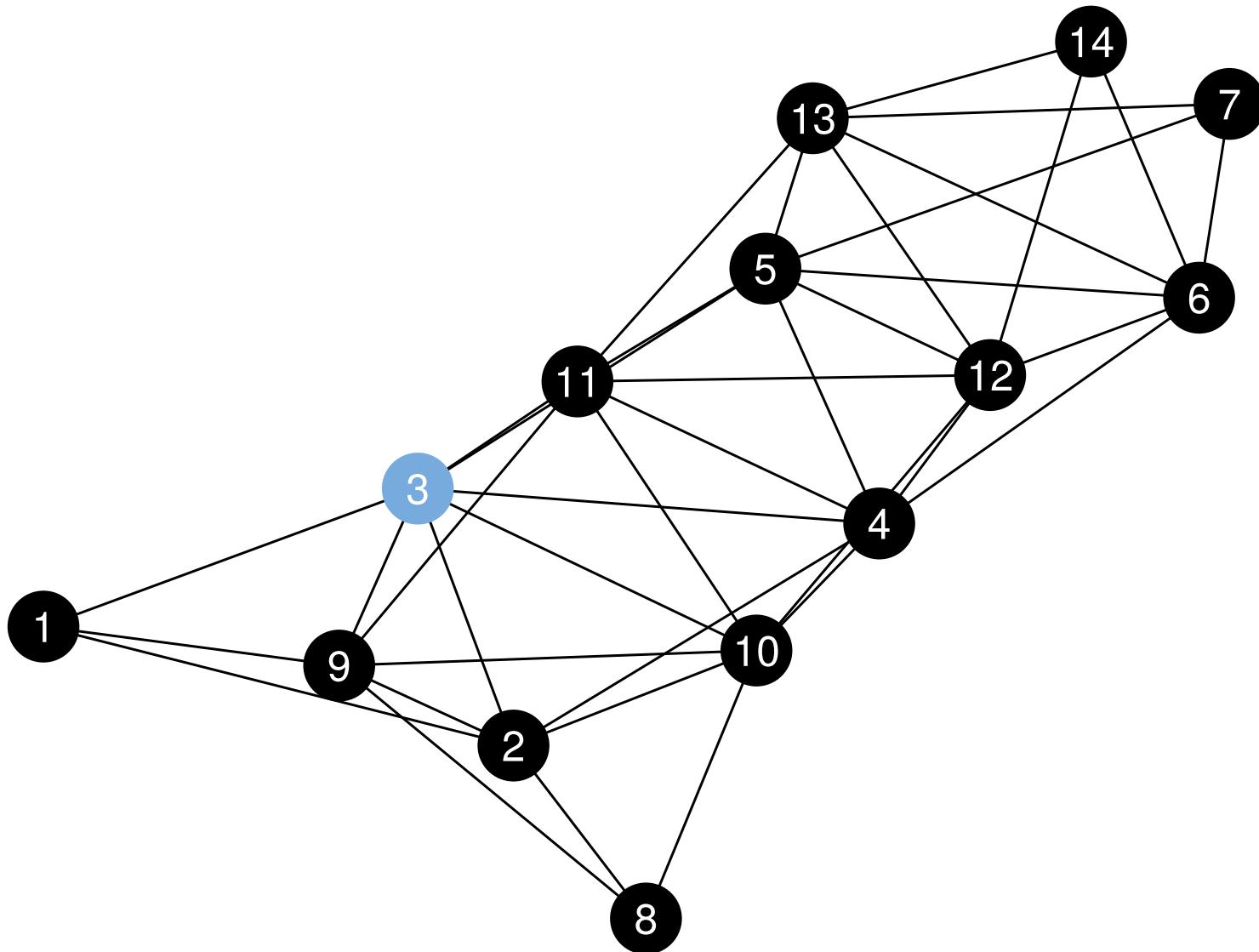
$$\underline{J}_g \underline{v} = (\underline{\nabla} \cdot \underline{g}) \underline{v} = \lim_{h \rightarrow 0} \frac{\underline{g}(\underline{x} + \underline{v}h) - \underline{g}(\underline{x})}{h}$$

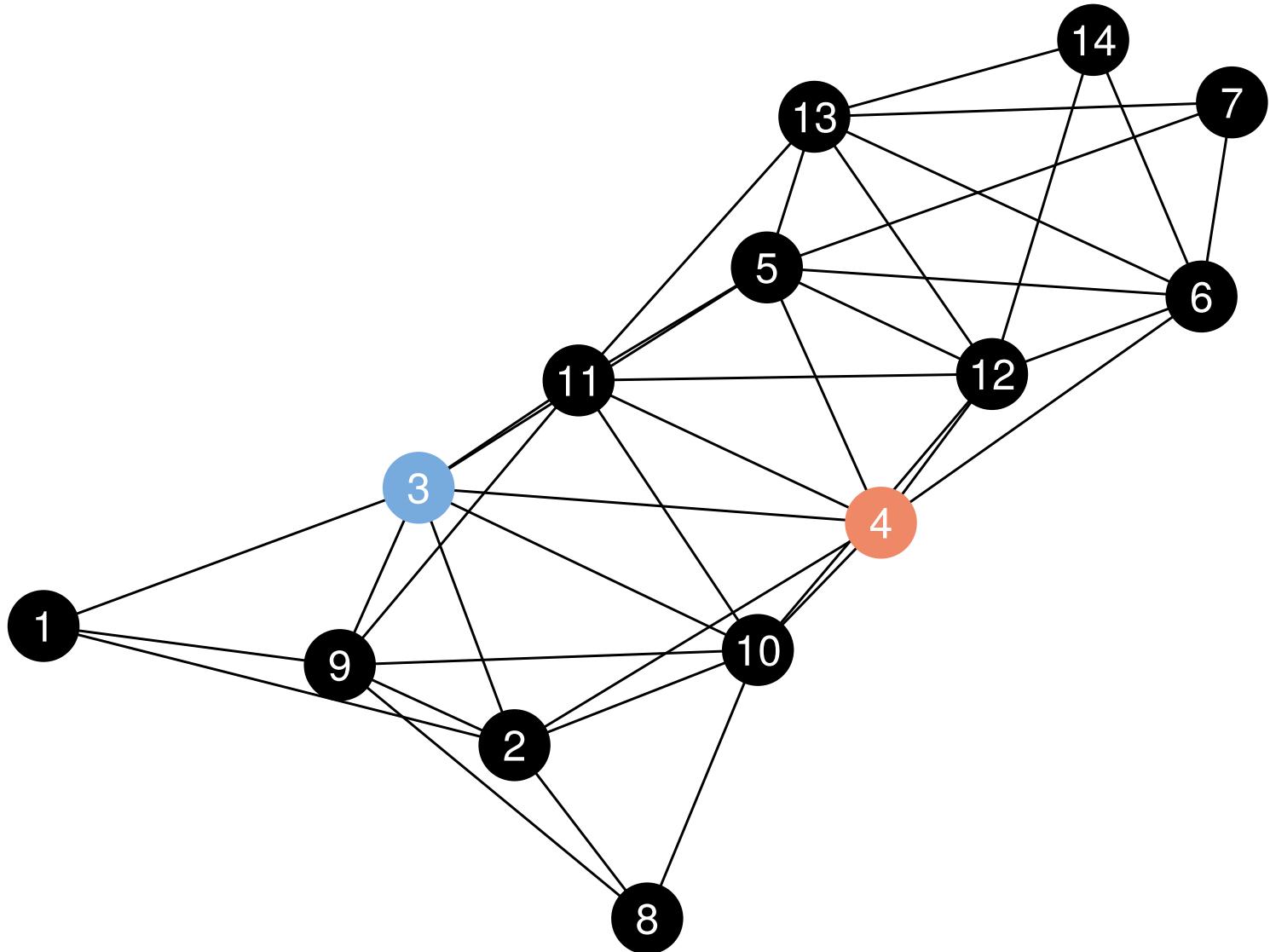
$$\left(\text{as } (\underline{J}_g \underline{v})_i = \sum_{j=1}^n (J_g)_{ij} v_j = \sum_{j=1}^n \frac{\partial g_i}{\partial x_j} v_j = ((\underline{\nabla} \cdot \underline{g}) \underline{v})_i \right)$$

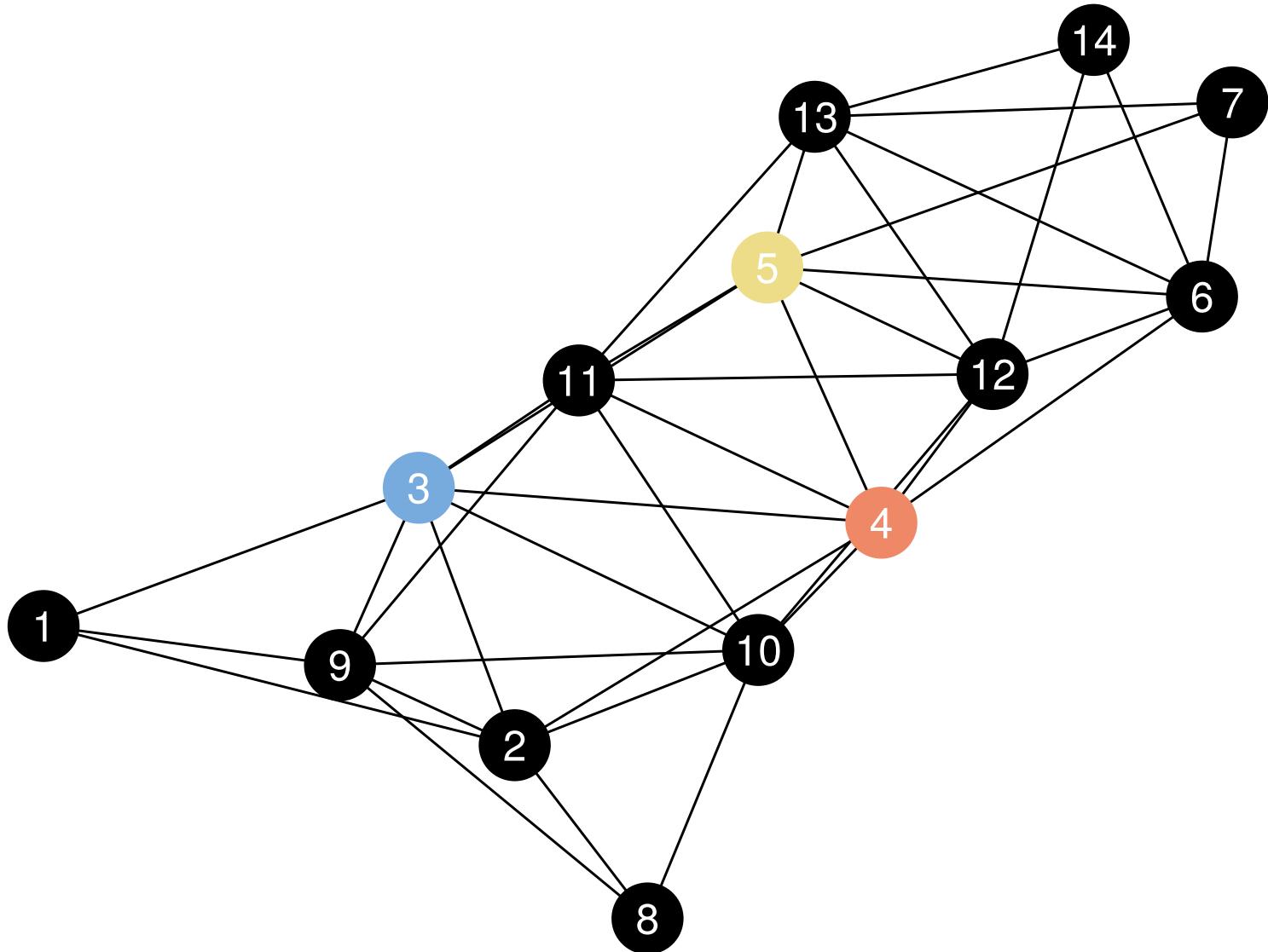
Greedy coloring

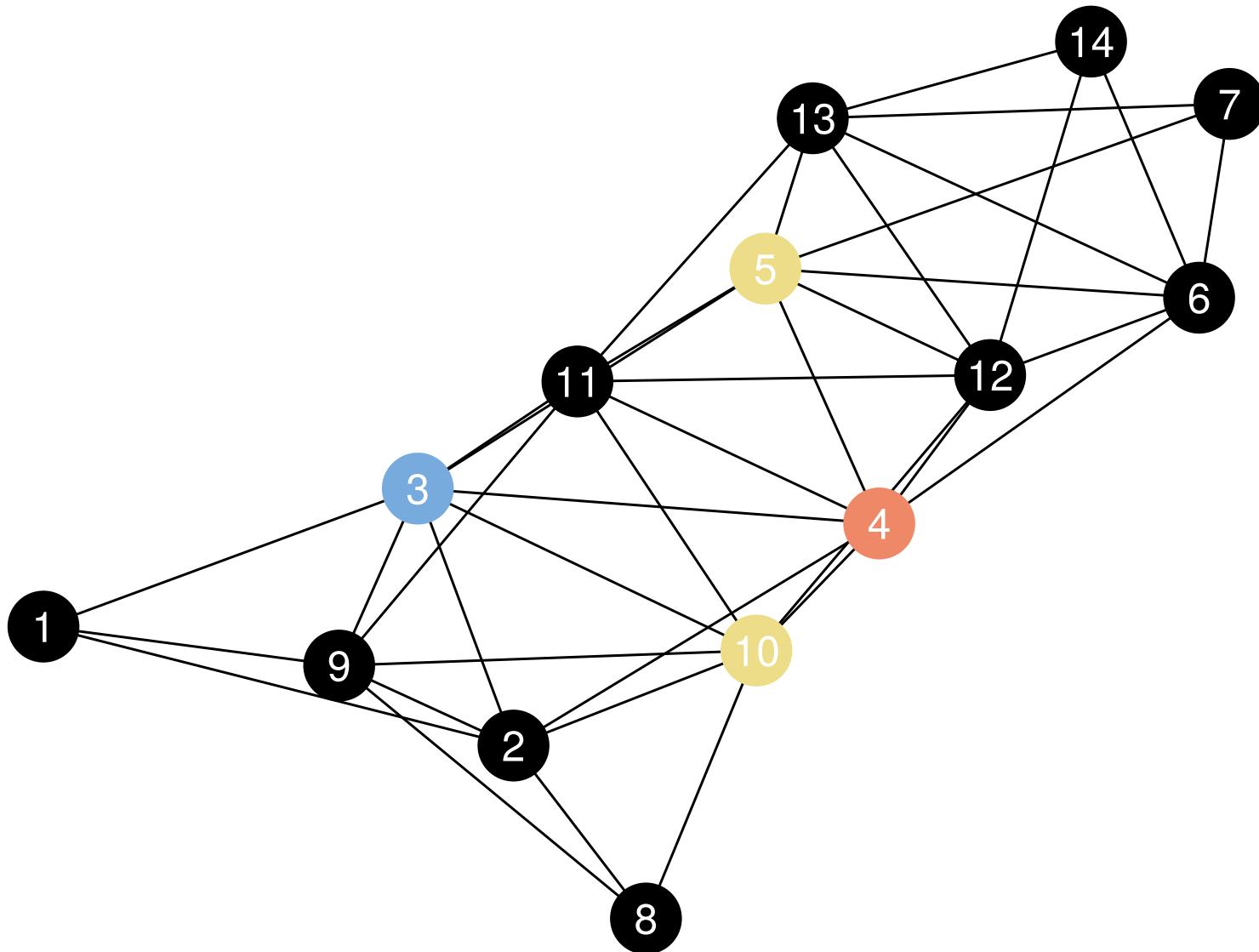
Practically we do not use the optimal solution but some good coloring found by a **greedy algorithm** (the following one worked the best for our problem)

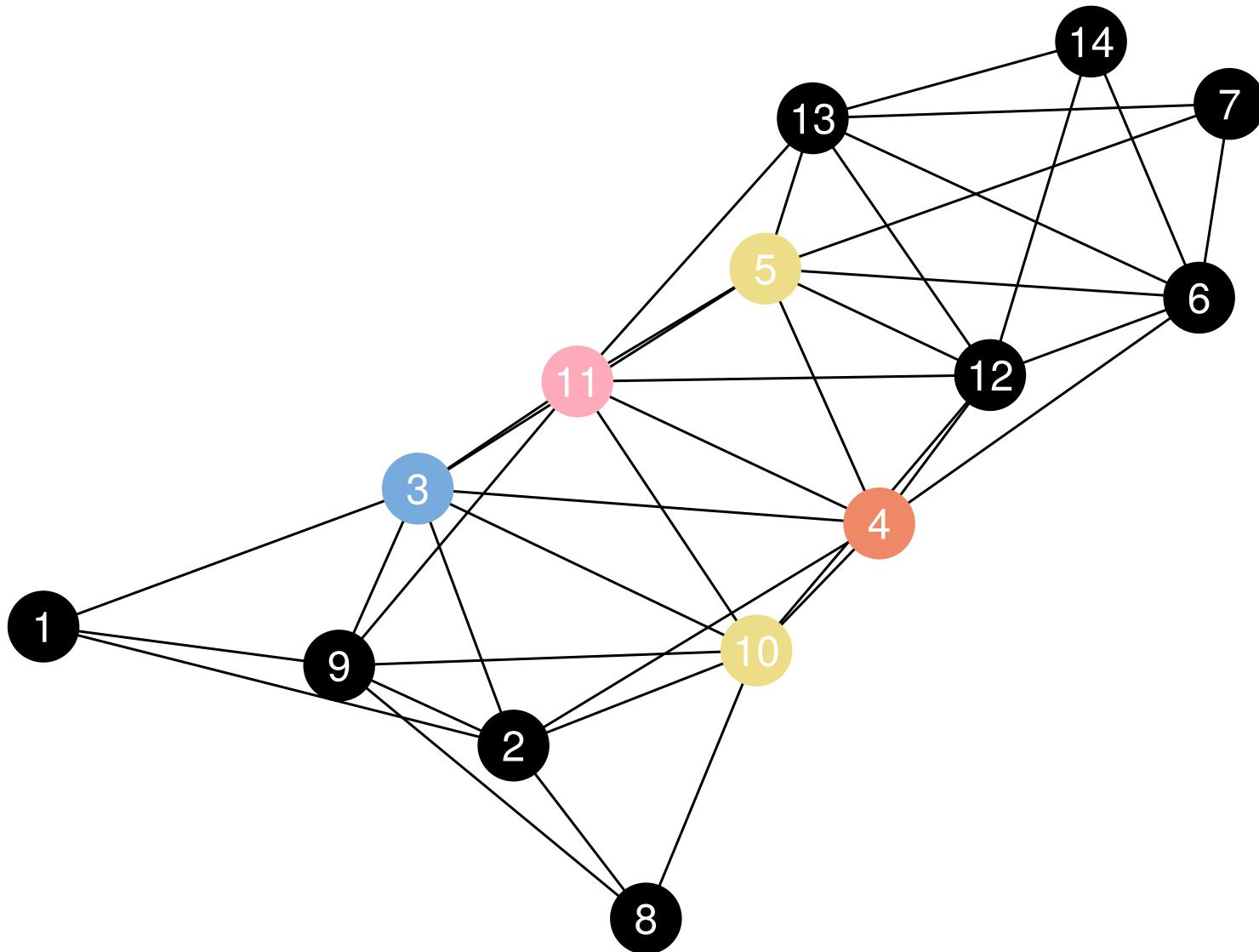
1. Generate a reverse-ordered list of the nodes by the node-degree, i. e. number of edges incident on a node
2. Go through this list (i. e. highest degree first) and choose the smallest color index possible for every node

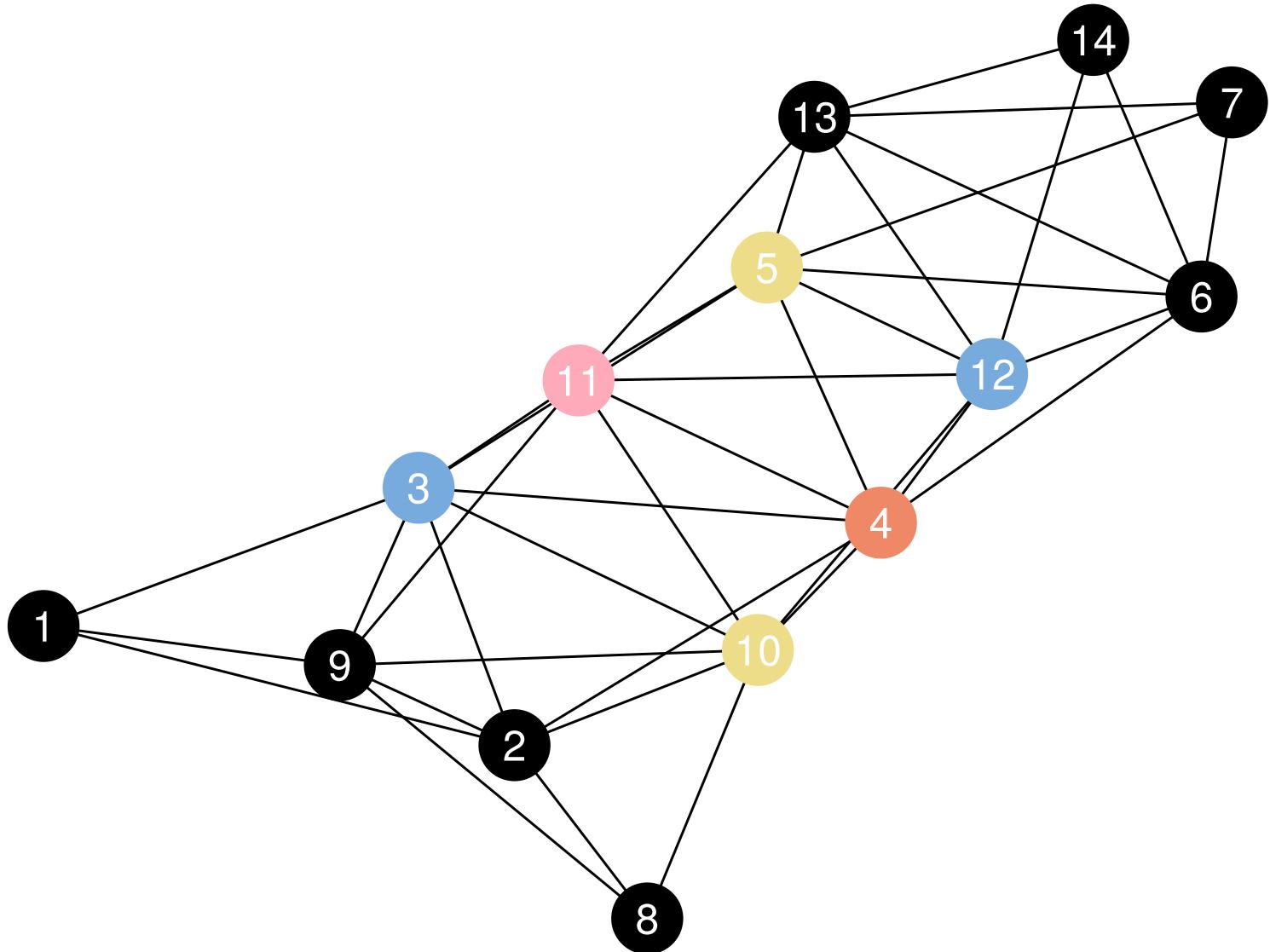


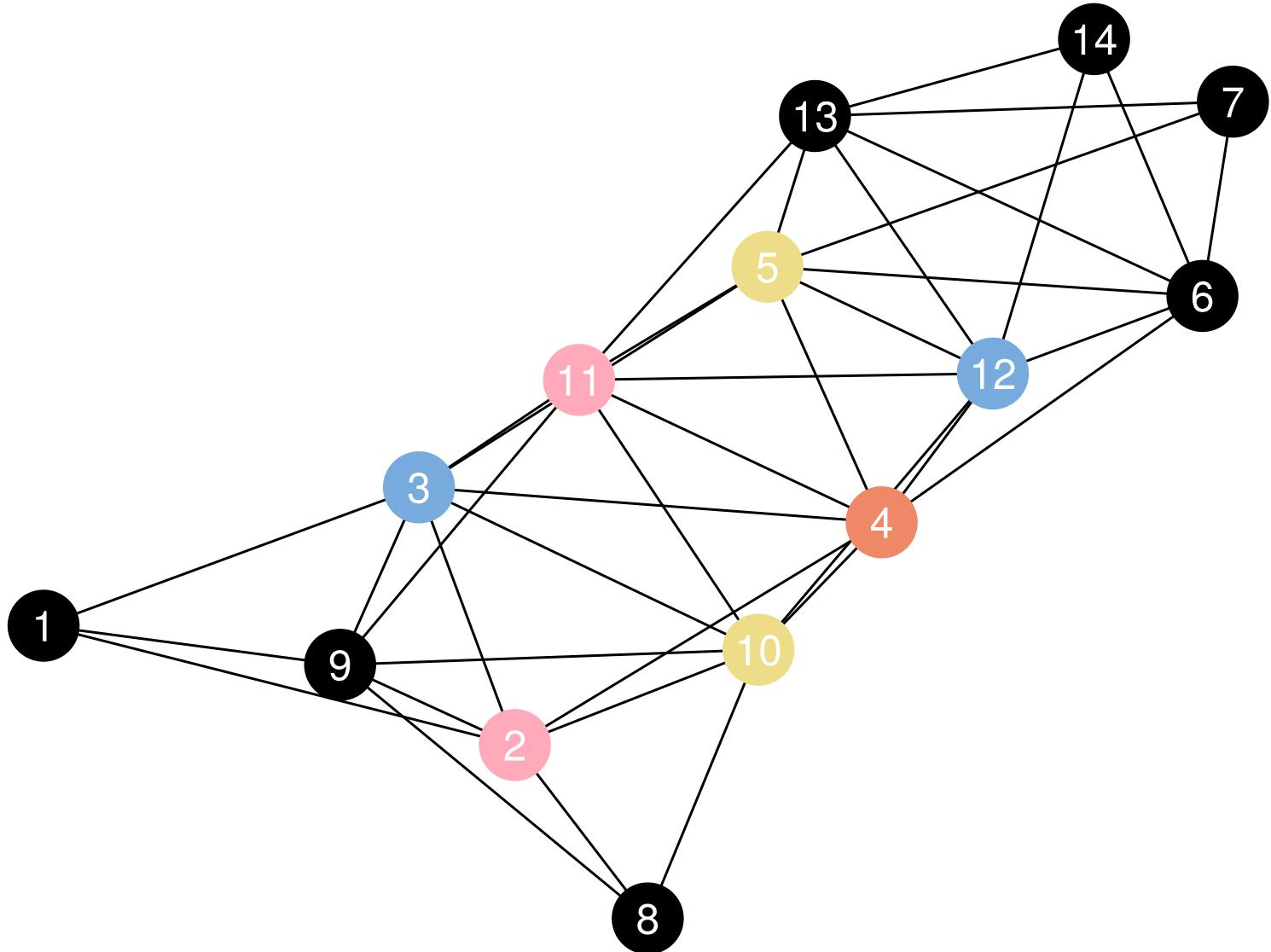


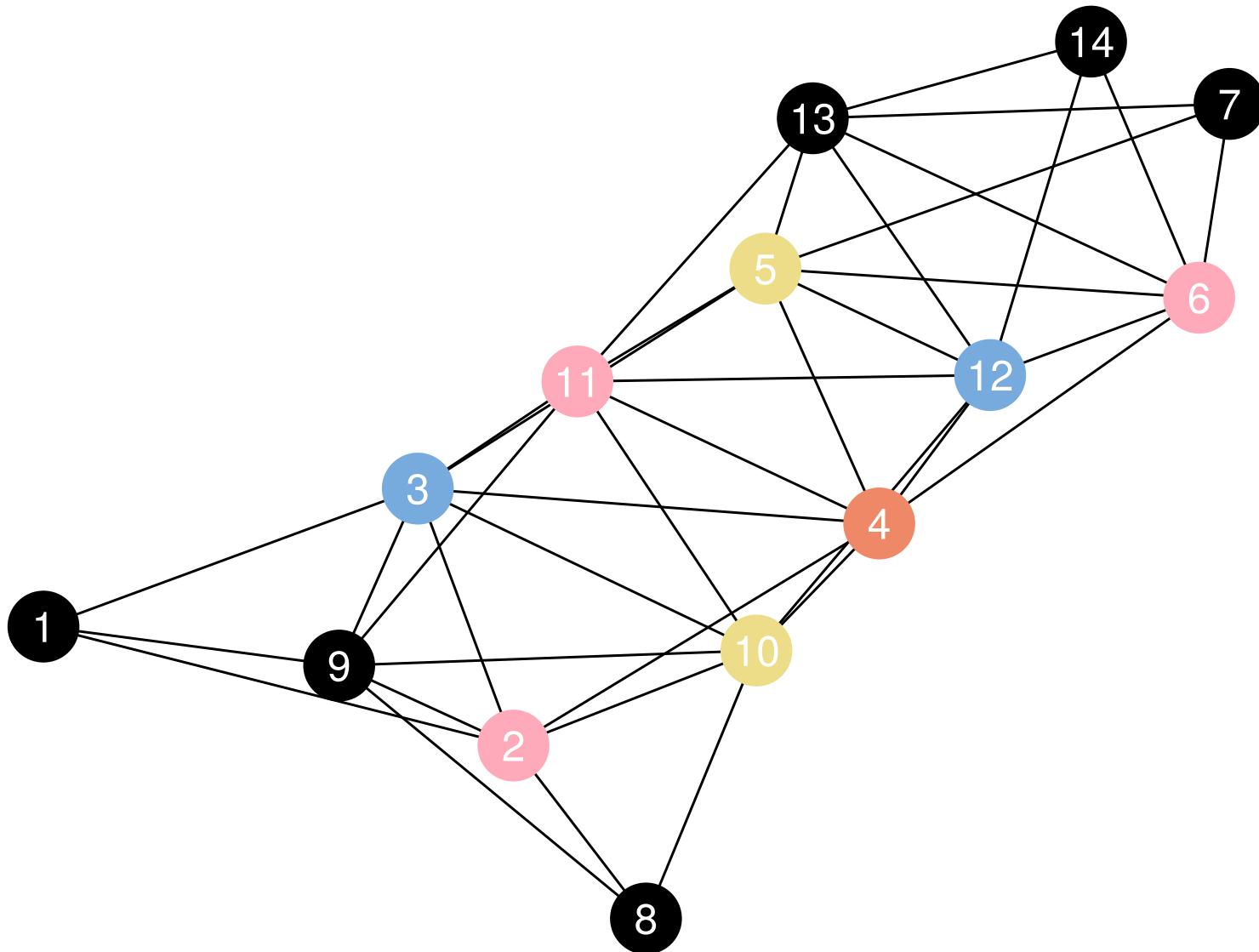


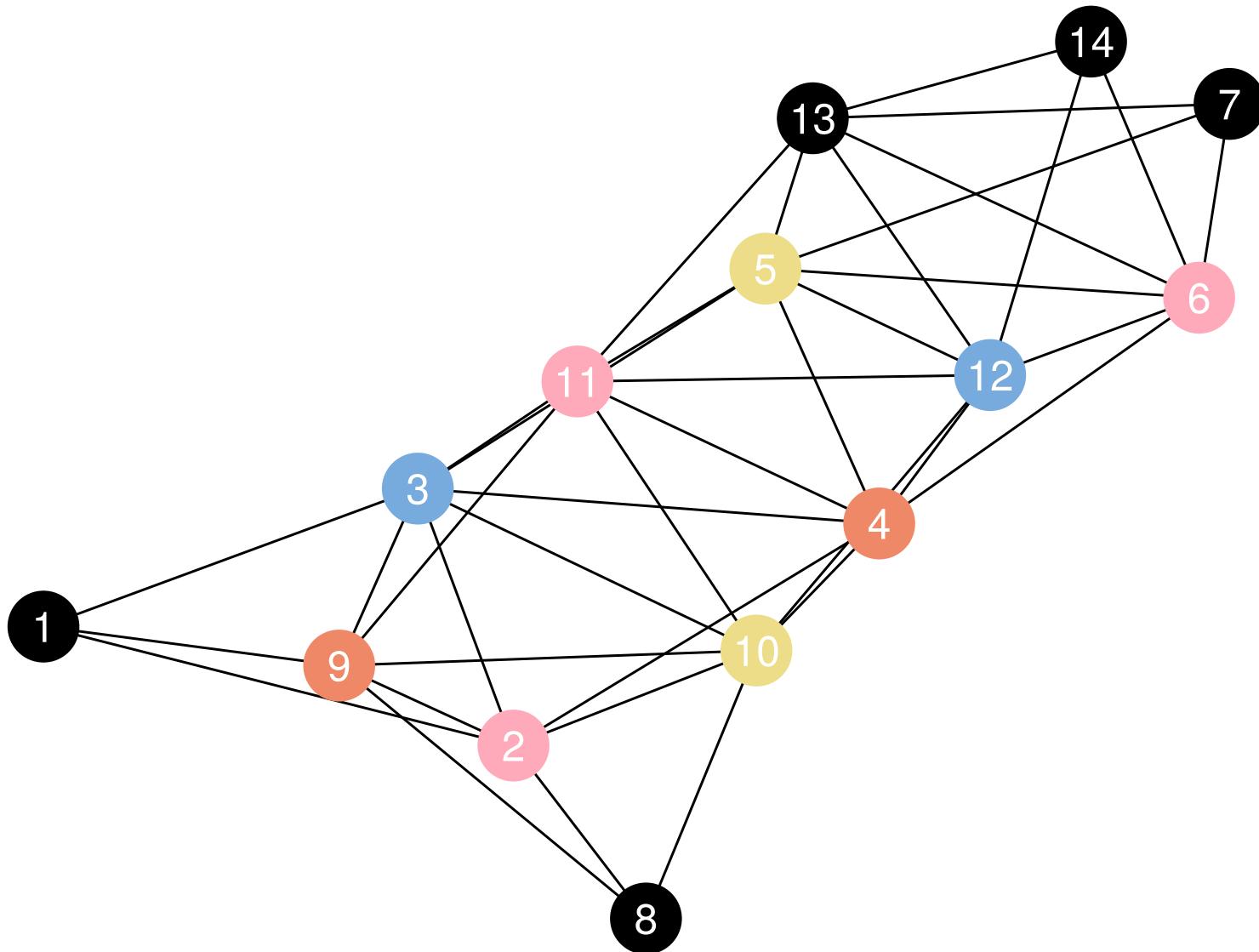


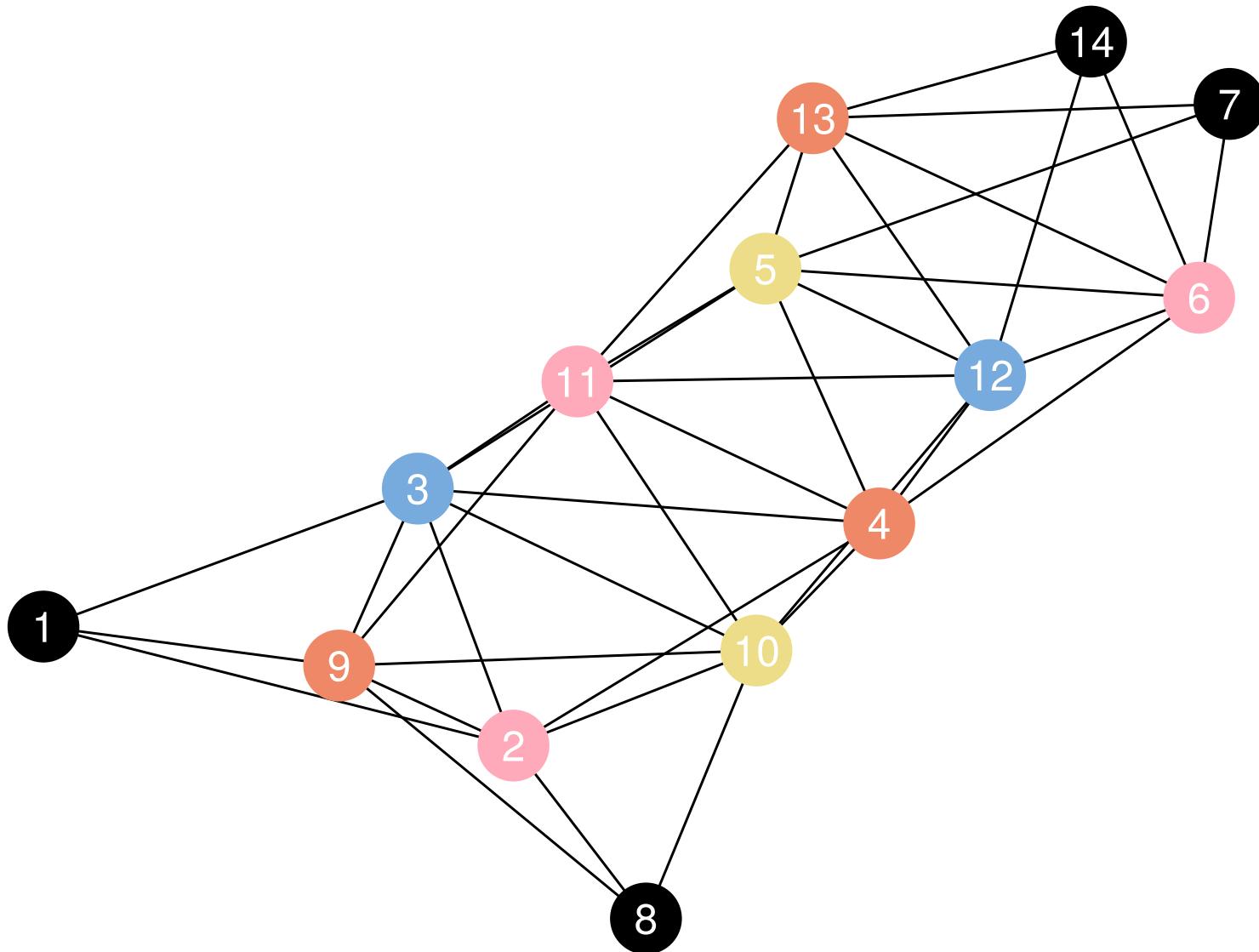


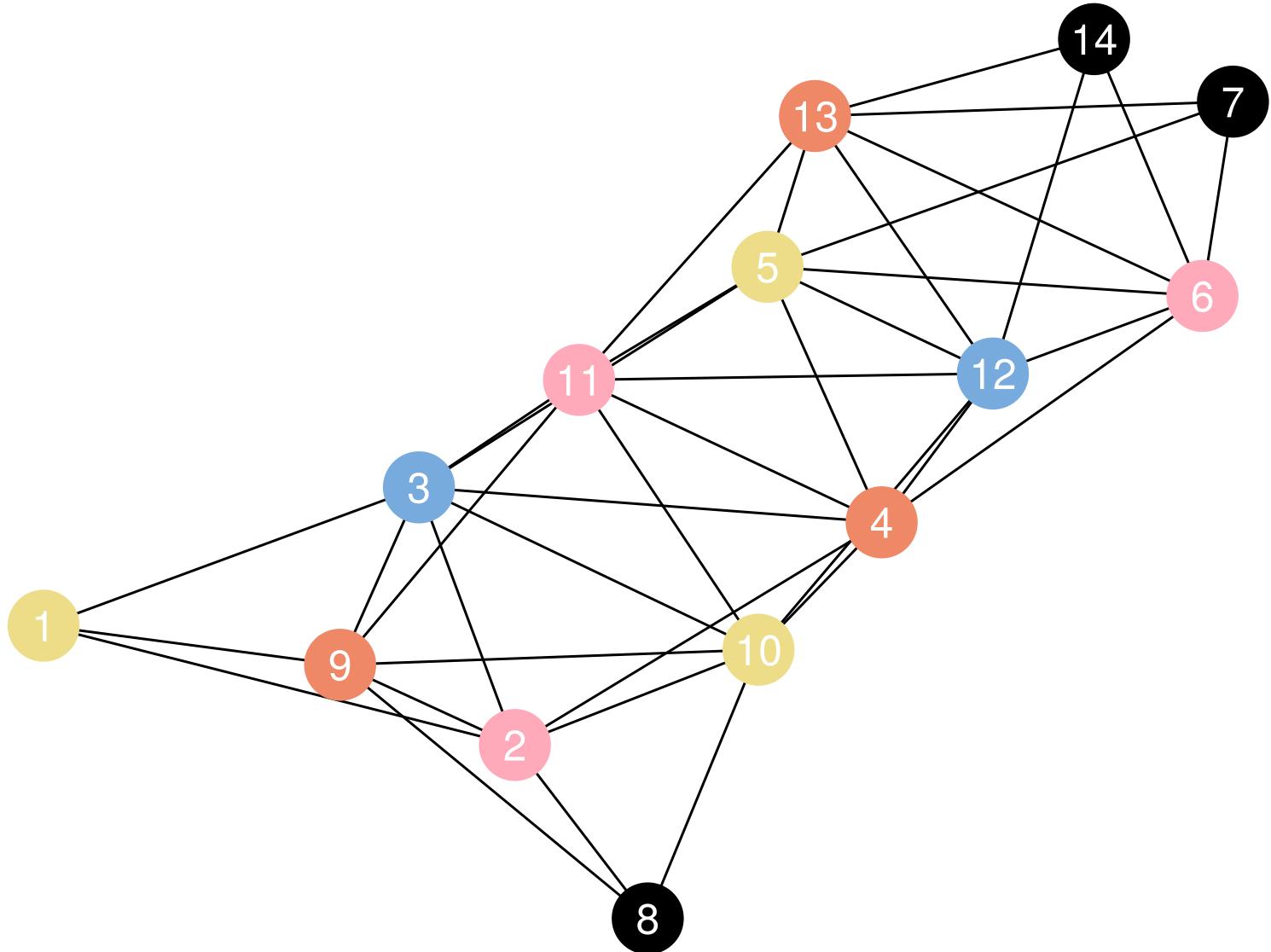


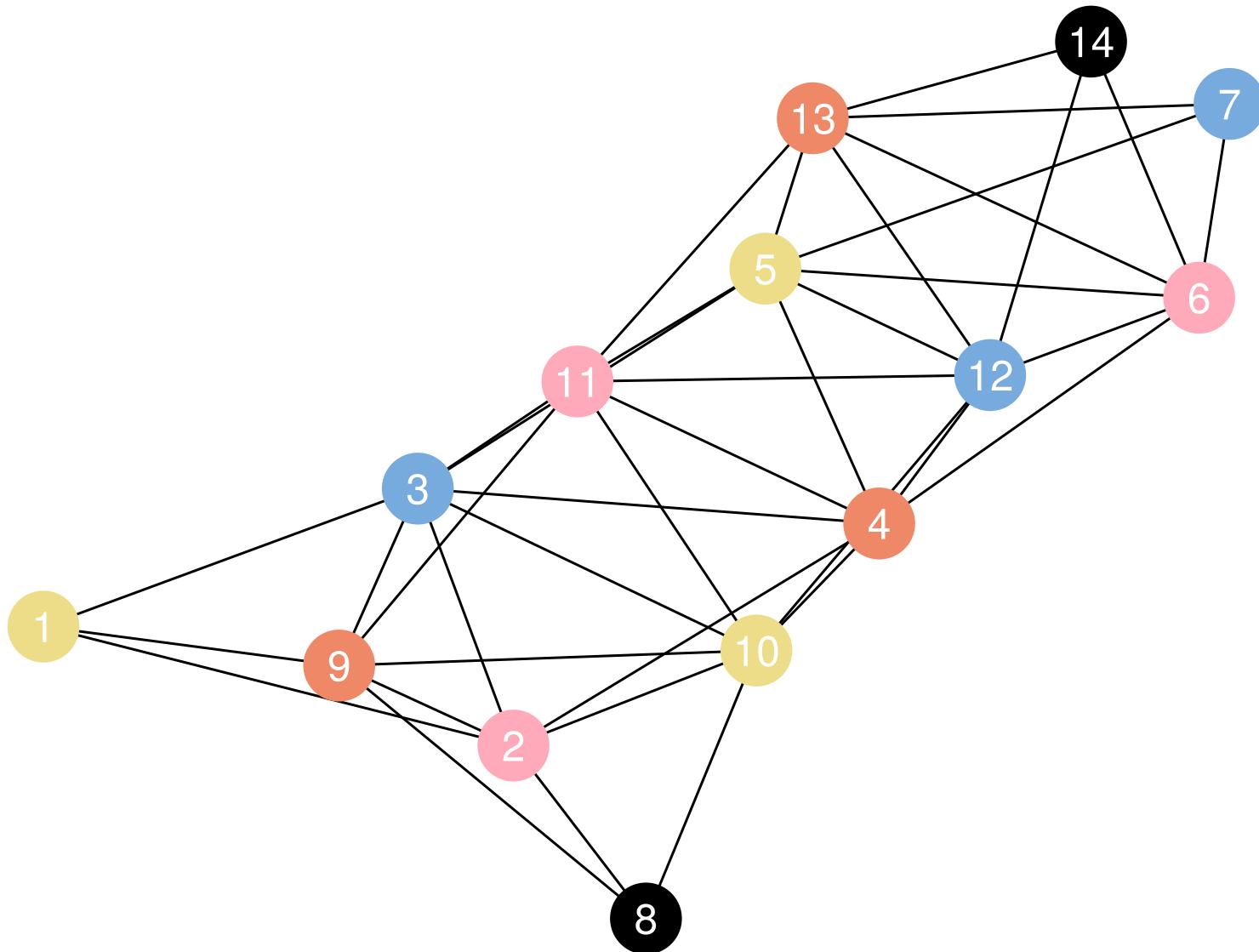


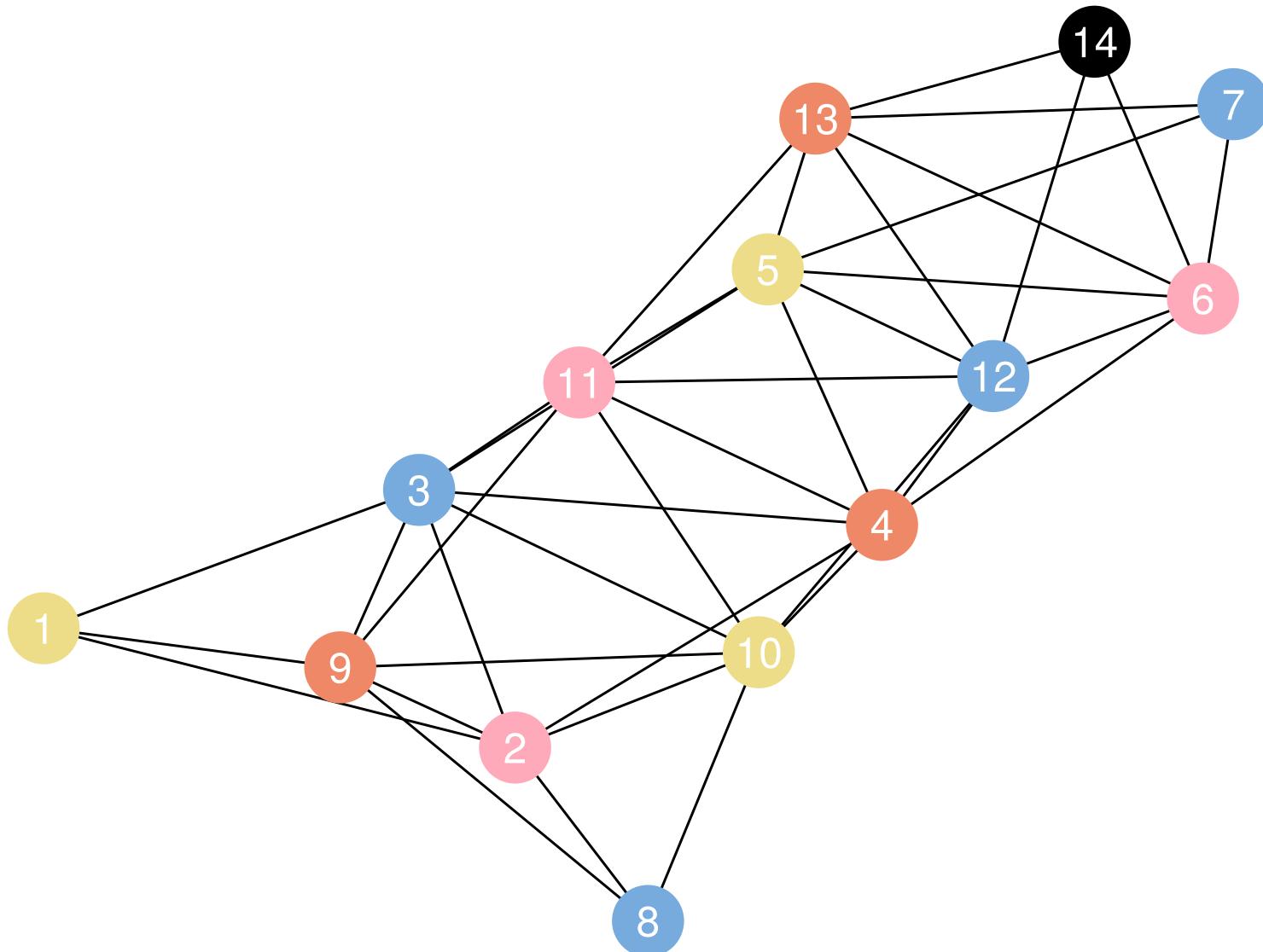


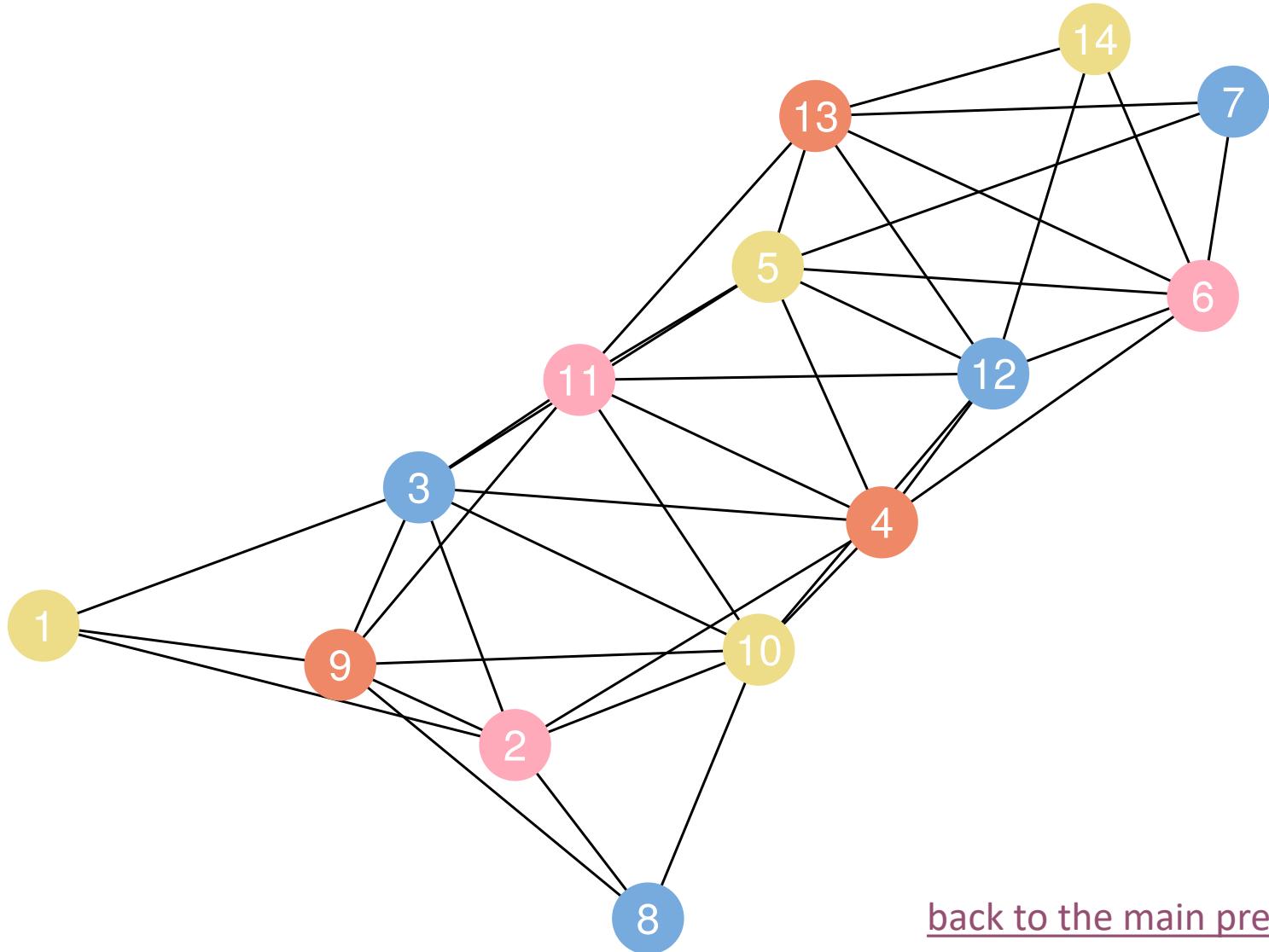












[back to the main presentation :\)](#)

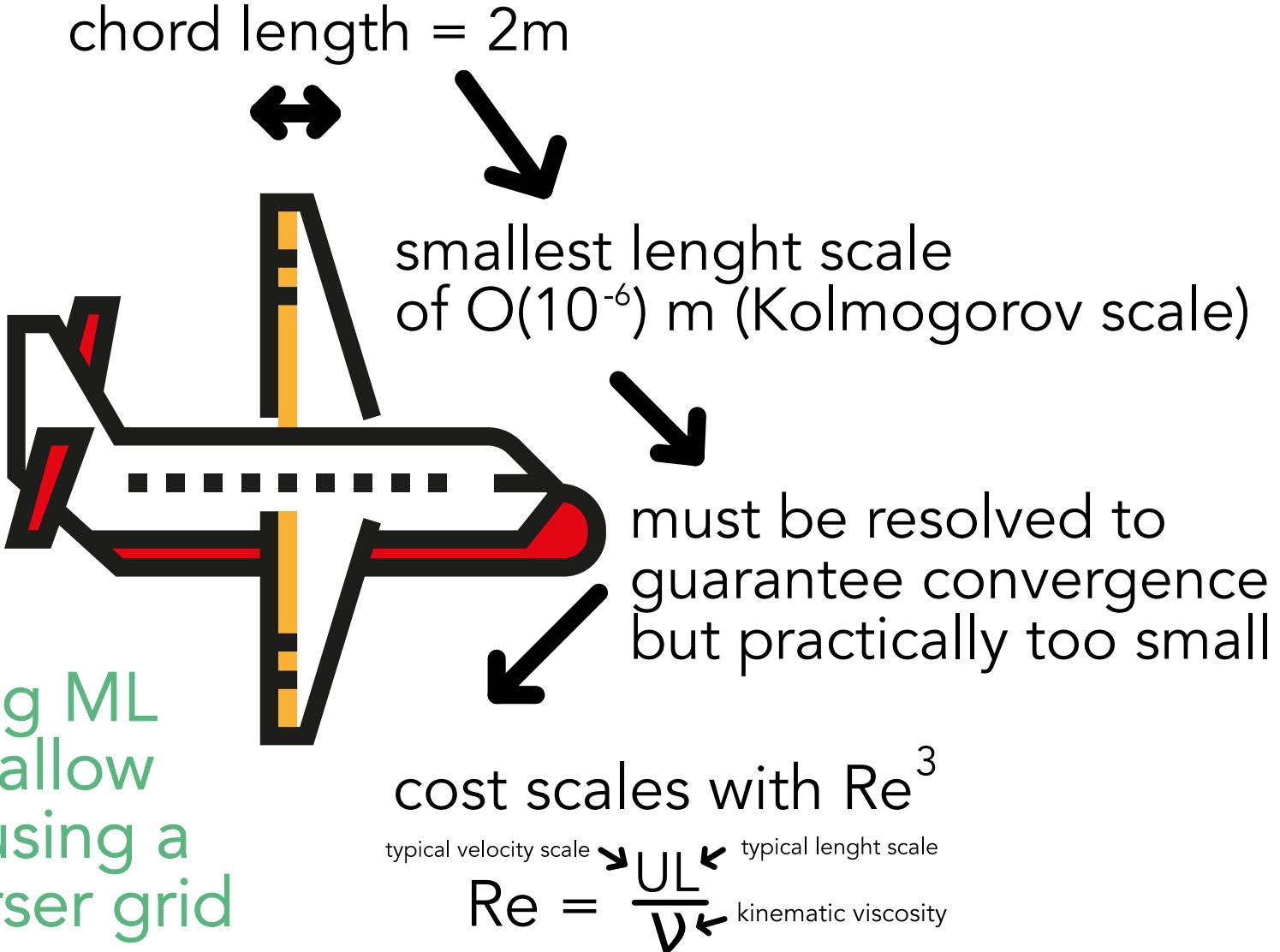
Symplectic integrators and Liouville's theorem

following texts taken from <https://www.av8n.com/physics/liouville-intro.htm>

It is interesting to look at the «area» in phase space occupied by our group of particles. Actually it's not really an area in the Euclidean sense, but rather a bivector in the Clifford Algebra sense. If you're not familiar with bivectors, to a good approximation you can cross out the word bivector every time you see it and replace it with the word “area” – but keep in mind that it is an *oriented* area. Just as two vectors can have the same length but different direction, two bivectors can have the same area and different orientation. In two dimensions, we reckon the bivector as negative or positive, depending on whether we go around the boundary in the clockwise or counterclockwise direction (respectively).

Liouville: For any group of points, the total bivector (in phase space) occupied by the group remains unchanged over time, as the group flows along in accordance with the laws of motion.

Additional figure regarding the outlook



RK45 Fehlberg

Butcher Tableau

		RK-nodes							
		$\alpha_{1,1} \dots \alpha_{1,m}$							
		$\vdots \quad \vdots \quad \vdots$							
		$\gamma_1 \quad \dots \quad \gamma_m$							
		$\alpha_{m,1} \dots \alpha_{m,m}$							
		$\beta_1 \dots \beta_m$							

RK-weights

+ integration up to node

version A
(same k_j)
version B

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

RKF
45

,

result from different versions used for error estimate ϵ



Search or jump to...

/

Pulls Issues Codespaces Marketplace Explore



+



leo1200 / diffeq

Public

Pin

Unwatch 1

Fork 0

Star 0

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

...

master ▾

Go to file

Add file ▾

<> Code ▾

About



Accompanying Notes on a Seminar
Talk on Numerically Solving (Stiff)
Differential Equations

Readme

0 stars

1 watching

0 forks

Releases

No releases published
Create a new release



leo1200 fixed some grammar stuff, added readme ...

6 minutes ago



8



figures fixed some grammar stuff, added readme

6 minutes ago



Numerical_Approa... fixed some grammar stuff, added readme

6 minutes ago



README.md fixed some grammar stuff, added readme

6 minutes ago



efficiently_solving... fixed some grammar stuff, added readme

6 minutes ago



efficiently_solving... fixed some grammar stuff, added readme

6 minutes ago



euler_live.ipynb fixed some grammar stuff, added readme

6 minutes ago



simple_stiff_exampl... fixed some grammar stuff, added readme

6 minutes ago

78

Alternative diffusion stability analysis

Let us make the simple analysis of considering $u_{i+1}^{(j)} = u_{i-1}^{(j)} = 0$ and demanding that the sign of $u_i^{(j+1)}$ and $u_i^{(j)}$ must be the same as of stability. Then

$$\alpha \leq \frac{1}{2} \Leftrightarrow \Delta t \leq \frac{1}{2} \frac{\Delta x^2}{D}$$

Problem of scaling the resolution: If we want to double the spatial resolution, we must quadruple the resolution in time, i. e. in total the 1D simulation is 8x more expensive (in 3D double spatial resolution means 8x more points in a certain volume so 32x more cost in total).

Note on Newton Iteration

Find $\underline{\xi}_{k+1}$ so that the directional derivative along $\underline{\xi}_k - \underline{\xi}_{k+1}$ takes us to zero

$$g(\underline{\xi}_k) - \underline{J}_g(\underline{\xi}_k) (\underline{\xi}_k - \underline{\xi}_{k+1}) = 0$$

Newton vs Quasi-Newton

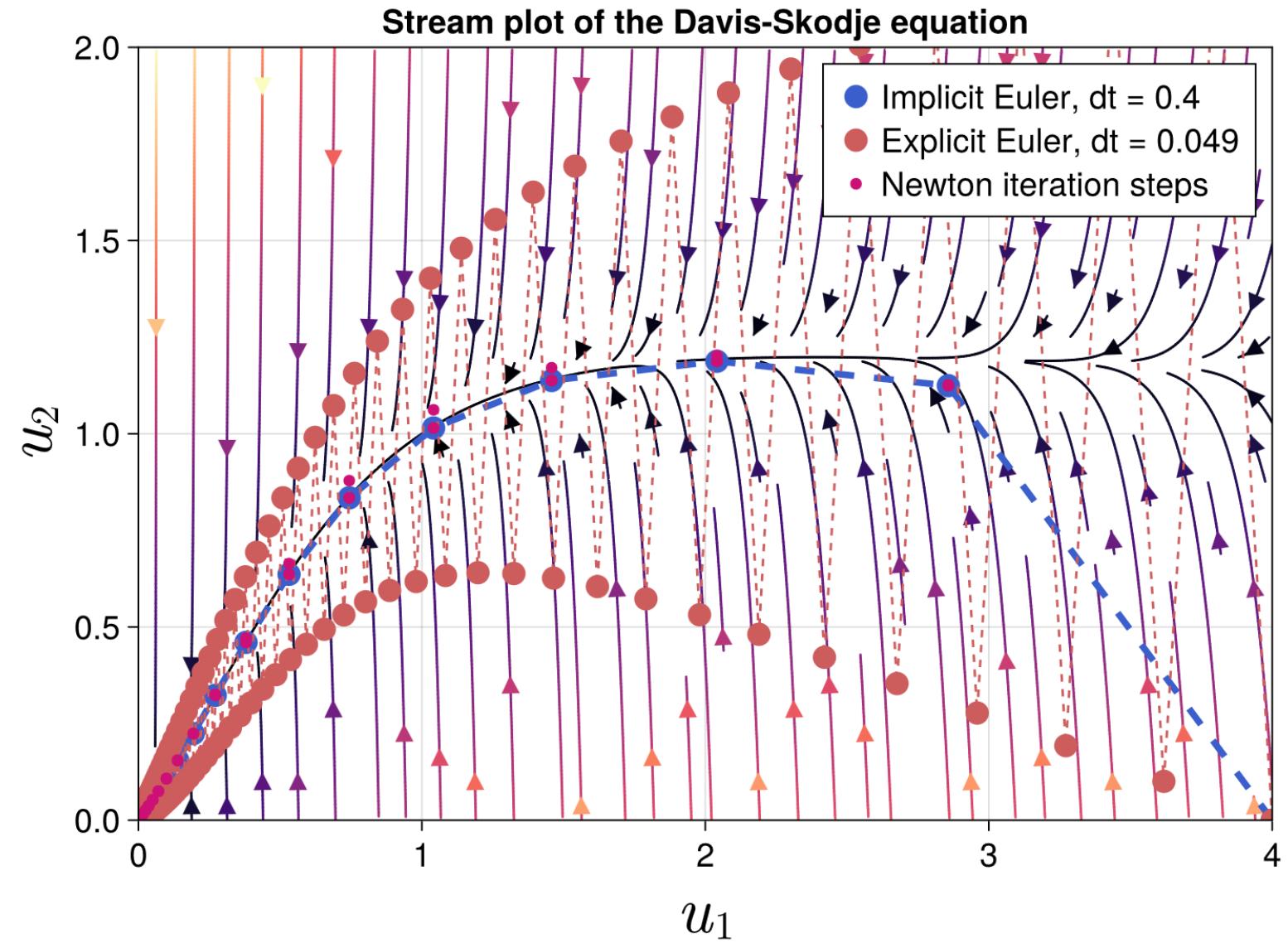
quadratic convergence $q = 2$ in

$$\lim_{n \rightarrow \infty} \frac{|\xi_{n+1} - \underline{y}^{(n+1)}|}{|\xi_n - \underline{y}^{(n+1)}|^q} = \mu$$

(rate of convergence μ), but
 $\mathcal{O}(N^3)$ LU per iteration step

No quadratic convergence,
but $\mathcal{O}(N^3)$ LU only once
per implicit timestep

Davis-Skodje with Explicit Euler



Further left out Notes

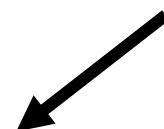
For small Δx the explicit scheme takes excessively many steps.

Remember the discretized differential equation

$$\partial_t u_i = 1 + u_i^2 v_i - 4u_i + \frac{\alpha}{(\Delta x)^2} (u_{i-1} - 2u_i + u_{i+1})$$

$$\partial_t v_i = 3u_i - u_i^2 v_i + \frac{\alpha}{(\Delta x)^2} (v_{i-1} - 2v_i + v_{i+1})$$

possible stiffness indicator



Coefficients of vastly different magnitude occur for small Δx .